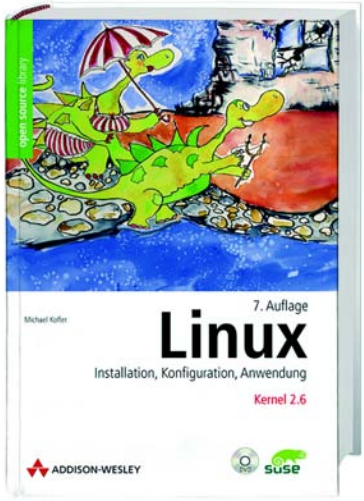
Plone

open source library

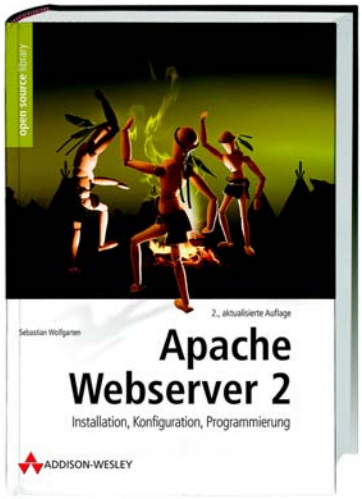
Open Source Software wird gegenüber kommerziellen Lösungen immer wichtiger. Addison-Wesley trägt dieser Entwicklung Rechnung mit den Büchern der **Open Source Library**. Administratoren, Entwickler und User erhalten hier professionelles Know-how, um freie Software effizient einzusetzen. Behandelt werden sowohl Themen wie Betriebssysteme, Netzwerke und Sicherheit als auch Programmierung.

Eine Auswahl aus unserem Programm:



Das Buch gehört für viele Linux-Anwender einfach neben den Linux-PC wie die Installations-CD-ROMs und die Tasse Kaffee. (Linux Enterprise). DAS Standardwerk für Linux-Einsteiger und -Anwender ist jetzt vollständig überarbeitet und aktualisiert: Von den ersten Schritten (Installation) führt es direkt zu den wichtigsten Desktop-Anwendungen (neu: OpenOffice, Gimp 2.0, Digitalkameras einbinden, DVDs brennen). Die Themenschwerpunkte Konfiguration und Server-Konfiguration wurden stark erweitert (neu: WLAN, Firewall und VPN, Apache, PHP und MySQL etc.). Das Buch berücksichtigt die aktuellen Distributionen von SUSE, Red Hat, Fedora, Mandrake und Knoppix. Michael Koflers Bestseller wurden unter anderem mit dem Jolt-Linux Award und dem Linux New Media Award ausgezeichnet und in fünf Sprachen übersetzt.

Michael Kofler
Linux
ISBN 3-8273-2158-1
1320 S.
Euro 59,95 (D), 61,70 (A)



Der mit dem Web tanzt!
Das erfolgreiche Buch über den Apache Webserver 2, jetzt in einer rundum aktualisierten und erweiterten Auflage. Sie erhalten hier das umfassende Wissen zur Installation, Konfiguration und Programmierung des Apache Webserver 2. Ein besonderer Schwerpunkt liegt auf den Themen LDAP, Sicherheit und Programmierung. Weiterhin geht es u.a. um Multiprotokollsupport, Ein- und Ausgabefilter, Mono (ASP.NET), IPv6, SSI, PHP, Perl und SSL. Egal, ob Sie unter Linux, Windows, FreeBSD oder Solaris arbeiten – hier erfahren Sie, wie Sie den Apache 2 effizient einsetzen.

Sebastian Wolfgarten
Apache Webserver 2
ISBN 3-8273-2118-2
912 S., 1 CD
Euro 44,95 (D), 46,30 (A)

PEARSON Education

ADDISON-WESLEY

open source library

Andy McKay

Plone

Leitfaden für Administratoren und Entwickler

Übersetzt von Dinu Gherman, unter fachlicher Mitarbeit von Robert Boulanger, Philipp Auersperg und Georg Bernhard

ADDISON-WESLEY

An imprint of Pearson Education

München • Boston • San Francisco • Harlow, England
Don Mills, Ontario • Sydney • Mexico City
Madrid • Amsterdam

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.

Autorisierte Übersetzung der englischen Originalausgabe *The Definitive Guide to Plone*.

Authorized translation from the English language edition, entitled *The Definitive Guide to Plone* by Andy McKay, published by Apress, Copyright © Andy McKay 2004.

10 9 8 7 6 5 4 3 2 1

07 06 05

ISBN 3-8273-2206-5

ein Imprint der Pearson Education Deutschland GmbH
Martin-Kollar-Straße 10–12, D-81829 München/Germany

Einbandgestaltung: Marco Lindenbeck, webwo GmbH (mlindenbeck@webwo.de)
Fachlektorat: Philipp Auersperg, Robert Boulanger, Georg Bernhard, Blue Dynamics
Lektorat: Boris Karnikowski, bkarnikowski@pearson.de
Herstellung: Monika Weiher, mweiher@pearson.de
Korrektorat: Friederike Daenecke
Satz: reemers publishing services gmbh, Krefeld, www.reemers.de
Druck: Bercker Graphischer Betrieb, Kevelaer
Printed in Germany

Inhaltsübersicht

Inhaltsverzeichnis

Vorwort zur deutschen Ausgabe

Wir stehen am Anfang des Jahres 2005. Um genau zu sein, es ist der 7. Februar 2005. Das heißt, Plone gibt es seit nun etwa fünf Jahren, und ich bin verblüfft, dass ich das Vorwort zur deutschen Übersetzung dieses Buchs schreibe.

In der Geschichte von Plone sind viele der wichtigsten Ereignisse mit Deutschland und seinen Nachbarländern verbunden. Die ersten international besuchten Coding-Veranstaltungen fanden vor fast zwei Jahren im schweizerischen Bern statt. Sieben Monate später hatten wir einen Sprint in einem Schloss im österreichischen Goldegg. Außerdem gibt es in Deutschland eine sehr aktive und effektiv arbeitende Zope- und Plone-User-Group, die DZUG. Europa ist, jedenfalls bisher, frei von Software-Patenten, und es waren deutsche Minister, die E-Mails an uns Plone-Entwickler geschrieben haben, um ihren Dank und ihre Wertschätzung für diese Software auszudrücken. In Europa scheinen der Open Source-Gedanke und Plone stark ausgeprägt zu sein.

Und nun das, Andy McKays Buch, *The Definitive Guide to Plone*, wurde auf Deutsch übersetzt. Es übertrifft einfach meine Erwartungen, wie groß die Zielgruppe von Plone ist. Seien Sie willkommen bei diesem Software-System, in seiner weltweiten Community, und lernen Sie eines der interessantesten verfügbaren Technologienbündel kennen, das Content-Management-System namens Plone!

Andy McKay habe ich auf einer O'Reilly-Konferenz getroffen, ein Jahr, bevor es mit Plone losging. Er war damals ein erfolgreicher Zope-Entwickler und arbeitete bei ActiveState in Vancouver, Kanada. Nachdem Plone einmal abgehoben hatte, war Andy einer derjenigen, die wegen ihrer Consulting-Projekte großes Interesse daran hatten. Er kümmert sich immer auch um das Open Source-Installationsprogramm für Plone unter Windows, betreibt die Website *zopezen.org* und hat viele Patches zu Zope, CMF und Plone beigesteuert.

Und nun hat Andy viel von seiner Zeit dem Schreiben dieses Buchs gewidmet, das nach dem Druck unter der Creative Commons-Lizenz allen öffentlich zur Verfügung steht. Das Buch wird heute in Nordamerika gut verkauft. Mir ist völlig schleierhaft, wie Andy es geschafft hat, das Buch zu schreiben, seine Consulting-Tätigkeit fortzuführen und auch noch mit seiner wunderbaren Frau Danae

am gesellschaftlichen Leben teilzunehmen. Wie viele andere Projekte von Andy ist auch dieses Buch keines, das auf Profit ausgelegt ist. Andy wollte den Leuten zeigen, wie man Plone und die ganzen Technologien dahinter benutzt. Während der letzten drei Jahre sind wir gute Freunde geworden. Wir sind sogar Partner in einer kommerziellen Software- und Consulting-Firma, Enfold Systems, in der wir Plone unter Windows wirksam einsetzen.

Plone begann als spaßige Zusammenarbeit zwischen Alexander Limi aus Norwegen und mir (ich komme aus Houston, Texas) und ist mittlerweile für uns beide wie auch für viele andere zu einer Vollzeit-Hauptbeschäftigung geworden. Aus diesen Ursprüngen wurde Plone größer und größer und hat sich zu einer Erfolgsgeschichte entwickelt. Dabei dürfen wir aber nicht vergessen, aus der Geschichte zu lernen, und wir sollten uns bewusst machen, was Plone zu dem Erfolg gemacht hat, der es heute ist.

Der wichtigste Grund für den Erfolg von Plone ist seine Community. Man vergisst sehr leicht die einfache Formel für eine gute Community: Je mehr man investiert, desto mehr bekommt man zurück. Das wird überdeutlich, wenn man jede Woche neue Komponenten findet, die für Zope/CMF/Plone entwickelt werden. Ich glaube, es gibt zwei Ursachen dafür, warum sich so schnell eine Community um Plone herum gebildet hat: seine einfache Installation (jedenfalls unter Windows ;-) und seine einfache Benutzung. Durch die einfache Installation ergibt sich eine hohe Zahl von Testbenutzern. Der größte Wert liegt aber, wie ich meine, in der einfachen Benutzung. Das habe ich immer wieder in meinen Gesprächen mit Consulting-Firmen und staatlichen Einrichtungen gehört. Dank seiner Lokalisierung, d.h. dadurch, dass die Benutzerschnittstelle in der Muttersprache des Benutzers vorhanden ist, verbreitet sich das Produkt sehr schnell. Heute ist Plone ein weltweites Phänomen, bei dem es jeden Monat neue Übersetzungen, Komponenten und mehr Dokumentation gibt. In Deutschland gibt es sogar ein Online-Magazin, das *Zope Magazine*.

Nun gibt es zu Plone drei englischsprachige Bücher und eines auf deutsch. Auf vielen Open Source-Konferenzen gibt es eine oder mehrere Präsentationen zu Plone, z.B. auf der PyCon (US-Ostküste), O'Reilly (US-Westküste), FISL (Porto Alegre, Brasilien), Solutions Linux (Frankreich), EuroPython (Schweden), dem LinuxTag (Deutschland) und Vancouver Python Workshop (Kanada). Ach ja, die jährliche Plone-Konferenz gibt es auch. Letztes Jahr fand sie im Wiener Volksgarten in einer Diskothek stand, in die sich 350 Teilnehmer in zwei Tracks mit Vorträgen und Tutorien hineinzwängten. Dort fand auch die erste Jahresversammlung der Plone Foundation statt. Ich muss jetzt vieles auslassen, da das Jahr einfach sehr reich an Ereignissen war. Zwischen diesen größeren Veranstaltungen gibt es auch »Sprints«, wo sich 10 bis 30 Entwickler treffen und an einem bestimmten Feature oder einem Teil des Technologiepakets von Plone arbeiten.

Mitglieder der Community, ob alteingesessene oder neue, haben Unmengen von Möglichkeiten, weltweit an der Entwicklung des Systems teilzunehmen.

Dieses Buch hilft Ihnen dabei, schnell Fortschritte mit Plone zu machen. Es gibt keine größere Genugtuung, als ein System wie Plone einzurichten, das dann von Ihnen selbst oder einer Gruppe von Leuten benutzt wird. Während Sie diesen Weg gehen, hoffe ich, dass Sie noch die Zeit haben, Kontakt mit der Community aufzunehmen, ob das nun mit Hilfe der Mailing-Listen, mit IRC, auf Konferenzen oder in lokalen User-Groups geschieht. Die Community bietet eine überwältigende Menge an Unterstützung und Weisheit. Sie ist das größte Gut eines jeden Open Source-Projekts, ja – sie *ist* das Open Source-Projekt. Viele Leute aus dieser Community haben mir erzählt, dass ihnen dieses Buch dabei geholfen habe, das Wissen über ihre fertigen Projektteile an Ihre Kunden weiterzugeben. Es ist die Community in Form von Freiwilligen und Firmen, die ihre Unterstützung angeboten haben, die Plone so weit gebracht hat. Ich hoffe, der Inhalt dieses Buchs hilft Ihnen dabei, sich den Wert von Plone einfach und schnell zu erschließen. Und wer weiß? Vielleicht finden Sie sich selbst als Mitglied der weltweiten Plone-Community wieder, nachdem Sie dieses Buch gelesen haben. Geben auch Sie Ihr Wissen weiter!

Alan Runyan

1 Einführung in Plone

Eine Firma ohne Website ist undenkbar, und die meisten Firmen und Organisationen verfügen gleich über mehrere. Egal, ob es eine externe Site für die Kommunikation mit Kunden oder ein Intranet für die eigenen Mitarbeiter ist oder eine Webseite für direkte Kommunikation mit und Feedback von Kunden – die meisten Websites haben ein Problem, nämlich die Verwaltung ihres Inhalts. Dies ist eine Herausforderung, für deren Lösung eine Organisation sehr viel Zeit investieren und großen Aufwand betreiben muss. Es ist keine leichte Aufgabe, für solche Sites ein mächtiges, aber dennoch flexibles System zu schaffen, das gleichzeitig mit den sich ständig ändernden Anforderungen und mit den sich weiterentwickelnden Bedürfnissen der Firma klarkommt.

Unabhängig von den Anforderungen Ihrer Website und von der Menge an Inhalten oder Benutzern stellt Plone eine benutzerfreundliche, mächtige Lösung dar, mit der Sie über das Web ganz einfach beliebige Inhalte hinzufügen und bearbeiten können sowie Navigations- und Suchmöglichkeiten für diese Inhalte hinzufügen können, aber auch Sicherheitsmaßnahmen und Workflow-Einstellungen dafür vornehmen können.

Mit Plone können Sie fast jede Website erstellen und sie dann ganz leicht aktualisieren. Damit können Sie inhaltsreiche Sites schnell erstellen und sich einen Wettbewerbsvorteil verschaffen. Und schließlich ist vielleicht das Beste an diesem System, dass es Open Source und gratis ist. Aufgrund seiner beeindruckenden Liste von Eigenschaften ist es vergleichbar mit, wenn nicht sogar besser als viele geschlossene Content-Management-Systeme, die viele Hunderttausende von Dollar kosten.

Folgendes schrieb Mike Sugarbaker in seinem Bericht von der Open Source Content Management-Konferenz (OSCOM) im Jahre 2002 für die Site *Mindjack* (*http://www.mindjack.com/events/oscom.html*):

> *»Ich werde nicht die vollständige Liste aller konkurrierenden Open-Source-Management-Frameworks behandeln. Ich lasse alle Verfolger aus: Der Sieger heißt Plone. Dieses Produkt, das auf dem sechs Jahre alten Web Application Framework Zope basiert, war das Paket mit den meisten Werkzeugen, der höchsten Professionalität, der größten Zugkraft, und vor allem mit dem größten Rummel.«*

Sie finden die Plone-Website unter *http://www.plone.org*, wie in Abbildung 1.1 zu sehen ist. Um Plone einfach mal auszuprobieren, ist eine Demonstrations-Site unter *http://demo.plone.org* verfügbar. Dort können Sie einfach und schnell Inhalte über das Web bearbeiten. Das heißt, Sie können Ereignisse und Dokumente hinzufügen, Bilder hochladen und alles über das Framework verarbeiten, das Plone zur Verfügung stellt.

Wenn Sie einen Schritt weiter gehen möchten, können Sie unter *http://www.objectis.org* eine kostenlose Hosting-Dienstleistung von Objectis in Anspruch nehmen und eine eigene nichtkommerzielle Zope- oder Plone-Site aufsetzen und betreiben. Dort verfügen Sie dann über ein eigenes Portal mit vollen Manager-Rechten und vielen zusätzlich installierten Produkten. Derzeit betreibt Objectis fast 6000 solcher Portale.

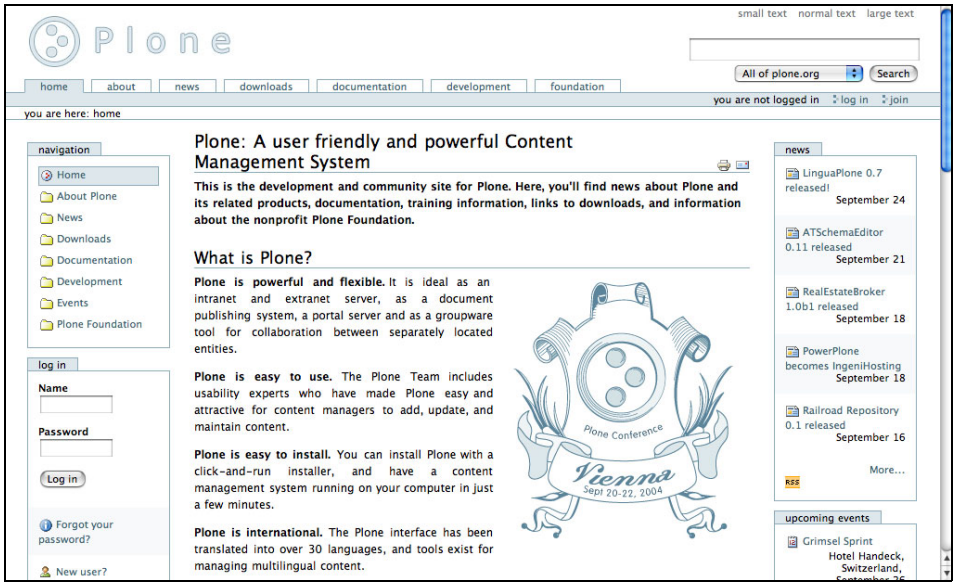


Abbildung 1.1: Die Plone-Website Plone.org-Website

1.1 Was ist ein Content-Management-System?

Eine einfache Definition eines Content-Management-Systems (CMS) ist die, dass es ein System für die Verwaltung von Inhalten ist. Da diese Definition wenig hilfreich ist, möchte ich eine vollständigere Erklärung in kleineren Teilen angeben. Ich beginne mit einer allgemeinen Definition von *Inhalt (Content)*: ein Inhalt ist eine Dateneinheit, die um dazugehörige Informationen erweitert ist. Eine Dateneinheit können dabei eine Webseite, ein bevorstehender Termin, ein Microsoft

Word-Dokument, ein Bild, ein Videofilm oder irgendwelche Daten sein, die in der Organisation, die mit dem System arbeitet, von Bedeutung sind.

All diese Einheiten werden *Inhalte* genannt, und sie alle verfügen über ähnliche Attribute, z.B. dass sie von bestimmten Benutzern hinzugefügt und bearbeitet und auf verschiedene Weisen veröffentlicht werden müssen. Diese Attribute werden von einem *Workflow*-System gesteuert. Dahinter verbirgt sich eine regelbasierte Logik, mit der die Verwaltung der Inhalte beschrieben wird.

Historisch betrachtet, kann man einen Unterschied zwischen Content-Management- und Dokumentenmanagement-Systemen beobachten, aber heute sind beide überwiegend zu einem konvergiert. Der wesentliche Unterschied liegt in den verwalteten Einheiten: Meistens werden unter *Inhalten* beliebige Daten verstanden, während *Dokumente* Dinge sind, die von Menschen z.B. mit Software wie Microsoft Office geschrieben und bearbeitet werden. Nehmen Sie etwa ein Buch: Ein Buch besteht aus vielen Dateneinheiten und erfordert möglicherweise eine Verwaltung, die sich leicht von der allgemeiner Inhalte unterscheidet. In den meisten Fällen ist dieser Unterschied jedoch recht klein, und Produkte wie Plone können die kleinen Bestandteile eines größeren Inhalts verwalten und zusammensetzen.

Durch die Allgegenwart des Webs werden heute viele CMS als Web-CMS bezeichnet, sei es, weil sie über eine webbasierte Schnittstelle verfügen oder weil sie sich auf ein webbasiertes Übertragungssystem per Internet oder Intranet stützen. Plone bietet beides: eine Verwaltungsschnittstelle und ein Übertragungssystem auf Basis des Webs.

Folgendes ist eine mögliche Definition eines CMS (*http://www.contentmanager.eu.com/history.htm*):

> *Ein CMS ist ein Werkzeug, das es vielen verschiedenen (zentralen) technischen und (dezentralen) nichttechnischen Mitarbeitern ermöglicht, eine Vielzahl von Inhalten (z.B. Text, Grafik, Video usw.) zu erstellen, zu bearbeiten, zu verwalten und schließlich zu veröffentlichen, und zwar unter zentralen Randbedingungen bzgl. Regeln, Prozessen und Workflow, die ein konsistentes und gültiges Aussehen im Web sicherstellen.*

1.2 Brauchen Sie ein Content-Management-System?

Auch wenn es nicht der einzige Vorteil eines CMS ist, so ist der offensichtlichste, dass eine Website damit einfach zu koordinieren ist. Stellen Sie sich eine Situation vor, in der eine Person, der *Webmaster*, eine Website koordiniert, sei es ein Intranet oder eine externe Site. Die Benutzer liefern Inhalte in den verschiedensten Formaten, und der Webmaster macht daraus benutzbare Webseiten, indem er sie in

HTML (Hypertext Markup Language) umwandelt. Wenn ein Benutzer diese Seiten ändern muss, schickt er diese Änderungen an den Webmaster, und so weiter.

Daraus ergeben sich für die Organisation viele Probleme, wobei das größte ist, dass der gesamte Inhalt von einer Person abgewickelt wird – ein offensichtlicher Engpass. Diese eine Person kann nur ein bestimmtes Arbeitspensum erledigen, und wenn sie krank wird oder die Firma verlässt, geht viel Produktivität dabei verloren, einen Ersatz zu finden. Der Vorgang der Veröffentlichung kann sehr frustrierend sein, wenn E-Mails zwischen Webmaster und Benutzern ausgetauscht werden, die versuchen, ihre Inhalte zu veröffentlichen.

Was man braucht, ist ein System, das Folgendes leistet:

- **Trennung des Inhalts einer Seite von seiner Präsentation:** Wenn der eigentliche Inhalt von der Art seiner Präsentation getrennt wird, muss der Autor des Inhalts nichts über HTML oder darüber wissen, wie die Seite ausgegeben wird. Tatsächlich könnten auf den gleichen Inhalt verschiedene Templates angewandt werden, inklusive anderer Formate als HTML, z.B. PDF (Portable Document Format) oder SVG (Scalable Vector Graphics). Wenn Sie das Aussehen der Site ändern möchten, müssen Sie nur noch das eine Template ändern und nicht mehr alle Inhalte.
- **Erlaubnis für bestimmte Benutzer, Inhalte hinzuzufügen und zu ändern:** Wenn bestimmte Benutzer Inhalte leicht hinzufügen und ändern können, müssen diese nicht mehr zum Webmaster oder ans Webteam geschickt werden. Stattdessen kann der Benutzer, der eine Seite erstellen möchte, das einfach tun und die Seite so lange bearbeiten, wie es nötig ist.
- **Anwendung von Regeln, die angeben, wer was und wann veröffentlichen kann:** Ihre Geschäftslogik soll evtl. verhindern, dass jeder Inhalte auf Ihrer Website veröffentlicht. So könnten z.B. Marketing-Mitarbeiter Inhalte auf der Seite der Pressemitteilungen veröffentlichen, aber nicht auf den technischen Seiten.
- **Anwendung von Geschäftslogik anf Inhalte:** Wenn ein Marketing-Mitarbeiter eine Pressemitteilung erstellt, muss sie evtl. jemand aus der Rechtsabteilung überprüfen. In diesem Fall wird das Dokument von einem Revisionsprozess erfasst, der sicherstellt, dass es nicht öffentlich wird, bevor die Prüfung erfolgt ist.
- **Informationen können intelligent gesucht und indiziert werden:** Da das CMS strukturierte Informationen über den Inhalt verwalten kann (z.B. den Namen des Autors, das Datum der Veröffentlichung bzw. von Änderungen, Kategorien usw.), kann es Listen von Inhalten nach Autor, Zeit usw. erstellen. Es kann auch Suchfunktionen anbieten, die wesentlich intelligenter und nützlicher sind als eine rein textuelle Suche.

Obwohl dieses Beispiel Vorteile darstellt, die besonders für große Organisationen von Bedeutung sind, profitieren Organisationen aller Größen davon. Tatsächlich können kleine Organisationen, die üblicherweise keinen Vollzeit-Webmaster beschäftigen können, besonders große Vorteile durch ein solches System erhalten. Durch die Einführung eines CMS können Sie all diese und noch weitere Probleme lösen.

Das Schlüsselkonzept eines jeden CMS besteht in der klaren Trennung seiner wesentlichen Bestandteile: Sicherheit, Workflow, Templates usw. So sind z.B. die Templates, mit denen ein Eintrag dargestellt wird, von seinem Inhalt getrennt. Dadurch können Sie seine Darstellung leicht ändern.

1.3 Ein erster Blick auf Plone

Plone ist Open Source-Software, die unter der GPL (General Public License) lizenziert wird. Dies ist eine häufig verwendete Lizenz, die es jedem gestattet, die Software gratis einzusetzen. Weitere Informationen zur GPL finden Sie auf der Website der Free Software Foundation unter *http://www.gnu.org*. Sie können alle Aspekte des Quellcodes von Plone untersuchen und für Ihre Anwendung anpassen. Sie müssen keine Lizenzgebühren bezahlen, es gibt keine Lizenz, die irgendwann abläuft, und der gesamte Code ist offen gelegt. Diese Open-Source-Philosophie bedeutet, dass Plone bereits eine große Anzahl von Benutzern hat, und es gibt Legionen von Entwicklern, Usability-Experten, Übersetzern, Fachautoren und Grafikdesignern, die in der Lage sind, *an* Plone selbst zu arbeiten. Wenn Sie auf Plone setzen, sind Sie nicht auf eine Firma angewiesen. Es gibt etwa ein Dutzend Firmen, die Dienstleistungen zu Plone anbieten.

In naher Zukunft kann es durchaus sein, dass Plone unter die kommerziell vorteilhaftere LGPL gestellt wird oder sogar unter einem dualen Lizenzmodell verfügbar sein wird.

1.3.1 Verpackung

Zu Plone gibt es einfache Installationsprogramme für Windows, Linux und Macs. Produkte und Zusätze von Dritten verfügen ebenfalls über Installationsprogramme. Das Angebot qualitativ hochwertiger Releases für solche Produkte erleichtert ihre Installation und Verwaltung. Außerdem beinhaltet jedes neue Release einen Migrationspfad und Aktualisierungen, mit denen Ihre Plone-Site weiter funktioniert und auf dem neuesten Stand bleibt.

1.3.2 Internationalisierung

Die gesamte Benutzerschnittstelle von Plone wurde in über 25 Sprachen übersetzt, darunter Koreanisch, Japanisch, Französisch, Spanisch und Deutsch. Wenn Sie Ihre eigene Übersetzung hinzufügen möchten, so ist das einfach möglich (siehe Kapitel 4).

1.3.3 Benutzerfreundlichkeit

Plone bietet eine hervorragende Benutzerschnittstelle, die ein hohes Maß an Benutzerfreundlichkeit und Zugänglichkeit bietet. Dabei geht es nicht nur darum, hübschen HTML-Code darzustellen, sondern das betrifft fundamentale Teile von Plone. Die Schnittstelle von Plone ist kompatibel mit den US-Standards WAI-AAA und U.S. Section 508, die in Wirtschaft und Verwaltung gelten. Dadurch können mit Plone erstellte Sites auch von Menschen mit Sehbehinderungen benutzt werden. Zusätzlich bietet das den überraschenden, aber damit in Zusammenhang stehenden Vorteil, dass Ihre Seiten von Suchmaschinen wie Google besser indiziert werden.

1.3.4 Flexibles Aussehen

In Plone ist der Inhalt von den eigentlichen Templates (dem Aussehen, oft auch *Skins* genannt) getrennt, mit denen Inhalte dargestellt werden. Dieses Aussehen wird mit dem großartigen HTML-Templatingsystem namens *Zope Page Templates* sowie einer großen Anzahl von Beschreibungen in CSS (Cascading Style Sheets) beschrieben. Mit nur wenig Wissen über Plone können Sie ein anderes Aussehen erreichen, zwischen mehreren Erscheinungsweisen wählen und die Präsentation Ihrer Website völlig umgestalten.

1.3.5 Registrierung und Personalisierung

Plone beinhaltet ein vollständiges Benutzerregistrierungssystem. Benutzer können sich auf einer Plone-Site mit eigenen Benutzernamen und -passwörtern sowie mit beliebigen anderen Angaben registrieren, die Sie über die Benutzer hinzufügen möchten. Anschließend können Sie die gesamte Benutzerschnittstelle für die jeweiligen Benutzer personalisieren. Außerdem können Sie mit Hilfe von Add-Ons bereits vorhandene Angaben über Ihre Benutzer aus vielen Stellen verwenden, z.B. aus relationalen Datenbanken, aus LDAP (Lightweight Directory Access Protocol), Active Directory und aus anderen Quellen. Kapitel 9 behandelt die Registrierung und Konfiguration von Benutzern.

1.3.6 Workflow und Sicherheit

Der Workflow steuert die Logik, nach der Inhalte auf der gesamten Site verarbeitet werden. Diese Logik können Sie über das Web mit grafischen Werkzeugen konfigurieren. Administratoren von Websites können diese so kompliziert oder einfach machen, wie sie es möchten. Man kann Benachrichtigungswerkzeuge hinzufügen, z.B. zum Verschicken von E-Mails oder Instant-Messages an Benutzer. Kapitel 8 behandelt den Workflow sehr detailliert.

Für jedes Stück Inhalt einer Plone-Site können Sie Listen für die Zugangskontrolle einrichten, die bestimmen, welche Benutzer Zugang zu diesem Inhalt haben und was sie damit anstellen können. Können sie ihn bearbeiten, ansehen oder kommentieren? All das lässt sich über das Web konfigurieren (siehe Kapitel 9).

1.3.7 Erweiterbarkeit

Da Plone Open Source ist, kann es leicht verändert werden. Sie können fast jeden Aspekt von Plone ändern und konfigurieren, um es an Ihre Bedürfnisse anzupassen. Zahllose Pakete für Plone bieten eine große Bandbreite an Möglichkeiten für kleinere Sites sowie für große Unternehmen. Depots für solche Gratisprodukte und -erweiterungen finden Sie unter *http://www.plone.org*. Mit Entwicklungswerkzeugen wie Archetypes (siehe Kapitel 13) können Sie Plone-Code sehr leicht über das Web oder mit UML-Werkzeugen (Unified Modeling Language) erstellen. Kapitel 10 behandelt die Integration von Plone mit Enterprise-Lösungen wie LDAP, Apache, Microsoft Internet Information Services (IIS), Macromedia Dreamweaver usw.

1.3.8 Anpassung von Inhalten

Die Benutzer einer Plone-Site können alle möglichen Inhalte hinzufügen, und die hinzugefügten Daten sind nicht in ihrer Menge oder Art eingeschränkt. Plone-Entwickler können ihre eigenen Inhaltstypen erstellen und hinzufügen, so dass fast jede Art von Inhalt verwaltet werden kann. Beschränkungen ergeben sich nur aus Ihrer Vorstellungskraft. In den Kapiteln 11 und 12 beschreibe ich, wie man Inhaltstypen anpassen kann. Kapitel 13 führt Sie in Archetypes ein, ein sehr mächtiges System für die Erstellung von Inhaltstypen, das ohne Programmierung auskommt. Sie können z.B. neue Inhaltstypen mit UML-Werkzeugen erstellen.

1.3.9 Dokumentation

Das Plone-Projekt verwaltet auch Dokumentation, darunter dieses Buch, das unter der Creative Commons-Lizenz veröffentlicht wird. Der beste Ausgangspunkt zur verfügbaren Dokumentation ist *http://www.plone.org/documentation*.

1.3.10 Community

Eines der besten Dinge an Plone ist seine Community von Entwicklern und Firmen, die Plone unterstützen und weiterentwickeln. Dank mehr als 60 Entwicklern weltweit, die in irgendeiner Weise daran beteiligt sind, ist es fast immer möglich, einen Plone-Entwickler zu finden, der gewillt und fähig ist, Ihnen zu helfen. Alan Runyan, Alexander Limi und Vidar Andersen haben angefangen, Plone zu entwickeln, aber es entwickelte sich schnell zu einem aktiven Projekt, nachdem weitere Entwickler sich daran beteiligten. Die Beiträge dieser Entwickler haben das Produkt Plone zu dem gemacht, was heute verfügbar ist.

Exkurs: Beispiele für Plone-Sites

Es existieren viele Plone-Sites. Bei manchen davon ist das wegen ihres Aussehens offensichtlich, bei anderen nicht. Die folgende Liste enthält lediglich Beispiele ganz verschiedener Sites:

- **Plone** (*http://www.plone.org*): Die definitive Plone-Site, die alles enthält, was Sie jemals über Plone wissen möchten, inklusive Dokumentation und Installationspaketen zum Herunterladen.
- **Plone Demo Site** (*http://demo.plone.org*): Eine Demo-Site für Plone mit einer großen Anzahl von Produkten.
- **Zope.org** (*http://www.zope.org*): Die definitive Zope-Community-Site, die von der Zope Corporation betrieben wird. Sie enthält eine Menge Informationen über Zope und ist wahrscheinlich eine der umfangreichsten Community-Sites zu Plone.
- **Liquidnet** (*http://www.liquidnet.com*): Die Website einer Investment-Firma, auf der viel Flash verwendet wird.
- **Design Science Toys** (*http://www.dstoys.com*): Eine Site, die Spielzeug verkauft und ein Open-Source-E-Commerce-Produkt für Plone verwendet.
- **Give Kids the World** (*http://www.gktw.org*): Eine Site für die Beschaffung von Fördermitteln für Kinder, geschrieben in Plone unter Verwendung von viel Flash.

- **Propane** (*http://www.usepropane.com*): Eine viel besuchte Kunden-Website mit ausgeklügelter Suche, vollständig in Plone implementiert.
- **Maestro Headquarters** (*http://mars.telascience.org*): Eine NASA-Website, mit Informationen über die Mars-Rover. Ihre Benutzerschnittstelle wird in Kapitel 7 beschrieben.

Weitere Plone-Sites werden unter *http://www.plone.org/about/sites* aufgeführt, darunter solche, die über eine ganz andere Benutzerschnittstelle verfügen. Wenn man nichts über die Entwicklung dieser Sites weiß, wird man kaum erkennen, dass diese Sites Plone verwenden.

1.4 Beitragen zur Plone-Entwicklung

Obwohl Plone über eine beachtliche Liste von Eigenschaften verfügt, ist die Liste seiner gewünschten Eigenschaften noch viel beachtlicher. Aus diesem Grund sucht das Projekt ständig nach Leuten, die ein wenig von ihrer Zeit dafür aufbringen können.

Da Plone auf den Endbenutzer zugeschnitten ist, gibt es zum Glück Bedarf an einem breiten Spektrum von Fähigkeiten. Helfer aus ganz verschiedenen Gebieten sind willkommen, nicht nur Programmierer oder Webentwickler. Plone benötigt Entwickler von Benutzerschnittstellen, Usability-Experten, Grafikdesigner, Übersetzer, Autoren und Tester. Den aktuellen Entwicklungsstand finden Sie auf der Plone-Website unter *http://plone.org/development*, und am leichtesten kann man mitmachen, indem man eine Mailing-Liste abonniert oder die Entwickler auf einem IRC-Kanal (Internet Relay Chat) besucht.

1.5 Was sind Zope und das CMF?

Plone baut auf Zope und CMF (Content Management Framework) auf. Um Plone zu verstehen, müssen Sie Zope und das CMF als darunter liegende Architektur verstehen. Aus diesem Grund werde ich in diesem Abschnitt diese beiden Dinge vorstellen und erklären, wie sie mit Plone zusammenspielen.

Zope ist ein mächtiger und flexibler Open-Source-Web-Application-Server, der von der Zope Corporation (*http://www.zope.org*) entwickelt wurde. Zope wurde als eigenständiges CMS entwickelt, hat aber mit der Zeit den Anforderungen seiner Benutzer nicht genügt. Dann hat die Zope Corporation das CMF als Open Source-Projekt entwickelt. Das CMF bietet Entwicklern die Werkzeuge zum

Erstellen von komplexen CMS. Es bietet Workflow, Site-Aussehen und weitere Funktionen.

Das CMF ist ein Framework für ein System, d.h., es bietet die Werkzeuge, mit denen Entwickler ein Produkt erstellen können, statt ein fertiges System, das die Benutzer sofort einsetzen könnten. Plone verwendet diese und viele weitere Eigenschaften und verbessert sie zu einem qualitativ hochwertigen Produkt für den Benutzer. Plone ist eine Schicht über dem CMF, die eine Anwendung darstellt, die unter Zope läuft. Sie müssen das CMF verstanden haben, um Plone zu verstehen. Für die meisten Verwaltungsfunktionen wird die Verwaltungsschnittstelle von Zope benötigt, und für die Entwicklung mit Plone muss man wissen, Zope und dessen Objekte funktionieren.

Dieses Buch behandelt Zope nicht in seiner ganzen Tiefe, sondern gibt Ihnen genug Informationen darüber, damit Sie Aufgaben in Plone erledigen können. Durch die Lektüre dieses Buches erhalten Sie ausreichende Informationen, damit Sie fast alles in Plone anpassen und ändern können. Wenn Sie weitere Informationen zu Zope benötigen, empfehle ich Ihnen *The Zope Book*, das ursprünglich bei New Riders erschienen ist. Danach ist es online verfügbar gemacht worden und wird von Mitgliedern der Zope-Community aktualisiert. Es ist gratis online unter *http://www.zope.org/Documentation/Books/ZopeBook/2_6Edition* verfügbar.

Sowohl Zope als auch das CMF sind Schlüsseltechnologien, ohne die Plone nicht existieren würde. Das Plone-Team ist vor allem der Zope Corporation zu sehr großem Dank für die Weitsicht verpflichtet, Zope und das CMF als Open Source verfügbar zu machen. Die Liste derer, denen ich dort sowie in den CMF-Communities danken möchte, ist sehr lang. Danke an alle Beteiligten!

1.6 Was ist Python?

Zope ist in Python geschrieben, einer objektorientierten Open-Source-Programmiersprache, die mit Perl oder Tcl vergleichbar ist. Man muss Python nicht kennen, um Plone zu benutzen oder gar einfache Administrationsaufgaben damit zu erledigen. Zum Anpassen von Produkten und beim Scripting von Plone ist jedoch ein wenig Python notwendig.

Tommy Burnette, ein erfahrener technischer Direktor bei Industrial Light & Magic, sagt Folgendes über Python (*http://www.python.org/Quotes.html*):

> *Python spielt in unserem Produktionsprozess eine Schlüsselrolle. Ohne Python wäre ein Projekt von der Größenordnung von Star Wars: Episode II sehr schwer zu realisieren gewesen. Python hält alles zusammen, vom Rendering sehr vieler Objekte über die Stapelverarbeitung bis hin zum Compositing.*

Wenn Sie vorhaben, irgendetwas Anspruchsvolles mit Plone zu machen, sollten Sie sich zwei bis drei Tage Zeit nehmen und die Grundlagen von Python lernen. Danach werden Sie nicht nur Plone wesentlich besser anpassen können, sondern Sie werden auch allgemein mit Objekten vertraut sein und damit, wie sie mit der Plone-Umgebung zusammenspielen. Es würde den Rahmen dieses Buches sprengen, Ihnen Python beizubringen. Daher setze ich voraus, dass Sie über Grundkenntnisse in Python verfügen. Diese Grundkenntnisse sind ausreichend, um mit diesem Buch zu arbeiten und eine Plone-Installation leicht anzupassen.

Glücklicherweise ist Python eine sehr leicht zu erlernende Programmiersprache. Ein erfahrener Programmierer braucht durchschnittlich einen Tag, um damit produktiv zu werden. Frischgebackene Programmierer brauchen etwas länger. Wenn Sie Plone mit den Installationsprogrammen unter Windows oder auf Macs installieren, ist die passende Version von Python darin enthalten. Sie können Python separat für fast jedes Betriebssystem unter *http://www.python.org* herunterladen.

Die beste Art, sich mit Python vertraut zu machen, ist die, es in einem Kommandointerpreter auszuprobieren. Sollten Sie eine Windows-Installation von Plone haben, dann gibt es im Startmenü einen Link zu Pythonwin, einer Python-IDE (Integrated Development Environment). Gehen Sie zu START – PROGRAMME – PLONE – PYTHONWIN (siehe Abbildung 1.2).

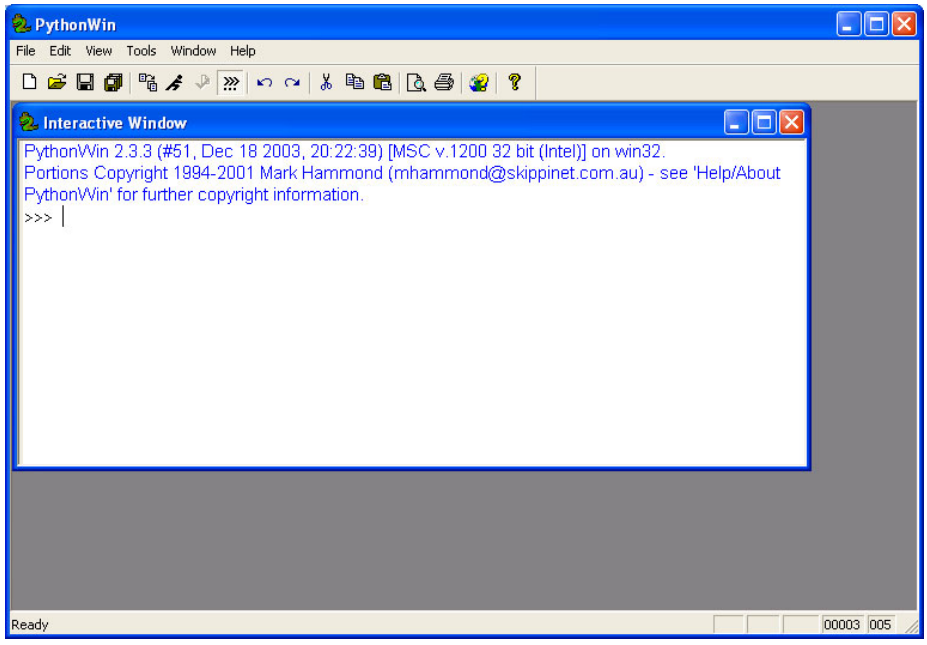


Abbildung 1.2: Der Python-Prompt unter Windows

Unter Linux und Mac OS X wird bei der Eingabe von **python** normalerweise der Python-Interpreter gestartet:

```
$ python Pyython 2.3.2 (#1, Oct  6 2003, 10:07:16)
[GCC 3.2.2 20030222 (Red Hat Linux 3.2.2-5)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Da Python eine interpretierte Sprache ist, müssen Sie nicht das ganze Python-Script kompilieren und ausführen, sondern können einzelne Zeilen an den Interpreter übergeben, während Sie diese schreiben. Das macht aus dem Interpreter ein überraschend nützliches Werkzeug zum Testen und zur Fehlersuche in Code. Im Interpreter hat jede Zeile, die auf eine Eingabe wartet, das Präfix >>>.

Das einfachste »Hello World«-Programm sieht z.B. wie folgt aus:

```
>>> print "Hello, world!"
Hello, world!
>>>
```

Um den Interpreter zu beenden, geben Sie unter Linux [Strg]+[D] (drücken Sie die Taste [D], während Sie [Strg] gedrückt halten) bzw. [Strg]+[Z] unter Windows ein (Sie können das später auch für eine fortgeschrittenere Interaktion mit Zope und Plone verwenden). Normale Python-Scripts können Sie ausführen, indem Sie diese an den Interpreter übergeben. Mit dem folgenden Script namens `hello.py`:

```
print "Hello, world!"
```

können Sie folgenden Befehl ausführen:

```
$ python hello.py Hello, world!
```

Die Python-Website unter *http://www.python.org* verfügt über eine hervorragende Dokumentation, besonders das Tutorium. Auch die folgenden Bücher bieten einen guten Überblick zu Python:

- ***Dive Into Python*** (Apress, 2004): Basierend auf Mark Pilgrims beliebtem webbasierten Tutorium bietet dieses Buch seinen Lesern eine schnelle Einführung in die Sprache Python. Dies ist ein tolles Buch, das für erfahrene Programmierer gedacht ist.
- ***Learning Python, Second Edition*** (O'Reilly, 2003): Dieses Buch behandelt die Version 2.3 von Python und bietet einen guten Überblick zu Python und all seinen neuen Eigenschaften. Es eignet sich gut für relativ neue Programmierer.

- ***Practical Python*** (Apress, August 2002): Diese sehr praxisorientierte Einführung zu Python bietet einen Überblick über die vielen Eigenschaften der Sprache. Die Leser können ihr Wissen sofort in die Praxis umsetzen, indem sie an zehn interessanten Projekten arbeiten. Dazu zählen ein webbasiertes schwarzes Brett und eine File-Sharing-Anwendung mit einer grafischen Benutzerschnittstelle.
- ***Python Essential Reference, Second Edition*** (Sams, 2001): Ein Referenzbuch, das eine sehr gute Übersicht aller wesentlichen Bibliotheken und Funktionen bietet. Dies ist ein exzellentes Buch für erfahrene Programmierer.
- ***Objektorientierte Programmierung mit Python*** (Mitp, 2004): Ein fundiertes Lehr- und Arbeitsbuch von Michael Weigend.
- ***Python für Kids*** (Mitp, 2003): Ein unterhaltsames Einsteigerbuch von Gregor Lingl, das durchaus nicht nur für Kinder zu gebrauchen ist.
- ***Python kurz & gut*** (O'Reilly 2002): Ein kleines praktisches Buch von Mark Lutz mit dem Essentiellen, was man über Python wissen muss, leider nicht mehr durchgehend aktuell.
- ***Einführung in Python*** (O'Reilly, 2000): Ein gut lesbares Einsteigerbuch von den erfahrenen Autoren Mark Lutz und David Ascher, leider nur in der ersten und nicht in der verbesserten zweiten Auflage auf deutsch übersetzt.

1.6.1 Typographische Konventionen

In diesem Buch werden folgende typographischen Konventionen verwendet:

- **Kursiv:** Neue Begriffe werden *kursiv* dargestellt. (Anhang C enthält ein umfangreiches Glossar, in dem alle Akronyme definiert werden.)
- **Fett:** Falls es im Text Anweisungen gibt, die etwas beinhalten, das Sie auf Ihrer Tastatur eingeben sollen, werden diese Worte **fett** gedruckt.
- **Code-Schrift:** Eine `nichtproportionale Schrift` wird bei Dateinamen, Verzeichnispfaden, Code, Variablen und URLs (Uniform Resource Locators) verwendet.
- **Kapitälchen:** KAPITÄLCHEN heben Menüoptionen, Tab- und Buttonbeschriftungen hervor.

Dieses Buch enthält viele Bildschirmdarstellungen von Zope und Plone. Da Plone ein sich schnell weiterentwickelndes Produkt ist, können die Bildschirmdarstellungen leicht von der Version der Software abweichen, die Sie verwenden. Diese Unterschiede sollten jedoch gering sein und Ihr Verständnis von dem System nicht beeinträchtigen.

1.7 Verwendete Software

Für dieses Buch werden die im Folgenden genannten Software-Versionen verwendet. Auch wenn dieses Buch mit genau diesen Versionen im Hinterkopf geschrieben ist, sollte die gesamte Software auf diesen wie auch auf kommenden Versionen funktionieren.

Beim Schreiben dieses Buches war Plone 2.0 die aktuellste Version von Plone. Es ist die zweite große Version dieser Software, und sie bietet, verglichen mit der Version 1.0, viele neue Eigenschaften, darunter die Verwaltung von Gruppenbenutzern, eine neue Schnittstelle sowie eine verbesserte Zope-Distribution. Es wird Ihnen wärmstens empfohlen, neue Projekte mit 2.0 oder höher zu beginnen, anstatt ältere Versionen zu benutzen.

Plone 2.0 ist von folgenden Versionen abhängig: Zope 2.7, CMF 1.4.2 und Python 2.3.3. Alle Code-Beispiele in diesem Buch wurden speziell so entworfen, dass sie nicht von diesen Versionen oder von einem bestimmten Betriebssystem abhängig sind. Es kann jedoch Situationen geben, in denen das nicht der Fall ist. Für daraus entstehende Unannehmlichkeiten möchte ich mich entschuldigen.

1.8 Buchlizenz

Die Idee zu diesem Buch hatte ursprünglich eine Gruppe von Plone-Benutzern, die eine qualitativ hochwertige Dokumentation erstellen wollten. Die erste Version dieses Buchs haben wir auf der Plone-Website als Open-Source-Dokumentationsprojekt veröffentlicht. Alle Inhalte, die auf der Plone-Website hinzugefügt wurden, standen unter der Open-Publication-Lizenz.

Das wachsende Interesse an Plone ließ ein kommerzielles Buch vernünftig erscheinen, und so haben Apress und ich im Sommer des Jahres 2003 dieses Buch begonnen. Ich habe einiges an Material vom alten Buch mit Erlaubnis der ursprünglichen Autoren verwendet. Nach dem Wechsel zu Plone 2 habe ich große Mengen an neuem Material hinzugefügt. Dieses Buch wird nun unter der Creative-Commons-Lizenz veröffentlicht, die die Wiederverwendung dieser Arbeit gestattet, sofern der ursprüngliche Autor erwähnt wird. Für kommerzielle Zwecke dürfen Sie dieses Werk jedoch nicht verwenden. Weitere Informationen zu der Lizenz finden Sie unter *http://creativecommons.org/licenses/by-nc-sa/1.0/*.

1.9 Schreiben Sie uns Ihre Meinung

Sollten Sie Anmerkungen, Kritik oder Anregungen zu diesem Buch haben, schicken Sie diese bitte an *info@pearson.de*. Wir freuen uns über Ihre Rückmeldung und antworten Ihnen so schnell wie möglich.

2 Plone installieren

Dieses Kapitel erklärt, wie man Plone auf einer Vielzahl von Plattformen installiert und grundlegende Konfigurationseinstellungen daran vornimmt. Wenn Sie Plone nur ganz schnell testen möchten, gehen Sie das am einfachsten zur Demo-Site unter *http://demo.plone.org*, wo Sie Inhalte sofort hinzufügen und ändern können, ohne etwas installieren zu müssen.

Anders als bei den anderen Kapiteln macht es keinen großen Sinn, dieses Kapitel von vorn bis hinten durchzulesen. Ich habe es nach Betriebssystemen unterteilt, damit Sie nur die für Sie notwendigen Abschnitte lesen müssen, um Plone zu installieren. Plone lässt sich auf allen Plattformen installieren, die Zope unterstützt: Windows, Mac OS X, Linux, die meisten Unix-Versionen und Solaris.

Ein leistungsfähiger Computer führt natürlich dazu, dass auch Plone eine bessere Leistung zeigt. Plone ist ein komplexes System, das gewisse Ansprüche an die Verarbeitungsgeschwindigkeit und den Hauptspeicher stellt. Bei einem System in Produktion wird allgemein empfohlen, keinen Rechner mit weniger als einem 2 GHz schnellen Prozessor und weniger als 1 Gbyte RAM einzusetzen, falls es eine große Website enthält. Für kleinere Sites funktioniert Plone aber auch bei einer 500-MHz-CPU und 64 Mbyte Hauptspeicher. Für detailliertere Informationen über Plones Performance, Caching und Optimierungsmöglichkeiten lesen Sie bitte Kapitel 14. Eine Grundinstallation von Plone benötigt etwa 50 Mbyte auf der Festplatte. Falls Sie bereits Zope oder Python installiert haben, wird wesentlich weniger Platz benötigt, etwa 2 Mbyte. Sie müssen auch die Objektdatenbank von Plone berücksichtigen, die fast beliebig groß werden kann, je nachdem, welche Datenmengen Sie darin speichern.

Um Plone zu benutzen, brauchen Sie einen Webbrowser mit Zugriff auf den Server. Wenn sich Benutzer auf Ihrer Site anmelden wollen, dann müssen diese Cookies eingeschaltet haben. JavaScript wird nicht verlangt, bietet aber reichhaltigere Benutzungsmöglichkeiten. Da Plone ausgiebigen Gebrauch von Cascading Style Sheets (CSS) macht, stellen moderne Browser die Plone-Schnittstelle reichhaltiger und attraktiver dar, aber funktionieren sollte sie in jedem vernünftigen Browser.

Ich empfehle, einen der folgenden Browser zu verwenden:

- Microsoft Internet Explorer 5.5 oder höher
- Netscape 7.0 oder höher (sowie jegliches Mozilla-Derivat)
- Opera 7.0 oder höher
- Konqueror 3.0 oder höher
- Safari 1.0 oder höher

Plone funktioniert in den folgenden Browsern ebenfalls, auch wenn es evtl. anders als im Original aussieht:

- Netscape 4.x
- Microsoft Internet Explorer 5.0
- Microsoft Internet Explorer 4.0
- Konqueror 2.x
- Lynx (textbasiert)
- w3m (textbasiert)
- AWeb
- Links (textbasiert, optional mit Grafik)
- Alle Browser, die eine Grundmenge an HTML (Hypertext Markup Language) sowie Cookies für Formulareingaben verstehen, darunter die meisten mobilen Browser und solche auf PDAs (Personal Digital Assistant)

2.1 Plone unter Windows installieren

Am einfachsten kann man Plone installieren, wenn man das Windows-Installationsprogramm von Plone verwendet, das die Installation automatisiert. Zu der Installation gehören zusätzliche Pakete und Optionen, eine vorkonfigurierte Datenbank, die Einrichtung von Diensten sowie Python-Pakete für Windows. Sie können dieses Installationsprogramm unter *http://www.plone.org/download* herunterladen.

2.1.1 Das Installationsprogramm verwenden

Das Installationsprogramm wurde unter Windows 9x, ME, NT 3.51+, 2000 und XP getestet, sollte aber auch unter anderen Windows-Versionen funktionieren. Sie sollten Administratorzugang zum gewünschten Rechner haben, da das Pro-

gramm versucht, in der Windows-Registry Dienste einzurichten und Einstellungen vorzunehmen. Falls Sie bereits Zope oder Python installiert haben, sollten Sie vielleicht den Quellcode separat installieren, um Plattenplatz zu sparen.

Bevor Sie Plone installieren, sollten Sie darauf achten, ob weitere Webserver gerade laufen. Neuere Windows-Versionen installieren und starten beispielsweise automatisch IIS (Microsoft Internet Information Services), der Port 80 abhört. Das Installationsprogramm startet Plone auf den Ports 80 und 8080. Am einfachsten testet man, ob schon jemand Port 80 verwendet, indem man in einem Browser die Adresse *http://127.0.0.1/* eingibt und wartet, ob eine Seite gefunden wird. Diesen Webserver können Sie deaktivieren oder aber die Ports für Plone ändern. Siehe dazu den Abschnitt »Konfigurieren des Webservers« weiter unten in diesem Kapitel. Falls Sie Plone hinter IIS oder Plone und IIS gleichzeitig auf dem gleichen Server betreiben möchten, erfahren Sie in Kapitel 14 mehr dazu. Im Moment ist es am einfachsten, diesen Webserver zu deaktivieren.

Nachdem Sie das Installationsprogramm heruntergeladen haben, beginnen Sie die Installation mit einem Doppelklick darauf (siehe Abbildung 2.1).

Abbildung 2.1: Start des Plone-Installationsprogramms

Das Installationsprogramm führt die bei der Software-Installation üblichen Schritte aus. Klicken Sie auf WEITER, um fortzufahren, oder auf ABBRECHEN. Das Programm erlaubt Ihnen die Auswahl eines Installationsverzeichnisses, wobei die Voreinstellung `C:\Programme\Plone 2` ist (siehe Abbildung 2.2).

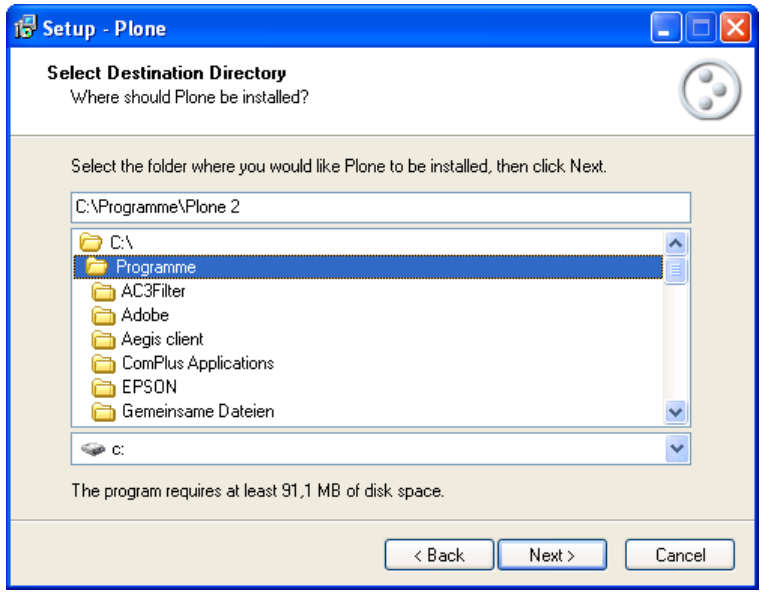


Abbildung 2.2: Wahl eines Verzeichnisses

Wenn Sie zum Passwort-Bildschirm gelangen (siehe Abbildung 2.3), müssen Sie einen Benutzernamen und ein Passwort eingeben. Dadurch wird für Sie ein Benutzer erstellt, und es wird eine Plone-Site unter diesem Benutzer angelegt. Oftmals erstellen die Leute dafür einen Benutzer namens *admin* oder so ähnlich. Diesen Benutzernamen und dieses Passwort benötigen Sie später, d.h., Sie sollten sich beides merken. Sollten Sie das Passwort verlieren, können Sie jedoch später ein neues erstellen.

Die Installation dauert etwa fünf Minuten, je nachdem, wie schnell Ihr Rechner ist. Am Ende der Installation werden einige Aufgaben erledigt, z.B. die Kompilierung aller Python-Dateien und die Einrichtung der Datenbank. Sobald alles beendet ist, erscheint eine Meldung, um Sie davon in Kenntnis zu setzen (siehe Abbildung 2.4).

Um Plone zu starten, gehen Sie zum Plone-Controller, indem Sie auf START – PROGRAMME – PLONE – PLONE klicken. Der Controller ist eine Anwendung mit einer hübschen Oberfläche zum Starten und Anhalten von Plone. Er beginnt mit einer Statusseite, auf der Sie Ihre Plone-Installation ganz leicht starten oder anhalten können (siehe Abbildung 2.5).

Wie in Abbildung 2.5 zu sehen ist, zeigt dieser Bildschirm den Status Ihrer Plone-Installation an. Plone startet nicht automatisch, sondern Sie müssen auf START klicken, um es zu starten. Danach müssen Sie evtl. eine Minute warten, bis der Startvorgang abgeschlossen ist (siehe Abbildung 2.6).

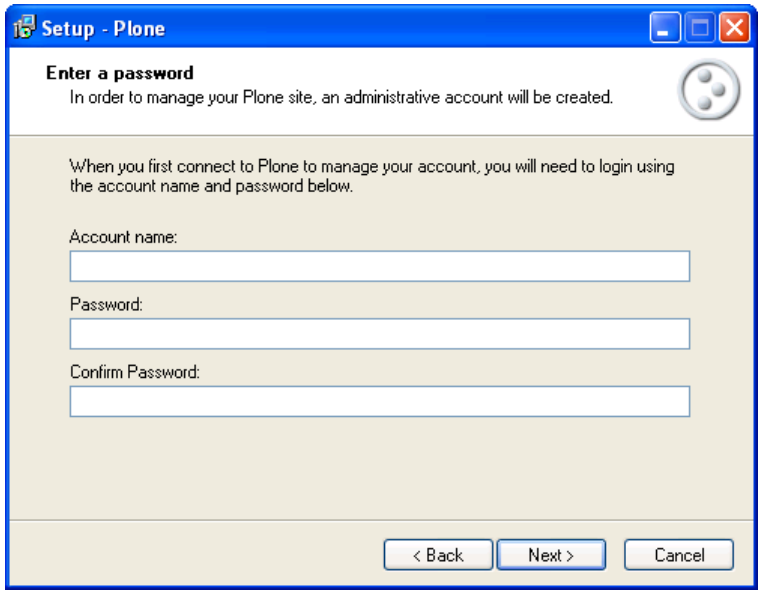


Abbildung 2.3: Eingabe eines Benutzernamens und -passworts

Abbildung 2.4: Letzter Einrichtungsbildschirm

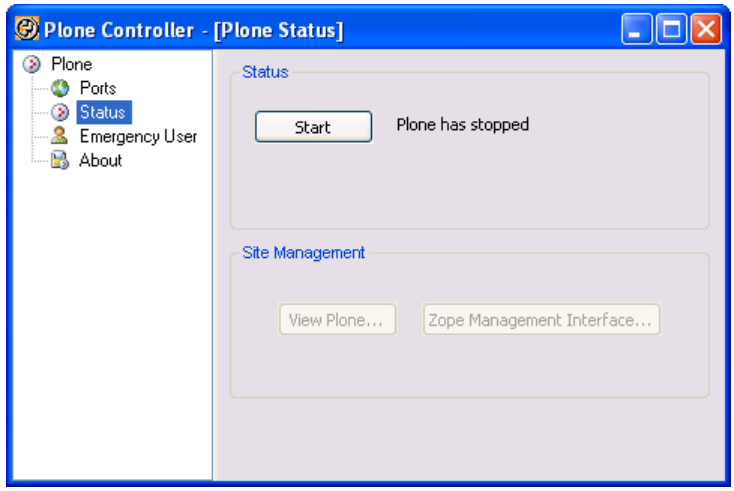


Abbildung 2.5: Plone läuft nicht.

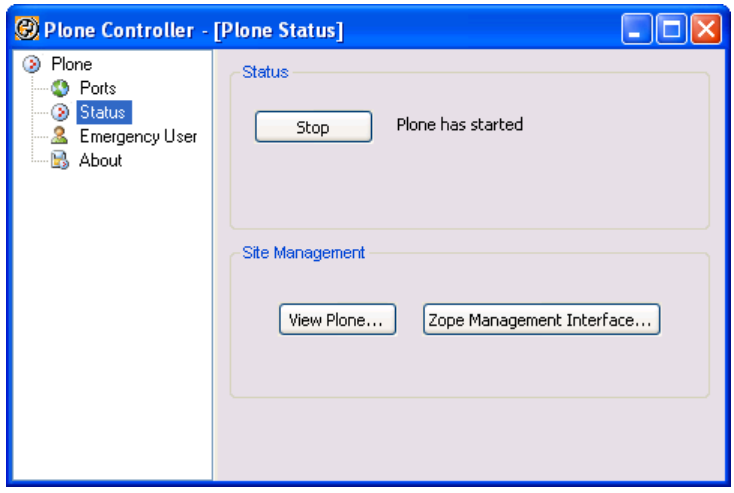


Abbildung 2.6: Nun läuft Plone.

Nachdem Plone gestartet wurde, können Sie auf die Plone-Site zugreifen, indem Sie auf den VIEW-Button klicken. Dabei wird ein Browser gestartet, der auf die Plone-Site zugreift. Dann sollten Sie die Willkommensseite von Plone sehen können. Beachten Sie, dass die Adresse im Browser *http://localhost/* lautet. Mit dieser Adresse greifen Sie auf Ihre Plone-Site zu. Ein Klick auf den ZOPE MANAGEMENT INTERFACE-Button startet einen Browser und greift auf die Management-Schnittstelle zu. Die Adresse im Browser dafür lautet *http://localhost:8080/manage*. Mit ihr haben Sie Zugriff auf den darunter liegenden Application Server. Wenn Sie den MANAGE-Button anklicken und auf Plone zugreifen, werden Sie nach einem

Benutzernamen und Passwort gefragt. Das sind diejenigen, die Sie im Installationsprogramm eingegeben haben.

Der Controller weiß, ob Sie Plone als Dienst installiert haben oder nicht. Wenn Plone als Windows-Dienst installiert wurde, können Sie Plone mit den Standard-Management-Werkzeugen und -Befehlen unter SYSTEMSTEUERUNG – VERWALTUNG – DIENSTE starten und anhalten. Ansonsten können Sie in der Task-Leiste ein kleines Icon sehen. An diesem Punkt werden Sie Inhalte eingeben wollen. Blättern Sie dazu zu Kapitel 3 weiter.

2.1.2 Server-Konfiguration unter Windows

Die Konfigurationsdaten von Plone befinden sich in einer Textdatei, die Sie bearbeiten können, um Ihre Plone-Instanz zu konfigurieren. Sie können die Ports, die Plone abhört, die verwendeten Logdateien und eine Menge anderer Einstellungen ändern. Unter Windows sind einige wesentliche Eigenschaften über den Controller und eine GUI (Graphical User Interface) verfügbar. Wenn Sie andere Einstellungen ändern möchten, lesen Sie bitte Anhang A mit der vollständigen Liste der Konfigurationsoptionen. Um auf den Controller zuzugreifen, wählen Sie START – PROGRAMME – PLONE – PLONE. Dann wird der Controller gestartet.

Wie bereits erwähnt wurde, sehen Sie zuerst die Statusseite, auf der Sie Plone starten oder anhalten können. Im linken Teil des Controllers befinden sich einige weitere Bildschirme, die ich nun beschreiben werde.

Ändern der Ports

Bei der Auswahl der Ports, wie sie in Abbildung 2.7 zu sehen ist, können Sie die Ports angeben, die Plone nach eintreffenden Verbindungen wie HTTP, FTP (File Transfer Protocol) und WebDAV (Web-based Distributed Authoring and Versioning) abhört.

Wie bei der Installation schon erwähnt wurde, möchten Sie wahrscheinlich sicherstellen, dass kein anderer Server wie IIS, Apache und PWS (Personal Web Server) den gleichen Port 80 abhört wie der Plone-Server. Beim Schreiben sind nur die Plone HTTP- und Zope Management HTTP-Ports aktiviert. Um diese zu aktivieren, müssen Sie sie in einer Textdatei konfigurieren. Folgende vier Felder kommen auf der Ports-Seite vor:

- **Plone HTTP**: Dieses Feld gibt den Port an, mit dem der Benutzer auf Plone zugreift. Der Vorgabewert dafür ist 80, der Standardwert für einen Webserver. Obwohl dieser Port nicht benötigt wird, könnten Sie in einem Webbrowser ohne ihn nicht auf Plone zugreifen. Wenn dieser Port aktiviert ist und Plone läuft, ist der VIEW-Button auf der Statusseite aktiviert.

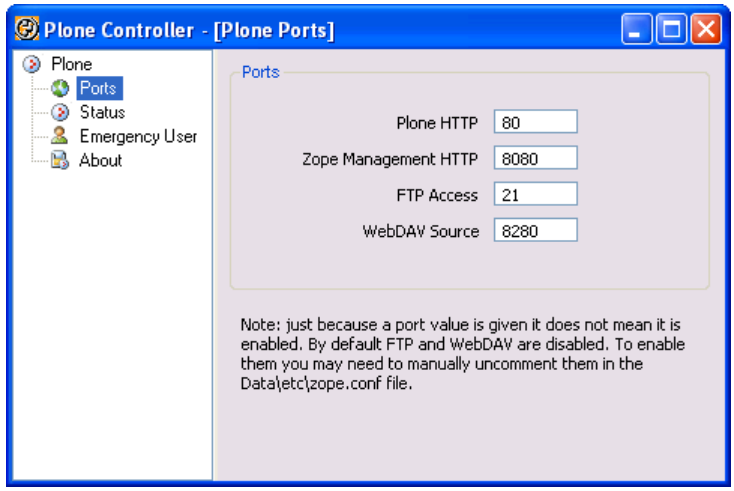


Abbildung 2.7: Die Ports-Seite zeigt die Ports an, auf denen Plone läuft.

- **Zope Management HTTP**: Dieses Feld gibt den Port an, mit dem man als Administrator auf Plone zugreift. Der Vorgabewert dafür ist 8080. Über diesen Port haben Sie Zugriff über das ZMI (Zope Management Interface) auf die Wurzel von Zope. Sie können dort immer noch über den HTTP-Port hinkommen, aber es ist einfacher und bequemer, einen eigenen Port dafür zu haben. Wenn dieser Port aktiviert ist und Plone läuft, ist der MANAGE-Button auf der Statusseite aktiviert.
- **FTP Access**: Dieses Feld gibt den Port an, mit dem man via FTP auf Plone zugreifen kann. Die Vorgabe dafür ist leer, d.h., er ist nicht aktiviert. Wenn Sie ihn aktivieren möchten: Der übliche Wert dafür beträgt 21. Über FTP können Sie große Dateien an und von Plone übertragen.
- **WebDAV Source**: Dieses Feld gibt den Port an, mit dem man über WebDAV auf Plone zugreift. Die Vorgabe dafür ist leer, d.h., er ist nicht aktiviert. Der übliche Wert dafür beträgt 8081. WebDAV ist ein Protokoll für die entfernte Bearbeitung von Plone-Inhalten. Damit können Sie Ihren Plone-Server z.B. auf einen Windows-Laufwerksbuchstaben abbilden.

Verwenden der Emergency User Page

Die Emergency User Page wird in Kapitel 9 behandelt, aber kurz gesagt erhalten Sie damit einen Systemzugriff in Notfällen, wenn Sie Ihren Benutzernamen oder Ihr Passwort vergessen haben.

Plone im Debug-Modus starten

Bisher haben Sie Plone im Produktionsmodus gestartet und angehalten. Dies ist die schnellste und empfohlene Art, Plone auszuführen. Bei der Entwicklung von Add-Ons oder bei der Fehlersuche müssen Sie Plone im Debug-Modus starten. Das ist die empfohlene Methode, Plone auszuführen, wenn Sie Produkte und Aussehen (Skins) entwickeln, wie Sie es in späteren Kapiteln tun werden. Dies ist nicht die Standardmethode, weil Plone dabei etwa zehnmal langsamer als normal ist.

Um Plone im Debug-Modus zu starten, wählen Sie START – PROGRAMME – PLONE – PLONE (DEBUG MODE), wonach eine Eingabeaufforderung erscheint, in deren Fenster alle Log-Informationen ausgegeben werden (siehe Abbildung 2.8).

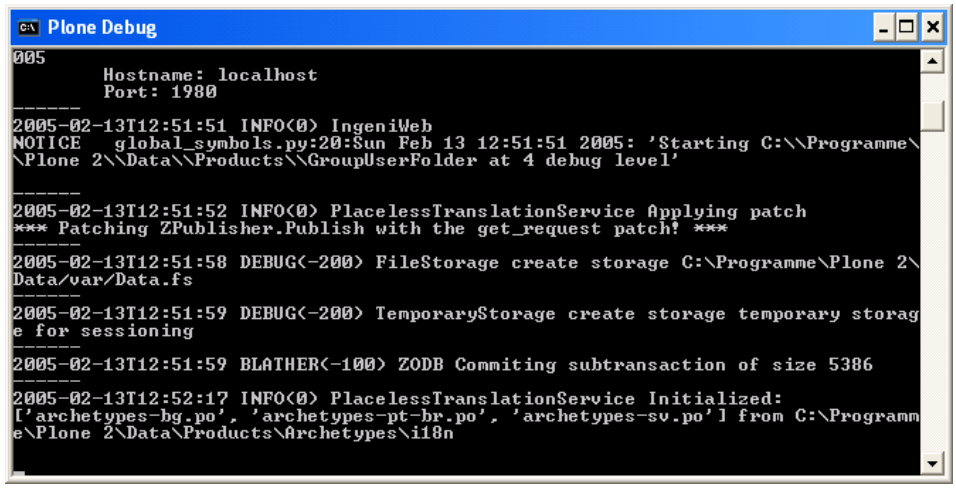


Abbildung 2.8: Plone von der Kommandozeile aus starten

Wenn Sie testen möchten, ob Plone läuft, starten Sie einen Browser und geben *http://localhost/* ein. Wenn Plone erfolgreich installiert ist, können Sie die Willkommensseite von Plone sehen.

2.2 Plone unter Mac OS X, Unix und Linux installieren

Unter Mac OS X, Unix und Linux unterscheidet sich die Installation leicht, aber ihre Konfiguration ist sehr ähnlich. Für die verschiedenen Betriebssysteme sind unterschiedliche Installationspakete vorhanden, z.B. für Mac OS X, Debian, Gentoo, FreeBSD, OpenBSD sowie RPM-Paketmanager (RPMs) für Red Hat, SuSE und Mandrake. In den folgenden Abschnitten behandle ich einige der weiter ver-

breiteten: Mac OS X, Red Hat und Debian. Weitere Informationen zu Ihrem eigenen Betriebssystem finden Sie in den systemspezifischen Installationsanweisungen.

2.2.1 Installation unter Mac OS X

Das Installationsprogramm automatisiert die Installation von Plone unter Mac OS X und wurde unter den Versionen 10.2.3 und höher getestet. Auf dem Zielrechner, wo die Installation erfolgen soll, benötigen Sie Administratorrechte. Sie können das Programm unter *http://www.plone.org/download* herunterladen. Anschließend führen Sie einen Doppelklick darauf aus, um das Archiv auszupacken. Doppelklicken Sie auf das dann entstandene Installationspaket, um die Installation zu beginnen. Danach sollten Sie den Bildschirm aus Abbildung 2.9 sehen.

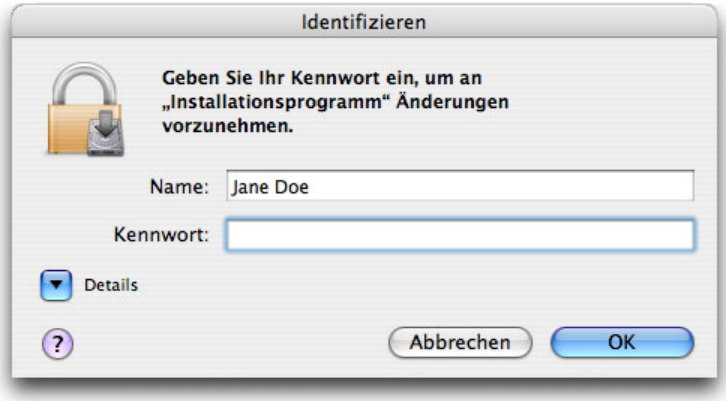


Abbildung 2.9: Genehmigen der Installation mit Ihrem Mac OS X-Passwort

Geben Sie das Passwort Ihres Mac OS X-Kontos ein, um die Installation zu genehmigen. Dazu muss Ihr Konto über Administratorrechte verfügen. Wenn das nicht der Fall sein sollte, melden Sie sich ab und wieder als jemand mit solchen Rechten an, um dann das Installationsprogramm wieder zu starten. Möglicherweise möchten Sie das Installationspaket nach `/Users/Shared` verschieben, bevor Sie sich abmelden, damit Sie von einem anderen Benutzerkonto darauf zugreifen können. Nach erfolgter Genehmigung können Sie den Bildschirm aus Abbildung 2.10 sehen.

Abbildung 2.10: Willkommen im Installationsprogramm

Das Installationsprogramm führt die üblichen Schritte einer Software-Installation durch. Wenn nötig, klicken Sie auf die unteren WEITER- und ZURÜCK-Buttons. Die meisten Schritte sind selbsterklärend, aber bei der Auswahl der Installationspartition für Plone müssen Sie jene Partition wählen, auf der Mac OS X installiert ist (siehe Abbildung 2.11).

Die Installation benötigt ca. fünf Minuten, je nachdem, wie schnell Ihr Rechner ist. Nach der Installation wird Plone nicht automatisch gestartet. Die Datei `ReadMe.rtf` in `/Applications/Plone` enthält eine Menge Informationen darüber, wie Sie Ihre Plone-Installation betreiben und verwalten, und auch darüber, wie Sie Plone starten. Mit dem folgenden Befehl z.B. wird Plone gestartet:

```
sudo /Library/StartupItems/Plone/Plone start
```

Um festzustellen, ob Plone ausgeführt wird, gehen Sie mit einem Webbrowser zu *http://localhost:9090/*, wo Sie eine Willkommensseite von Plone sehen. In der `ReadMe`-Datei finden Sie den Benutzernamen und das Passwort, das Plone für den Zugriff auf den Server durch Sie eingerichtet hat.

Abbildung 2.11: Auswahl der Startpartition

2.2.2 Installation mit einem RPM

RPMs gibt es für die Distributionen von Red Hat, Mandrake und SUSE. Die neuesten Pakete finden Sie unter *http://www.plone.org/download*. RPM setzt bei Python die Version 2.3 voraus. Um festzustellen, welche Python-Version Sie verwenden, führen Sie den folgenden Befehl in einer Shell aus:

```
$ python -V
Python 2.3.2
```

In diesem Fall ist Python 2.3.2 installiert. Wenn Sie diese Version nicht haben, finden Sie RPMs auf der Python-Website unter *http://www.python.org*. Nach dem Herunterladen der Dateien installieren Sie sie mit dem normalen `rpm`-Befehl. Zum Glück werden bei der Installation von Plone einige nützliche Informationen ausgegeben, z.B.:

```
[root@lappi i386]# rpm -ivh Plone2-2.0.0rh-2.i386.rpm
Preparing... ##########################################################
[100%]
Making group plone (not altered if already exists).
Making user plone.
```

```
~ 1:Plone2 ##################################################
[100%]
Creating initial 'main' instance...
Instance created. Listening on 127.0.0.1:8080, initial user: 'plone'
with password: 'plone'.
Setup of initial database in 'main' instance...
/usr/lib/plone2/lib/python/AccessControl/Owned.py:79:
DeprecationWarning: Owned.getOwner(1) is deprecated; please use
getOwnerTuple() instead.
~ DeprecationWarning)
Created initial database content.
look at /etc/plone2/main/zope.conf.
Run then "/etc/rc.d/init.d/plone2 start" to start Plone2.
you may create new Plone instances with mkploneinstance.
```

Wie in der vorherigen Ausgabe zu sehen ist, können Sie Plone wie folgt starten:

```
/etc/rc.d/init.d/plone2 start
```

Um festzustellen, ob Plone funktioniert, gehen Sie mit einem Webbrowser zu *http://localhost:9090/*, wo Sie eine Willkommensseite von Plone sehen. Der Benutzername *plone* und das Passwort *plone* wurden für Sie eingerichtet.

2.2.3 Installation unter Debian Linux

Unter Debian ist Plone ein Standardpaket und geht durch den standardisierten Release-Prozess, d.h., Sie können sich entweder eine stabile oder eine instabile Version von Plone holen, je nachdem, wie Ihre Debian-Installation konfiguriert ist. Plone installieren Sie einfach mit Debians *apt*-System. Hier ist ein Beispiel einer Installation:

```
agmweb:/home/andy# apt-get install plone
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  zope zope-cmf zope-cmfcalendar zope-cmfcore zope-cmfdefault
zope-cmfplone zope-cmftopic zope-cmfworkflow
  zope-formulator zopectl
Suggested packages:
  zope-cmfwiki python-unit zope-devguide zope-book
Recommended packages:
  zope-cmfforum zope-localizer
The following NEW packages will be installed:
  plone zope zope-cmf zope-cmfcalendar zope-cmfcore zope-cmfdefault
zope-cmfplone zope-cmftopic zope-cmfworkflow
  zope-formulator zopectl
```

```
0 upgraded, 11 newly installed, 0 to remove and 49 not upgraded.
Need to get 4743kB of archives.
After unpacking 24.9MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Geben Sie [Y] ein, um fortzufahren und alle benötigten Pakete zu installieren. Um Zope zu starten und anzuhalten, wurde ein Installationsscript namens `zope` im Verzeichnis `init.d` erstellt. Geben Sie Folgendes ein, um Plone zu starten:

```
/etc/init.d/zope start
```

Das Debian-Installationsprogramm startet Zope auf dem ungewöhnlichen Port 9673. Weil das so ist, wird empfohlen, die Dokumentation zu lesen, die unter `/usr/share/doc/zope` und `/usr/share/doc/zope-cmfplone` zu finden ist.

2.3 Installation aus den Quellen

Alternativ zur Benutzung eines Installationsprogramms oder -pakets können Sie die Installation auch aus einem Quellcode-Archiv vornehmen. Wenn Sie mit einer Quellcode-Installation nicht vertraut sind: Sie ist nicht schwierig, aber man muss sich ein wenig mit einfachen Werkzeugen wie `tar` auskennen. Die folgenden Abschnitte beschreiben, wie man den Quellcode unter Linux installiert.

Diese Art der Installation setzt voraus, dass Sie mit grundlegenden Operationen wie dem Auspacken und Verschieben von Dateien vertraut sind. Es wird auch eine funktionierende Zope-Installation benötigt.

Hinweis

Um Zope zu installieren, lesen Sie die Installationsanweisungen zu Zope in der Datei `doc/INSTALL.txt` Ihres heruntergeladenen Zope-Archivs. Weitere Informationen finden Sie unter *http://zope.org/Documentation/Books/ZopeBook/2_6Edition/Installing-Zope.stx*).

Führen Sie folgende Schritte aus, um Plone zu installieren:

1. Laden Sie Plone 2 von *http://www.plone.org/download* herunter, und wählen Sie die Quellcode-Datei.
2. Packen Sie das Archiv mit `tar xzf CMFPlone2.0.tar.gz` aus.
3. Sie werden feststellen, dass ein Verzeichnis namens `CMFPlone-xxx` erstellt wurde, wobei `xxx` die Version ist (z.B. `CMFPlone-2.0`).

4. Bewegen Sie den Inhalt dieses Verzeichnisses in das Produktverzeichnis Ihrer Zope-Installation. Wenn das Zope-Produktverzeichnis z.B. `/var/zope` ist, dann führen Sie Folgendes aus: `mv CMFPlone2.0/ /var/zope/Products`

Nach dem Ende der Installation starten Sie Zope neu. Anschließend greifen Sie auf Zope zu, indem Sie in einem Browser *http://localhost:8080/manage* eingeben. Dafür benötigen Sie einen Benutzernamen und ein Passwort (z.B. jene, die Sie bei der Zope-Installation eingegeben haben).

Im ZMI gibt es in der oberen rechten Ecke eine Dropdown-Liste für Produkte, die Sie hinzufügen können. Überprüfen Sie, dass eines davon Plone Site ist. Wenn das der Fall ist, dann ist Ihre Installation vollständig (siehe Abbildung 2.12).

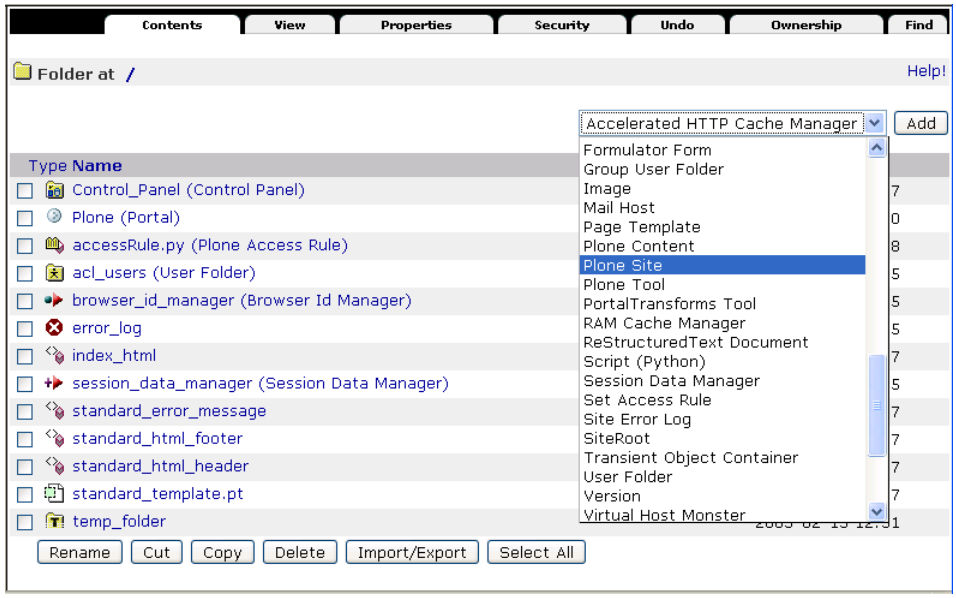


Abbildung 2.12: Der Plone-Site-Eintrag in der Dropdown-Liste Installation aus CVS

2.3.1 Installation aus CVS

Der Zugriff per CVS (Concurrent Versioning System) wird nur erfahrenen Benutzern und Entwicklern empfohlen. Informationen zum Zugriff per CVS finden Sie unter *http://ww.plone.org/development/cvs*. Der aktuelle CVS-Checkout-Befehl lautet wie folgt:

```
cvs -d:pserver:anonymous@cvs.sf.net:/cvsroot/plone login
cvs -d:pserver:anonymous@cvs.sf.net:/cvsroot/plone co CMFPlone
```

Plone 2 bringt eine ganze Reihe von Abhängigkeiten mit sich (z.B. DCWorkflow, Formulator, Group User Folder usw.), die nicht im Plone-CVS enthalten sind. Das heißt, die Benutzer müssen diese Abhängigkeiten selbst auflösen. Wenn Sie Plone starten, gibt es eventuelle Fehler aus, die sich auf nicht gefundene Pakete beziehen, z.B.:

```
2003-11-21T12:23:11 ERROR(200) Plone Dependency
CMFActionIcons not found.  Please download it from http://cvs.zope.org/Products/
```

2.3.2 Hinzufügen einer Plone-Site

Nachdem Sie Plone aus seinen Quellen installiert haben, müssen Sie eine Plone-Instanz erstellen. Dazu müssen Sie sich beim ZMI anmelden und dort die Plone-Site hinzufügen. Dorthin gelangen Sie, wenn Sie die URL (Uniform Resource Locator) für die Management-Schnittstelle eingeben, normalerweise *http://localhost:8080/manage* (dieser Port variiert je nach Installation). Für den Zugriff auf den ZMI benötigen Sie einen Benutzernamen und ein Passwort eines Managers, die bei der Zope-Installation angelegt werden.

Hinweis

Falls Sie das Plone-Passwort vergessen haben, das bei der Installation erstellt wurde, keine Panik! Sie können ein neues erstellen; lesen Sie dazu Kapitel 9.

Alle Objekte fügen Sie mit der Dropdown-Liste in der oberen rechten Ecke hinzu, die in Abbildung 2.12 zu sehen ist. Scrollen Sie die Liste runter, bis Sie Plone Site finden, und klicken Sie auf HINZUFÜGEN.

Nach Auswahl des Eintrags Plone Site erscheint ein Formular, in dem weitere Angaben verlangt werden (siehe Abbildung 2.13):

- **Id**: Dies ist die eindeutige ID der Plone-Site (geben Sie z.B. **Plone** oder **Site** ein).
- **Title**: Dies ist der Titel der Plone-Site (geben Sie z.B. **Plone-Demo-Site** ein).
- **Membership source**: Belassen Sie dies vorläufig beim Vorgabewert, CREATE A NEW USER FOLDER IN THE PORTAL. Damit können Sie eine Benutzeridentifikation außerhalb des Portals verwenden (siehe Kapitel 9).
- **Description**: Dies ist eine Beschreibung des Portals, die Mitglieder in E-Mails sehen können (geben Sie z.B. **Das ist eine Demo-Site für Plone.** ein). Machen Sie sich jetzt nicht viele Gedanken darüber. Sie können sie später in den Portal-Eigenschaften ändern.

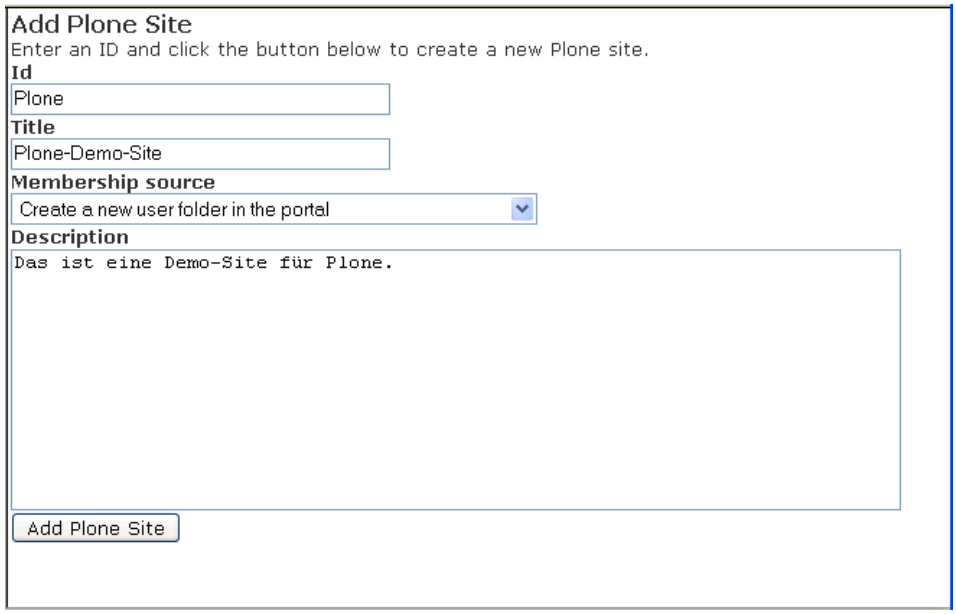


Abbildung 2.13: Hinzufügen einer Plone-Site

Nach einem Klick auf ADD PLONE SITE wird eine Plone-Site erstellt. Auf langsamen Rechnern kann das ein bis zwei Minuten in Anspruch nehmen, da viele Daten dabei verarbeitet werden. Dieser Bildschirm führt Sie dann zur Willkommensseite von Plone weiter.

2.4 Konfigurieren des Webservers

Nach der Installation von Plone sollten Sie die Plone-Site so konfigurieren, dass sie unter einem anderen Port läuft, über FTP verfügt, Logdaten in eine andere Datei schreibt usw. Von diesen grundlegenden Einrichtungsmöglichkeiten handelt dieser Abschnitt. Beachten Sie, dass Sie nicht die Plone-Sites selbst konfigurieren, sondern den darunter liegenden Webserver.

Hinweis

Falls Sie das Windows-Installationsprogramm benutzt haben, erfolgt der Großteil dieser Konfiguration mit einem Programm, das eine nette Benutzerschnittstelle hat; siehe den Abschnitt »Server-Konfiguration unter Windows« weiter oben in diesem Kapitel.

Hinweis

Falls Sie das Installationsprogramm unter Mac OS X oder Windows verwendet haben, werden Sie eine weitere Datei finden (`plone.conf`), die Port-Definitionen enthält, die in Zopes Hauptkonfigurationsdatei verwendet werden.

Zope 2.7 erstellt eine Konfigurationsdatei in jeder einzelnen installierten Instanz. Diese Datei enthält die gesamte Konfiguration für diesen Server. Anhang A enthält eine vollständige Liste der Konfigurationsmöglichkeiten. Sie finden die Datei, wenn Sie nach `zope.conf` im Verzeichnis `etc` Ihrer Plone-Installation suchen. Manche Installationsprogramme (z.B. bei Mac OS X und Windows) erstellen eine zweite Konfigurationsdatei namens `plone.conf`, die Plone-spezifische Konfigurationen enthält. Falls Ihre Installation eine Datei `plone.conf` enthält, dann nehmen Sie dort Änderungen an der Konfiguration vor. Sie werden dann in der Hauptkonfigurationsdatei importiert.

Die Konfigurationsdatei ist extrem ausführlich und enthält eine große Menge hilfreicher Kommentare und Beispiele. Wenn Sie mit Unix-Konfigurationsdateien z.B. für Apache vertraut sind, wird Ihnen die Zope-Konfigurationsdatei bekannt vorkommen. Um die Konfiguration von Zope zu ändern öffnen Sie einen Texteditor und ändern die Dateien nach Bedarf. Danach müssen Sie Zope neu starten.

Man kann Plone 2.0 mit einer Zope-Version kleiner als 2.7 betreiben, allerdings bietet Zope 2.7 eine erhöhte Stabilität und neue Eigenschaften wie eine einfachere Konfiguration. Wenn Sie eine frühere Zope-Version als 2.7 verwenden, müssen Sie in der Dokumentation nachlesen, wie Sie die Konfiguration ändern.

2.4.1 Ändern der Ports

Einen Port ändern Sie dadurch, dass Sie eine Adresszeile dafür hinzufügen. Um Plone z.B. auf Port 80 statt auf dem Vorgabeport zu betreiben, ändern Sie die folgende fette Zeile in `zope.conf`:

```
<http-server>
  # valid keys are "address" and "force-connection-close"
  address 8080
  # force-connection-close on
</http-server>
```

auf Folgendes:

```
<http-server>
  # valid keys are "address" and "force-connection-close"
  address 80
  # force-connection-close on
</http-server>
```

Wenn Sie das Windows- oder Mac OS X-Installationsprogramm verwendet haben, dann finden Sie diese Port-Definitionen in `plone.conf`. Diese Werte werden dann in die Hauptkonfigurationsdatei importiert. Um also den Port auf einem Mac zu ändern, ändern Sie in `plone.conf` diesen Teil:

```
## PLONE_WEBSERVER_PORT
## --------------------
## This is the port you will access your Plone site from.  Set this to a port
## number above 1024 not used for any other server on your computer.
%define PLONE_WEBSERVER_PORT 8080
```

wie folgt:

```
%define PLONE_WEBSERVER_PORT 80
```

2.4.2 Verwenden des Debug-Modus

In Zope 2.7 ist der Debug-Modus standardmäßig eingeschaltet. Beachten Sie aber, dass Plone im Debug-Modus wesentlich langsamer läuft, und zwar etwa 10- bis 20-mal langsamer. Fügen Sie folgende Zeile in der Konfigurationsdatei hinzu, um den Debug-Modus abzuschalten:

```
debug-mode off
```

Um das erstmalige Benutzererlebnis für Windows-Anwender eindrucksvoller zu machen (der Debug-Modus verlangsamt Plone unter Windows noch mehr als unter Linux), wird hier Zope bereits mit abgeschaltetem Debug-Modus ausgeliefert. Wenn Sie für eine laufende Plone-Site wissen möchten, ob der Debug-Modus eingeschaltet ist, gehen Sie im ZMI zu PORTAL_MIGRATION und sehen sich die dort aufgelisteten Variablen an.

2.4.3 Verwenden von Logs

In Plone gibt es standardmäßig zwei Logmechanismen: einen für Zugriffe, mit dem man Site-Statistiken produzieren kann, und einen für Ereignisse, der Debug-Informationen über Plone-Produkte enthält. Für Plone-Fehler und -Meldungen ist der Ereignis-Log zuständig. Die Standardkonfiguration sieht wie folgt aus:

```
<eventlog>
  level all
  <logfile>
    path $INSTANCE/log/event.log
    level INFO
  </logfile>
</eventlog>

<logger access>
  level WARN
  <logfile>
    path $INSTANCE/log/Z2.log
    format %(message)s
  </logfile>
</logger>
```

Hier können Sie den Pfad zur Datei ändern, indem Sie eine neue Datei angeben. Die geloggten Werte basieren auf einem Level, der mit den Fehlermeldungen mitgeschickt wird. Ernstere Fehler werden mit höherem Level geschickt. Standardmäßig werden nur Informationen und die vorherige Meldung ans Log geschickt, aber dieser Wert könnte einer der folgenden sein: `CRITICAL`, `ERROR`, `WARN`, `INFO`, `DEBUG` und `ALL`. Wenn Sie nur Fehler loggen möchten, würden Sie `level INFO` auf `level ERROR` ändern.

3 Inhalte hinzufügen und bearbeiten

»Inhalte hinzufügen und bearbeiten« bedeutet eine arge Vereinfachung der vielfältigen Möglichkeiten, die Plone zur Verfügung stellt. Mit Plone lassen sich sehr leicht Webseiten erstellen, die verschiedenste Inhalte und Möglichkeiten bieten. Falls Sie Plone lokal installiert haben, erfahren Sie in diesem Kapitel, wie Plone unmittelbar funktioniert. Wenn Sie es nicht installiert haben, müssen Sie sich aber nicht etwa beeilen, denn dann können Sie Plone unter *http://demo.plone.org* online ausprobieren.

Bevor Sie eine Plone-Site ändern oder bearbeiten können, müssen Sie sich auf der Plone-Site erst einmal anmelden. Sofern Sie Plone installiert haben, sollten Sie über den Benutzernamen und das Passwort verfügen, die bei der Installation eingerichtet wurden. Dieser Benutzer hat die Rolle eines Administrators, mit Hilfe derer Sie sich anmelden und beliebige Inhalte ändern können. Die meisten Benutzer einer Plone-Site registrieren sich auf der Site und melden sich über den Mechanismus an, der im Abschnitt »Registrierung auf einer Site« beschrieben wird. Natürlich können Benutzer eine Plone-Site auch anschauen, ohne sich zu registrieren, aber dann können sie keine Inhalte hinzufügen oder ändern.

In diesem Kapitel beschreibe ich schrittweise, was ein Benutzer tun muss, um auf einer Plone-Site Inhalte zu erstellen. Zuerst behandle ich, wie man Mitglied wird und sich anmeldet. Danach beschreibe ich, wie man ein Dokument erstellt und es bearbeitet. Und schließlich zeige ich Ihnen, wie Sie nach diesem Inhalt suchen und ihn veröffentlichen können. Kurz gesagt: Dieses Kapitel beschreibt, wie man Plone benutzt.

3.1 Registrierung auf einer Site

Wenn Sie sich auf einer Plone-Site registrieren, erstellen Sie auf dem Server ein Benutzerkonto. Mit diesem Konto erhalten Sie als Mitglied das Recht, Inhalte wie Bilder, Dokumente usw. hinzuzufügen. Um sich auf einer Site zu registrieren, kli-

cken Sie oben rechts auf der Webseite auf den Link MITGLIED WERDEN (siehe Abbildung 3.1).

Abbildung 3.1: Klick auf Mitglied werden in der oberen rechten Ecke der Seite

Damit gelangen Sie zu einem Registrierungsformular, das Sie ausfüllen müssen (siehe Abbildung 3.2). Da dies das erste Plone-Formular ist, dem Sie begegnen, sollten Sie auf Folgendes achten:

- Manche Felder müssen ausgefüllt werden. Auf solche Felder weist ein kleiner roter Punkt neben dem Text hin.
- Zu den meisten Feldern gibt eine graue Texthilfestellung unter dem Feldnamen an, was Sie darin eingeben sollen.

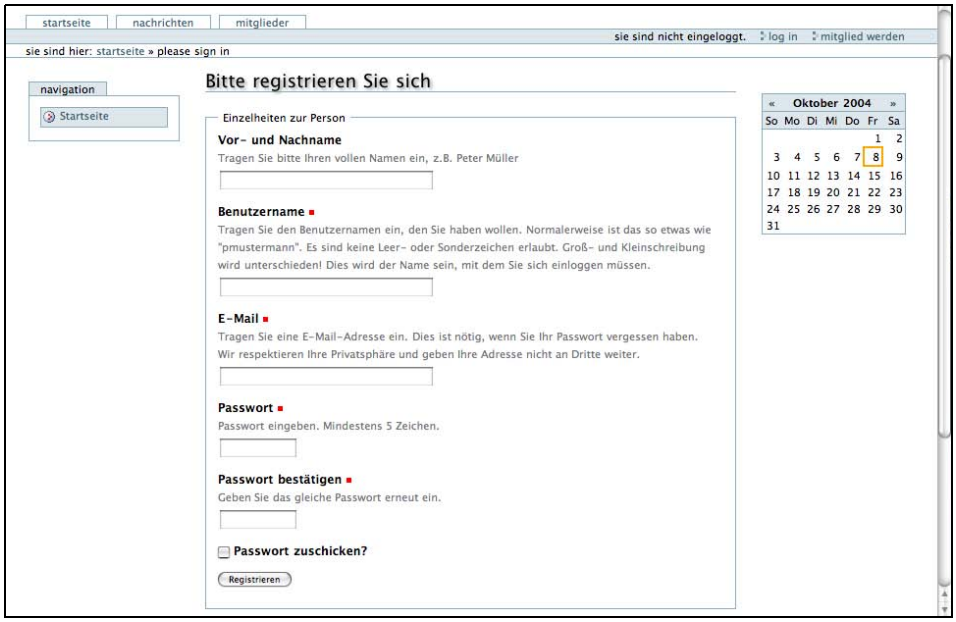


Abbildung 3.2: Die Registrierungsseite

Hinweis

Da die meisten Plone-Seiten ziemlich groß sind, wurden die Abbildungen in diesem Buch so beschnitten, dass nur die wesentlichen Teile (in diesem Fall das Formular) dargestellt werden und nicht das Plone-Logo oder die Fußzeilen. Diese Elemente sind ebenfalls vorhanden, aber hierbei unwichtig.

Füllen Sie das Formular aus, indem Sie folgende Angaben in den jeweiligen Feldern machen:

- **Vor- und Nachname**: Geben Sie Ihren vollständigen Namen ein. Dieses Feld ist optional.
- **Benutzername**: Geben Sie den gewünschten Benutzernamen ein. Die meisten Leute wählen einen alphanumerischen Wert ohne Leerzeichen, z.B. `bob` oder `jane97`. Dieser Benutzername wird auf der gesamten Website als Kürzel für Sie verwendet. Dieses Feld muss ausgefüllt werden.
- **E-Mail**: Eine gültige E-Mail-Adresse ist ebenfalls notwendig. Damit kann der Site-Administrator Sie kontaktieren und Ihnen ein Passwort schicken. Diese E-Mail-Adresse können Sie später in Ihren Mitgliedervoreinstellungen ändern. Dieses Feld muss ausgefüllt werden.
- **Passwort und Passwort bestätigen**: Dies ist das gewünschte Passwort. Es muss mehr als vier Zeichen haben und darf aus Buchstaben, Zahlen und Unterstrichen (_) bestehen. Die Schreibweise in Passwörtern ist wichtig (d.h. `EinPasswort` ist etwas anderes als `einpasswort`). Diese Felder müssen ausgefüllt werden.
- **Passwort zuschicken?**: Kreuzen Sie dieses Feld an, wenn Ihr Passwort an Ihre angegebene E-Mail-Adresse geschickt werden soll. Dieses Feld ist optional.

Wenn Sie das Formular vollständig ausgefüllt haben, klicken Sie auf REGISTRIEREN, um Ihre Angaben abzuschicken. Wenn Sie Fehler beim Ausfüllen des Formulars gemacht haben, sehen Sie oben einen Hinweis und eine Hervorhebung in den betroffenen Feldern. In Abbildung 3.3 habe ich ein Passwort eingegeben, aber das Feld PASSWORT BESTÄTIGEN nicht ausgefüllt. Dies ist die normale Art, wie Ihnen Fehler in Plone-Formularen angezeigt werden.

Falls Sie das Formular korrekt ausgefüllt haben, erhalten Sie die Möglichkeit, sich sofort anzumelden. Dazu klicken Sie auf den Button LOG IN. Dann sehen Sie die Seite in Abbildung 3.4.

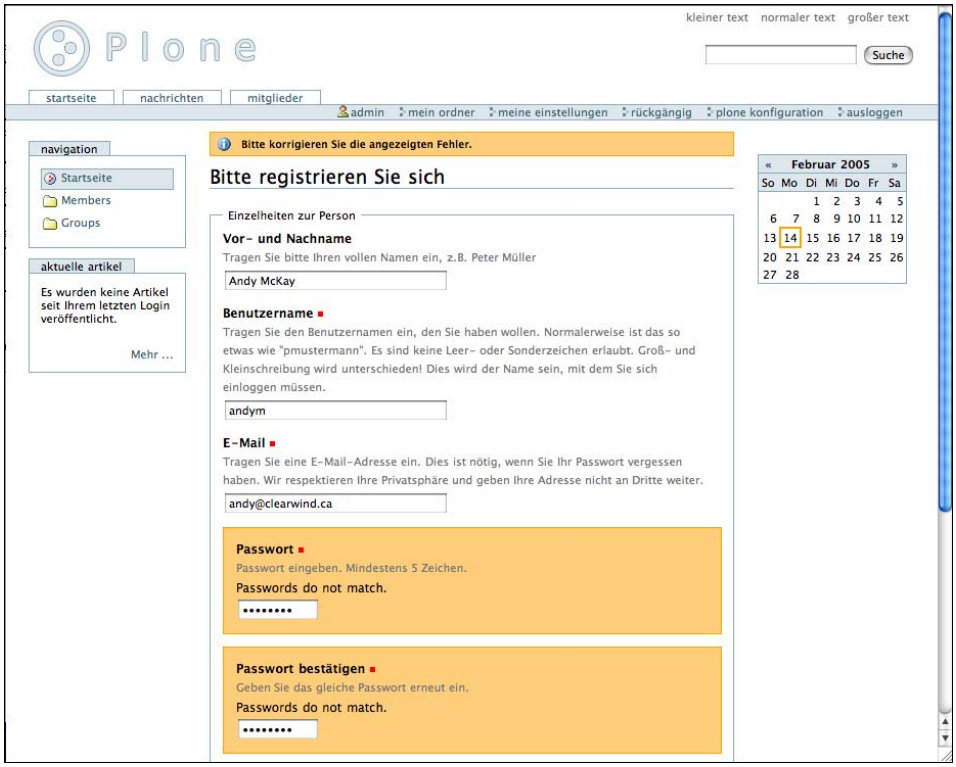


Abbildung 3.3: Fehler in einem Formular

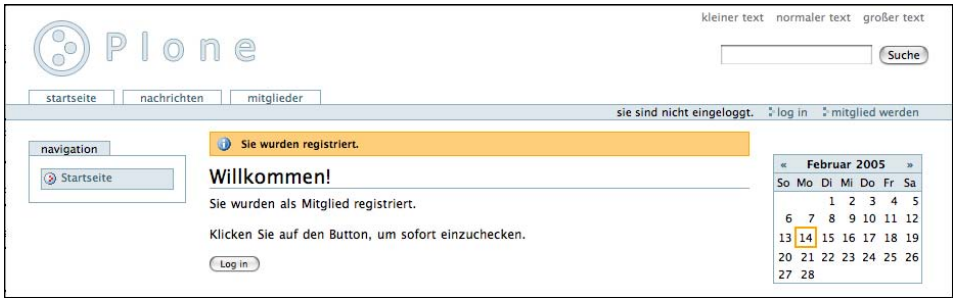


Abbildung 3.4: Nach der Registrierung

Wenn Sie bereits einen Benutzernamen und ein Passwort haben oder zu einer Site kommen, auf der Sie schon registriert sind, können Sie Benutzernamen und Passwort in der linken Spalte der Seite eingeben und auf den Button LOG IN klicken.

Cookies aktivieren

Um sich auf einer Plone-Site anzumelden, müssen Sie Cookies aktiviert haben. Wenn Sie versuchen, ohne Cookie-Unterstützung auf eine Plone-Site zuzugreifen, erhalten Sie einen freundlichen Hinweis darauf, dass Sie Cookies einschalten müssen, sowie einen Link auf weitere Informationen. Je nach verwendetem Browser können Sie Cookies wie folgt einschalten:

Internet Explorer 6.x

1. Wählen Sie EXTRAS – INTERNETOPTIONEN.
2. Klicken Sie oben auf dem Bildschirm auf den PRIVACY-Reiter.
3. Bewegen Sie den Rollbalken auf MEDIUM, und klicken Sie auf OK.

Internet Explorer 5.x

1. Wählen Sie EXTRAS – INTERNETOPTIONEN.
2. Klicken Sie oben auf dem Bildschirm auf den SECURITY-Reiter.
3. Klicken Sie auf CUSTOM LEVEL, und scrollen Sie bis zum COOKIES-Abschnitt hinunter.
4. Setzen Sie ALLOW PER-SESSION COOKIES auf ENABLE, und klicken Sie auf OK.

Internet Explorer 4.x

1. Wählen Sie ANSICHT – INTERNETOPTIONEN.
2. Klicken Sie oben auf dem Bildschirm auf den SECURITY-Reiter.
3. Klicken Sie auf CUSTOM LEVEL, und scrollen Sie bis zum COOKIES-Abschnitt hinunter.
4. Wählen Sie ALWAYS ACCEPT COOKIES oder PROMPT BEFORE ACCEPTING COOKIES, und klicken Sie auf OK.

Mozilla 1.x

1. Wählen Sie BEARBEITEN – EINSTELLUNGEN.
2. Finden Sie im linken Menü DATENSCHUTZ & SICHERHEIT. Wenn ein Pluszeichen (+) links von DATENSCHUTZ & SICHERHEIT steht, klicken Sie darauf.
3. Wählen Sie COOKIES.
4. Wählen Sie COCKIES NUR VON DER ORIGINAL-WEBSEITE AKZEPTIEREN oder ALLE COCKIES AKZEPTIEREN, und klicken Sie auf OK.

Opera

1. Drücken Sie F12.
2. Klicken Sie auf COCKIES AKZEPTIEREN.

Netscape Navigator 6.x:

1. Wählen Sie BEARBEITEN – EINSTELLUNGEN.
2. Finden Sie im linken Menü PRIVACY & SECURITY. Wenn ein nach rechts zeigendes Dreieck neben PRIVACY & SECURITY steht, klicken Sie darauf.
3. Wählen Sie COOKIES unter DATENSCHUTZ & SICHERHEIT.
4. Wählen Sie COCKIES NUR VON DER ORIGINAL-WEBSEITE AKZEPTIEREN oder ALLE COCKIES AKZEPTIEREN, und klicken Sie auf OK.

Falls Sie irgendwann einmal Ihr Passwort vergessen, können Sie es sich an die E-Mail-Adresse schicken lassen, die Sie bei Ihrer Registrierung an der Plone-Site angegeben haben. Damit Ihnen das Passwort per E-Mail geschickt wird, klicken Sie auf den Link PASSWORT VERGESSEN? in der linken Spalte der Seite. Dann wird das Formular PASSWORT VERGESSEN? angezeigt, das in Abbildung 3.5 zu sehen ist. Geben Sie Ihren Benutzernamen ein, und Sie erhalten ein Passwort per E-Mail.

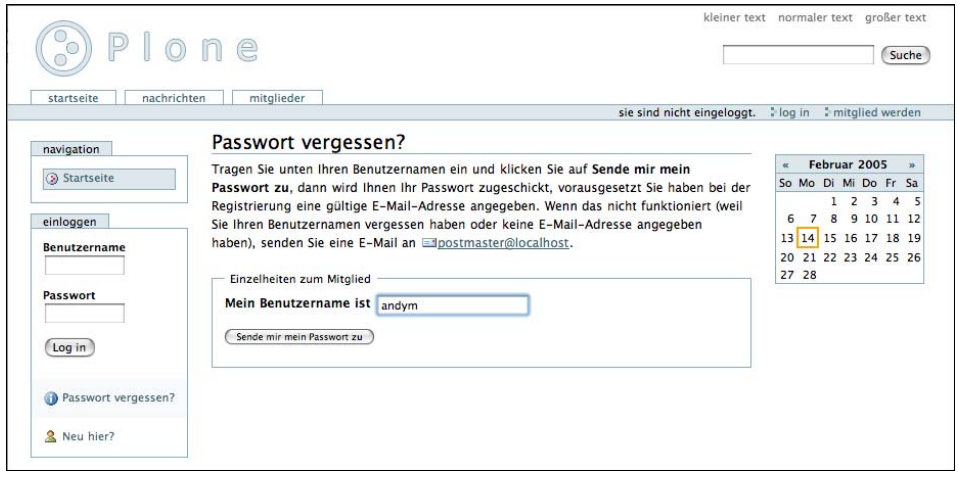


Abbildung 3.5: Anfordern eines vergessenen Passworts

Wenn Sie allerdings keinen Zugriff mehr auf diese E-Mail-Adresse haben oder gar Ihren Benutzernamen vergessen haben, müssen Sie leider einen Site-Administrator kontaktieren. Mit den in Kapitel 9 beschriebenen Techniken kann der Administrator Ihre E-Mail-Adresse ändern und Ihr Benutzerkonto ausfindig

machen. Wenn Sie einmal bei einer Plone-Site angemeldet sind, sehen Sie in der oberen rechten Ecke einen Link namens AUSLOGGEN. Wenn Sie Ihre Arbeit beendet haben, ist es ratsam, sich von einer Plone-Site abzumelden, besonders dann, wenn Sie von einem Rechner darauf zugreifen, der sehr wahrscheinlich von anderen Leuten benutzt wird.

3.2 Ihren Ordner und Ihre Voreinstellungen einrichten

Nachdem Sie sich angemeldet haben, ändert sich die Mitgliederleiste oben rechts, um die für Sie als Site-Mitglied verfügbaren Optionen darzustellen (siehe Abbildung 3.6).

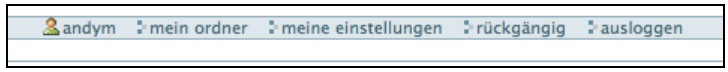


Abbildung 3.6: Ihre persönlichen Optionen in der oberen rechten Ecke haben sich verändert.

Eine dieser Optionen besteht in der Einrichtung eines Ordners für jedes Mitglied, das sich bei einer Site registriert. Für diesen Ordner ist eine spezielle Sicherheitsstufe eingestellt, damit nur dieses Mitglied (und Administratoren) Inhalte darin hinzufügen und bearbeiten können. Um auf Ihren persönlichen Ordner zuzugreifen, klicken Sie auf den Link MEIN ORDNER in der persönlichen Leiste oben rechts auf der Seite. Dort sehen Sie auch einen Link namens MEINE EINSTELLUNGEN. Ein Klick darauf öffnet eine Liste persönlicher Einstellungen. Im Moment gibt es dort zwei Einträge. Sie können Ihr Passwort ändern, oder Sie können zu den persönlichen Voreinstellungen gehen und wesentliche Einstellungen an Ihrer Site ändern.

Mit dem Formular PASSWORT VERÄNDERN können Sie, nun ja, Ihr Passwort verändern. Um das Formular auszufüllen, geben Sie zunächst das aktuelle Passwort und dann zweimal das neue ein. Nach dieser Änderung wird das neue Passwort sofort gültig. Sie müssen sich nicht erneut anmelden, aber Sie sollten sich das neue Passwort merken, wenn Sie wiederkommen.

In dem Formular EINZELHEITEN ZUM MITGLIED können Sie eine Reihe von Voreinstellungen setzen, die die Art und Weise verändern, wie Sie die Site sehen. Diese Voreinstellungen werden auf dem Server gespeichert, d.h., sie bleiben über die einzelnen Sitzungen hinaus erhalten (siehe Abbildung 3.7).

Folgende Optionen stehen zur Verfügung:

- **Vor- und Nachname**: Dies ist der vollständige Name, den Sie bei Ihrer Registrierung auf der Site angegeben haben.

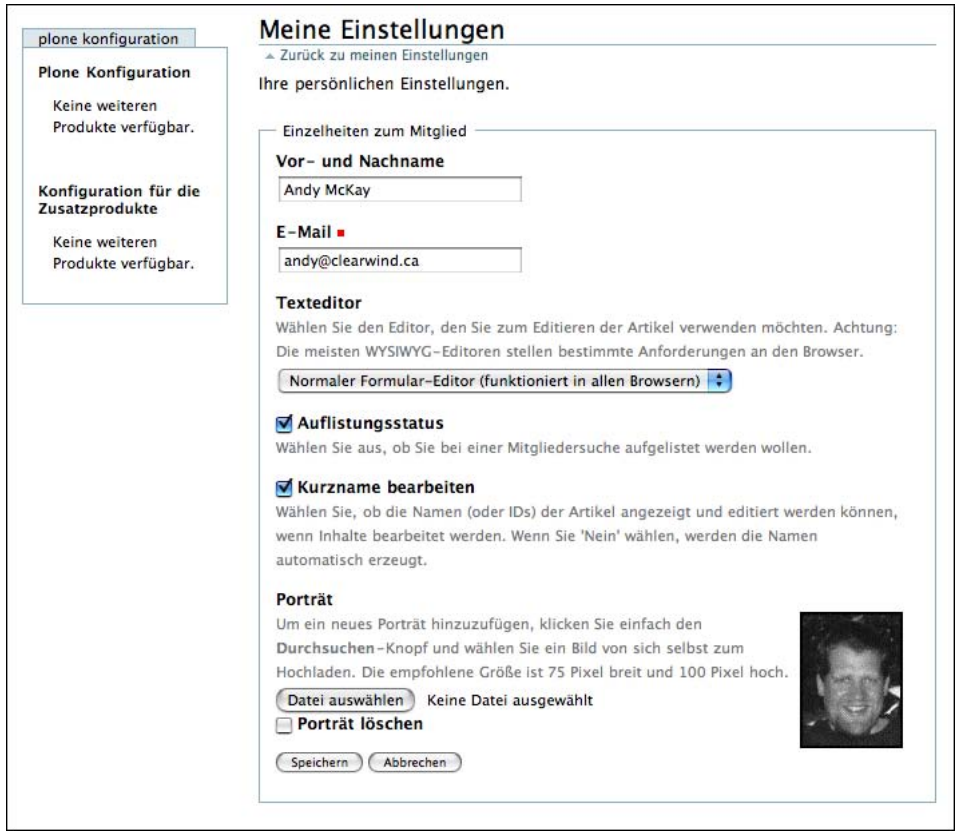


Abbildung 3.7: Ändern der Voreinstellungen

- **E-Mail**: Dies ist die mit Ihrer Mitgliedschaft verbundene E-Mail-Adresse, die an einer Reihe von Stellen in einer Plone-Site verwendet wird. Insbesondere ist es die Adresse, an die das System ein verlorenes oder vergessenes Passwort schickt.
- **Texteditor**: Beim Bearbeiten komplexer Inhalte benötigen Sie evtl. die Hilfe eines Editors. Falls Ihr Site-Administrator einen Editor verfügbar gemacht hat, können Sie ihn hier auswählen. Er wird dann benutzt, wenn Sie auf den Reiter BEARBEITEN EINES OBJEKTS klicken. Wenn Sie unsicher sind, lassen Sie die Voreinstellung unverändert.
- **Auflistungsstatus**: Diese Eigenschaft gibt an, ob Ihr Profil unter dem Reiter MITGLIEDER und dann erscheint, wenn jemand die Mitgliederliste durchsucht.
- **Zeige Namen an**: Objekte haben eine ID bzw. eine Kurzname-Eigenschaft, der für die interne Darstellung des Inhaltsobjekts verwendet wird. Sie erscheint auch in der Webadresse dieses Elements und in dessen URL (Uniform Resource

Locator). Standardmäßig sehen sie ungefähr aus wie z.B. `News_Item.2002-11-16.4102`, aber Sie könnten sie auch viel einfacher machen, z.B. `november_news`, indem Sie einfach den Wert des Kurznamens ändern.

Hinweis

Wenn Sie den Namen eines Objekts ändern, wird alles ungültig, was auf den alten Namen verweist. Das führt dazu, dass die Seite nicht gefunden wird. Am besten ändern Sie den Namen nicht mehr, nachdem Sie ein Objekt zur Überprüfung einreichen bzw. von anderswo darauf verweisen. Aus diesem Grund empfehle ich, diese Option auf NEIN zu setzen.

Sollten Sie diese Option dennoch aktivieren, so empfiehlt sich, keine Umlaute, Sonderzeichen oder etwa Leerzeichen für den Namen zu verwenden, da dieser, wie bereits erwähnt, in der URL des Objekts vorkommt. Auch wenn Ihr Browser damit umgehen kann, sollten Sie daran denken, dass andere Browser dies evtl. nicht unterstützen. Im Titel eines Objekts sind Ihrer Kreativität dagegen keine Grenzen gesetzt.

- **Porträt**: In großen Organisationen ebenso wie auf Community-Websites ist es hilfreich, Bilder von anderen Mitgliedern sehen zu können. Mit dem PORTRÄT-Feld können Sie ein Bild von sich selbst hochladen. Das Bild sollte 75 mal 100 Pixel groß sein.

Nachdem Sie die gewünschten Änderungen vorgenommen haben, klicken Sie auf den Button SPEICHERN. Nun, da Sie angemeldet sind, ist es Zeit, Inhalte hinzuzufügen und zu bearbeiten.

3.3 Dokumente hinzufügen und bearbeiten

Wie schon gesagt, verfügen Sie nun, da Sie Mitglied auf der Site sind, über einen Ordner, in dem Sie Inhalte speichern können. Natürlich können Sie Inhalte in allen Ordnern erstellen, in denen Sie vom Administrator die entsprechenden Rechte erhalten haben, aber standardmäßig kann jeder Benutzer etwas in seinem eigenen Ordner speichern.

Sie können unterschiedliche Arten von Inhalt erstellen und diesen auf verschiedene Weise bearbeiten und anschauen. Aus diesem Grund werden diese verschiedenen Inhaltstypen in Plone unterschiedlich bezeichnet, z.B. Bilder, Links, Dokumente usw. Von sich aus bietet Plone folgende Inhaltstypen:

- **Dokument**: Dieses Element stellt irgendeine statische Information für den Benutzer dar. Dies ist der am häufigsten erstellte Inhaltstyp und entspricht noch am ehesten einer typischen Webseite.
- **Nachricht**: Dies ist ein Dokument, das unter dem NACHRICHTEN-Reiter erscheint, z.B. eine Pressemitteilung.
- **Link**: Dies ist ein Verweis auf ein anderes Element, das intern oder extern zu einer anderen Website sein kann.
- **Bild**: Dies ist ein Bild, z.B. eine GIF- oder JPEG-Datei.
- **Termin**: Dies ist ein kommender Termin, z.B. eine Besprechung, Konferenz, oder etwas anderes.
- **Ordner**: Dies entspricht einem Ordner auf einer Festplatte. Es ist ein Ort für Inhalte, die man später leicht wiederfinden kann.
- **Thema**: Dies ist eine Gruppierung anderer Inhalte. Im Grunde genommen ist es ein gespeichertes Suchkriterium, das Sie später wiederverwenden können. Nur privilegierte Benutzer können Themen erstellen.
- **Datei**: Dies ist ein weiteres Stück Inhalt, z.B. ein Film, ein Sound-Clip, eine Textdatei, ein Spreadsheet, eine komprimierte Datei oder irgendetwas anderes, was Sie hochladen möchten.

Ich werde die Liste all dieser Elemente durchgehen, wobei ich das Dokument als Beispiel nehmen werde, um zu zeigen, wie man Dokumente einfach und schnell erstellt und bearbeitet. Mit Hilfe dieser einfachen Inhaltstypen werde ich zeigen, wie man im Browser eine dynamische Site baut, ohne das Geringste programmieren zu müssen.

Tatsächlich verfügen Sie in Plone über viele Möglichkeiten, Inhalte zu erstellen und zu bearbeiten, nicht nur den Webbrowser. Der Zugang über FTP, WebDAV sowie über Scripts ist ebenfalls möglich. In Kapitel 10 beschreibe ich, wie man das jeweils einrichtet. Bis dahin behandle ich vorläufig nur die Browser-Schnittstelle. In den Kapiteln 11 bis 13 beschreibe ich, wie man einen neuen speziellen Inhaltstyp erstellt, den Sie auf die Bedürfnisse einer speziellen Site zuschneiden können.

3.3.1 Was sind Dokument-Inhaltstypen?

Anstatt für alle verschiedenen verfügbaren Inhaltstypen detailliert zu beschreiben, wie sie erstellt und bearbeitet werden, behandle ich die Erstellung eines Typs, nämlich eines Dokuments, im Detail. Nach der Erstellung und Bearbeitung einiger solcher Dokumente sollte Ihnen das Prinzip ihrer Erstellung in Fleisch und Blut übergegangen sein, und Sie werden ohne weiteres andere Inhalte erstellen können.

Ein *Dokument* ist eine Seite mit einem Inhalt, normalerweise ein eigenständiger Text. Zwar kann man auf jedes in Plone erstellte Element mit einer Webseite zugreifen, und wenn Sie an einen Inhaltstyp für eine Webseite denken, dann ist dies genau ein solcher. Die Standard-Homepage einer Plone-Site, die Sie bereits gesehen haben – die berühmte Plone-Willkommensseite – ist ein Beispiel für ein Dokument (siehe Abbildung 3.8).

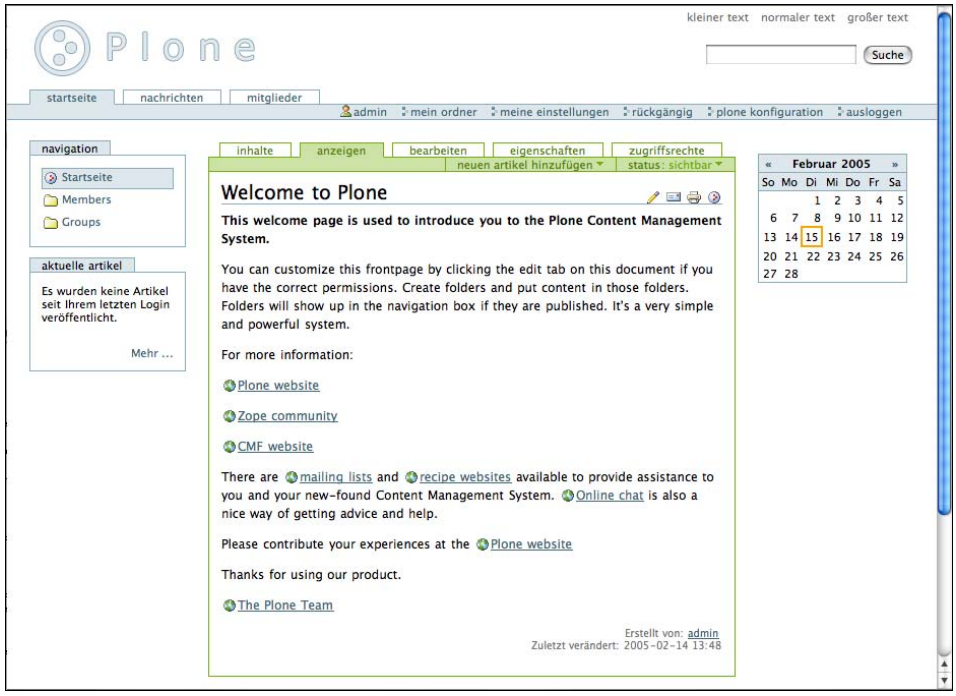


Abbildung 3.8: Willkommen in Plone, ein einfaches Dokument

3.3.2 Erstellen eines Dokuments

Mit einem Webbrowser können Sie auf zweierlei Weise beliebige Inhalte erstellen. Zuerst müssen Sie angemeldet sein, da nur solche Benutzer Inhalte erstellen dürfen. Klicken Sie dann auf den Link MEIN ORDNER in der Navigationsleiste oben rechts. Dadurch gelangen Sie zu Ihrem eigenen Ordner, einem Bereich, den Sie selbst kontrollieren. Wenn Sie Inhalte in einem Ordner erstellen dürfen, dann erscheint dieser Ordner mit einem grünen Rahmen am oberen Rand (siehe Abbildung 3.9).

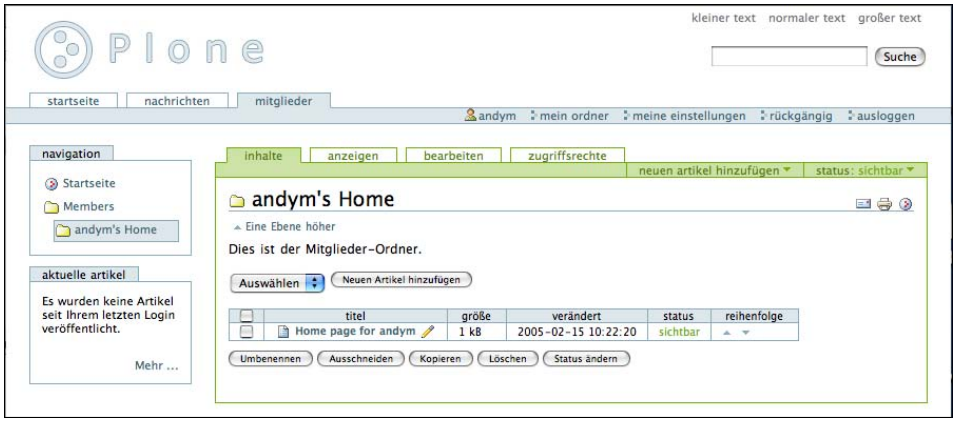


Abbildung 3.9: Mein Inhalt

Falls der grüne Rahmen nicht zu sehen ist, können Sie keine Inhalte hinzufügen. Dieser Rahmen enthält die Aktionen, die Sie an dieser Stelle ausführen dürfen. In Abbildung 3.9 sehen Sie, dass die Seite den Inhalt des Ordners anzeigt, weil das der gewählte Reiter ist. Es sind auch andere Reiter zu sehen wie ANZEIGEN, ZUGRIFFSRECHTE und EIGENSCHAFTEN, die weitergehende Möglichkeiten bieten. In der oberen rechten Ecke des grünen Rahmens sehen Sie die zwei Dropdown-Menüs NEUEN ARTIKEL HINZUFÜGEN und STATUS. Klicken Sie auf das erste Menü, um die Liste von Elementen darin zu sehen (siehe Abbildung 3.10).

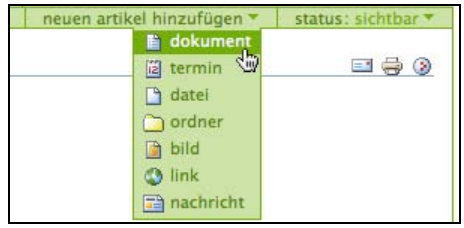


Abbildung 3.10: Erstellen eines Dokuments aus dem grünen Dropdown-Menü

Wählen Sie DOKUMENT, um ein neues Dokument zu erstellen. Alternativ dazu sehen Sie im Rumpf der Seite eine weitere Dropdown-Box namens NEUEN ARTIKEL HINZUFÜGEN. Klicken Sie auch hier auf den Pfeil nach unten, um eine Liste von Elementen zu erhalten, und wählen Sie dann das gewünschte Element (siehe Abbildung 3.11).

Es ist sehr praktisch, die Liste unter NEUEN ARTIKEL HINZUFÜGEN zu benutzen, da sie fast immer verfügbar ist.

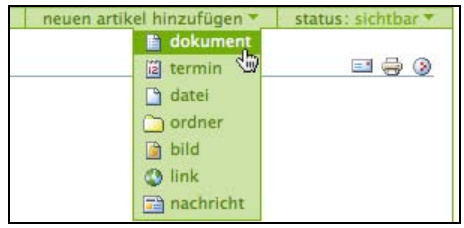


Abbildung 3.11: Erstellen eines Dokuments aus dem Inhalte-Menü des Hauptordners

Achtung

Falls Sie mit Zope vertraut sind, sollten Sie wirklich nie, nie, nie Inhalte aus dem ZMI (Zope Management Interface) heraus hinzufügen. Je nachdem, wie Sie Plone installiert haben, haben Sie das ZMI evtl. bereits gesehen und für die Anpassung und Entwicklung von Plone übers Web benutzt. Beim Erstellen von Inhalten mit dem ZMI werden jedoch Inhaltselemente generiert, die für Plone unvollständig sind und in Plone demnach nicht korrekt funktionieren.

Exkurs: Wo sollen Dokumente erstellt werden?

Zu Beginn erstellt man Dokumente am einfachsten im Benutzerordner eines Site-Mitglieds, der über den Link MEIN ORDNER verfügbar ist. Das ist zwar nützlich, aber wahrscheinlich nicht der beste Ansatz für eine langfristige Lösung. Insbesondere werden dabei lange URLs erzeugt, z.B. `/Members/andy/Docum...`. Es führt auch dazu, dass Ihr Inhalt in der Navigationsleiste nicht richtig erscheint.

Wie Sie später noch sehen werden, gibt es hierfür eine Reihe von Lösungen. Meistens erzeugt man einen Ordner und gibt gewissen Benutzern das Recht, darauf zuzugreifen. Dieser Ordner kann z.B. `Help` oder `News` heißen. Im Abschnitt »Ordner benutzen«, der weiter unten folgt, wird die Erstellung von Ordnern besprochen, und Kapitel 9 behandelt Arbeitsbereiche und Sicherheit für Gruppen.

3.3.3 Bearbeiten eines Dokuments

Nach einem Klick zur Erstellung eines Dokuments gelangen Sie auf die Seite DOKUMENT BEARBEITEN mit einem Hinweis darauf, dass das Dokument erzeugt wurde. Wenn das nicht passiert, können Sie auf ein Dokument und dann auf den

Reiter BEARBEITEN klicken. Wieder sehen Sie, dass der Reiter BEARBEITEN grün hinterlegt wird (siehe Abbildung 3.12).

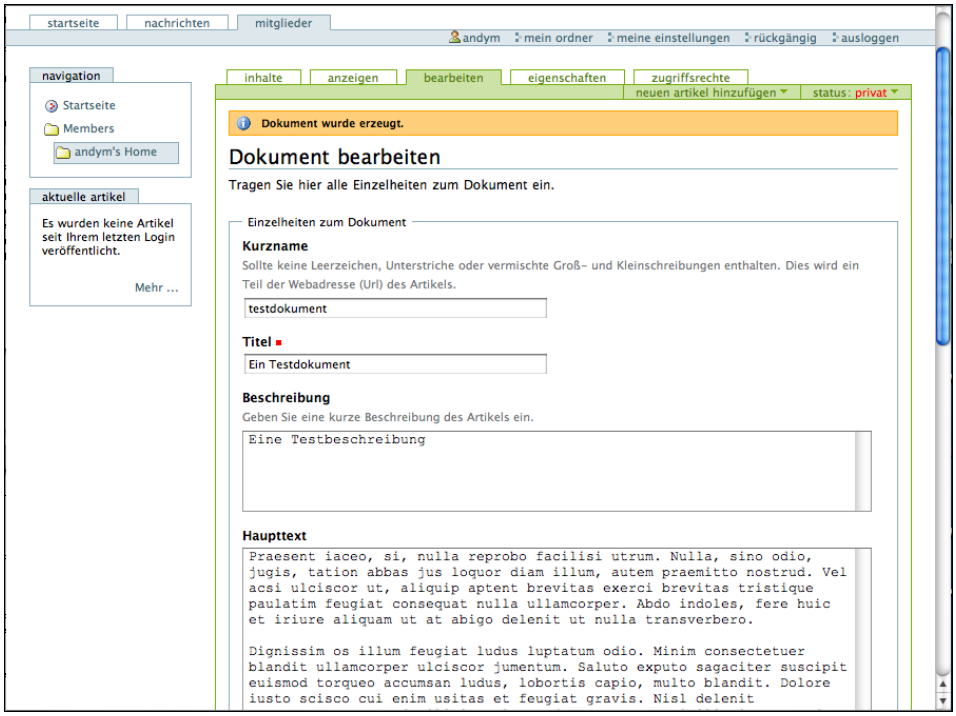


Abbildung 3.12: Bearbeiten eines Dokuments

Nun können Sie das Dokument in Ihrem Webbrowser mit dem vorhandenen Formular bearbeiten. Wenn Sie sich die URL in der Adressleiste Ihres Browsers anschauen, bemerken Sie, dass für Sie ein Kurzname für das Objekt erzeugt wurde, der so ähnlich wie `Document.2003-12-29.43787` aussieht. Folgende Liste führt die Felder und ihre Bedeutung auf:

- **Kurzname**: Der Kurzname wird ein Teil der Dokument-URL, d.h., Sie sollten ihn kurz und selbsterklärend halten, am besten ohne Leerzeichen. Bei Beachtung dieser Regeln lassen sich URLs leichter lesen. Benutzen Sie z.B. etwas wie `audit-report-2003`. Wenn Sie keinen Namen angeben, erzeugt Plone einen für Sie.
- **Titel**: Dies ist der Titel des Elements, der überall auf der Site angezeigt wird (z.B. oben auf der Seite, in der Suchschnittstelle, im Browsertitel usw.). Dieses Feld muss ausgefüllt werden.

Hinweis

Dieses Feld erscheint nicht, wenn Sie auf der Seite mit Ihren Voreinstellungen NEIN für die Kurznamen gewählt haben.

- **Beschreibung**: Dies ist eine kurze Erklärung zum Dokument, normalerweise etwa 20 Worte als Einleitung zum Dokument und als Aufmacher für den Rest des Dokuments. Das ist bei Seiten nützlich, die eine Zusammenfassung des Dokuments anzeigen, z.B. bei Suchergebnissen und Ordnerinhalten.
- **Haupttext**: Dies ist der Rumpf des Dokuments. Das Format des Inhalts wird mit dem FORMAT-Feld eingestellt (wird anschließend beschrieben).
- **Format**: Für das Format des Haupttextes haben Sie drei Möglichkeiten: STRUKTURIERTER TEXT, HTML und EINFACHER TEXT. Diese Textarten werden im Exkurs »Wahl eines Textformats« beschrieben. Wenn Sie sich nicht sicher sind, lassen Sie das Feld so, wie es ist, und geben Sie den Haupttext ganz normal ein.
- **Inhalt hochladen**: Falls Ihr Dokument als Datei auf Ihrem Rechner vorliegt, können Sie es hochladen, anstatt den Inhalt ins HAUPTTEXT-Feld einzutippen. Benutzen Sie den Button DATEI AUSWÄHLEN unten auf der Seite, um eine Datei auszuwählen. Der Inhalt einer hochgeladenen Datei *ersetzt* den gesamten Inhalt im HAUPTTEXT-Feld.

Wenn Sie mit der Bearbeitung Ihres Dokuments fertig sind, klicken Sie auf den Button SPEICHERN, um Ihre Änderungen zu bestätigen. Sie gelangen dann zu dem Reiter ANZEIGEN, wo Sie sehen können, wie das Dokument für die Benutzer aussehen wird (siehe Abbildung 3.13). Um es weiterzubearbeiten, klicken Sie auf den Reiter BEARBEITEN.

Wenn Sie in dem BEARBEITEN-Formular fehlerhafte Eingaben machen, kommen Sie beim Abspeichern zur BEARBEITEN-Seite zurück, wo die Fehler dann hervorgehoben sind. In dem Moment sind Ihre Änderungen nicht wirksam, d.h. Sie müssen die Fehler korrigieren und auf SPEICHERN klicken, damit die Änderungen festgehalten werden. Der in Abbildung 3.13 gezeigte Reiter ANZEIGEN zeigt das erzeugte Dokument an. Wie Sie sehen, werden Titel, Beschreibung und Inhalt in jeweils anderen Stilen angezeigt. Unten auf der Seite ist eine Fußzeile mit Angaben zum Autor des Dokuments sowie dem Datum, an dem das Dokument erzeugt wurde.

Abbildung 3.13: Beim Speichern des Inhalts gelangen Sie zum Reiter Anzeigen.

Sie werden bemerken, dass Sie dann, wenn Sie zum Inhalt von Ordnern zurückgehen, nachdem Sie Ihre Änderungen gespeichert haben, zwei Dokumente in Ihrem Ordner sehen werden: das vorhandene, das für Sie erzeugt wurde, und das neue, das Sie gerade erstellt haben. Sie können beide Dokumente bearbeiten, indem Sie darauf klicken und somit den Reiter ANZEIGEN öffnen, von wo aus Sie den Reiter BEARBEITEN wählen können.

Exkurs: Wahl eines Textformats

Wie schon erwähnt wurde, können Sie den Dokumentinhalt in mindestens drei Formaten bearbeiten: STRUKTURIERTER TEXT, HTML und EINFACHER TEXT. Dieser etwas verwirrende Umstand kommt daher, dass man versucht, einfache Systeme zu produzieren, mit denen die Benutzer angereicherte, ausgezeichnete Inhalte in Form von einfachem Text schreiben können, ohne spezielle Editoren zu verwenden.

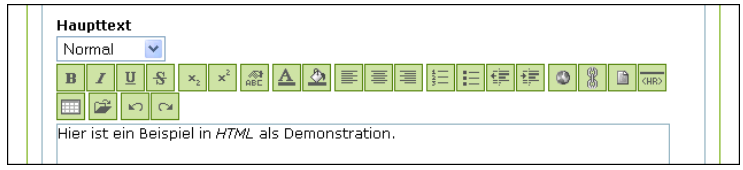
In den meisten Fällen funktioniert das aber leider nicht, und man benötigt einiges an Training, um die Formatierung zu begreifen. Strukturierter Text verlangt für sich genommen schon einiges an Verständnis, weil er eine frustrierende Syntax verwendet und Mängel bei fremdsprachigen Texten aufweist. Wenn ich ein Format allen anderen vorziehen müsste, dann wäre das HTML, weil es von vielen verstanden wird and weil Sie es mit WYSIWYG-Editoren (What You See Is What You Get) wie Epoz generieren können.

HTML

HTML ist das am weitesten standardisierte Format. Wenn ein Dokument in HTML eingegeben wird, wird es im gleichen Format dargestellt. Dieses HTML sollte keine komplette Seite sein, sondern ein Ausschnitt. Beispiel:

```
<p>Hier ist ein Beispiel in <i>HTML</i> als Demonstration.</p>
```

Idealerweise sollte das HTML auch gültiges XHTML (Extensible HTML) sein, damit es zum Rest des Plone-Systems passt. Ansonsten genügen Ihre Seiten nicht den geltenden Webstandards. Die Eingabe von Text in XHTML ist nichts für Unbedarfte, deswegen werden Sie in Kapitel 9 sehen, wie Sie umfangreiche Editorwerkzeuge in Plone integrieren können, mit denen Benutzer sehr einfach XHTML schreiben können. Die folgende Abbildung zeigt den Einsatz von Epoz in Plone, damit Benutzer kein HTML kennen müssen:



Einfacher Text

Einfacher Text ist leicht zu verstehen, denn dabei wird keine große Umwandlung oder Manipulation des eingegebenen Textes vorgenommen. Es ist lediglich einfacher Text. Die einzige Änderung, die vorgenommen wird, ist die, dass Zeilenenden nach HTML umgewandelt werden, damit sie in einem Webbrowser als solche dargestellt werden. Sonst wird nichts verändert. Beispiel:

```
Hier ist ein Beispiel in einfachem Text als Demonstration
```

Strukturierter Text

Strukturierter Text ist ein System zum Schreiben von Dokumenten mit einfachem Text in einem bestimmten Format, das dann auf verschiedene Weisen interpretiert werden kann. Wenn ein Textteil z.B. hervorgehoben werden soll, dann kann er als `*kursiv*` geschrieben werden, was dann *kursiv* angezeigt wird. Mit diesen Regeln kann der Benutzer leicht eine Seite schreiben, die Formatierungsanweisungen enthält. Eine vollständige Liste von Regeln und Beispielen für strukturierten Text finden Sie im Anhang A. Hier ein Beispiel:

```
Hier ist ein Beispiel in *strukturiertem Text* als Demonstration
```

3.3.4 Dokument-Metadaten setzen

Jeder Brocken an Inhalt kann mit einer Reihe von Eigenschaften versehen werden. Diese Eigenschaften werden auch als *Metadaten* bezeichnet und enthalten Angaben z.B. zu Stichworten, Copyright und beteiligten Personen.

Diese gesamte Menge an Eigenschaften ist optional und wird normalerweise nur dann benutzt, wenn es spezielle Anforderungen an diesen Inhalt gibt, besonders deswegen, weil die Person, die diesen Inhalt sieht, diese Informationen normalerweise nicht zu sehen bekommt. Daher ist der Hauptgrund für die Eingabe solcher Informationen der, Aufgaben wie Suche und Kategorisierung zu erleichern.

Wenn Sie auf den grünen Reiter EIGENSCHAFTEN klicken, können Sie auf die Eigenschaften eines Objekts zugreifen. Dieses EIGENSCHAFTEN-Formular verfügt über folgende Felder, die bei allen Inhaltstypen gleich sind:

- **Diskussion erlauben**: Das ermöglicht, dass das Dokument von Benutzern diskutiert oder kommentiert werden kann, die das Recht dazu haben. Wenn der voreingestellte Wert beibehalten wird, wird für diesen Inhaltstyp die Regelung der gesamten Site übernommen.
- **Stichworte**: Jedes Element kann über Stichworte verfügen, mit denen eine Gruppierung und Sortierung möglich ist. Ein Artikel über die letzten Ereignisse in der Politik könnte z.B. die Stichworte *Politik* und *Premierminister* enthalten. Stichworte sind flexibel, und Sie können beliebige Stichworte aus der angegebenen Liste verwenden. Das Plone-System enthält zu Beginn keine Stichworte, aber die Site-Administratoren können neue Stichworte hinzufügen, damit andere Benutzer sie verwenden können.
- **Sperrfrist**: Das Sperrfrist-Datum ist der Tag, an dem ein bestimmter Inhalt erstmalig verfügbar sein soll. Dieses Datum können Sie durch Angabe der Werte im Formular oder durch einen Klick auf das Kalender-Icon, das dann einen Kalender öffnet, und Auswahl eines Datums eingeben (siehe Abbildung 3.14).
- **Löschdatum**: Das Löschdatum ist der letzte Tag, an dem ein bestimmter Inhalt verfügbar sein soll. Normalerweise werden die Felder für Sperrfrist und Löschdatum leer gelassen.
- **Format**: Das ist der MIME-Typ (Multipurpose Internet Mail Extensions) des Elements. Der Begriff *MIME-Typ* bedeutet eine Computer-Definition des Inhaltstyps (z.B. *application/msword* oder *image/jpeg*). Dafür ist ein Vorgabewert gesetzt. Wenn Sie bei diesem Feld unsicher sind, ignorieren Sie es einfach.
- **Sprache**: Dies ist die Sprache, in der das Dokument geschrieben ist. Der Vorgabewert hierfür lautet Englisch.

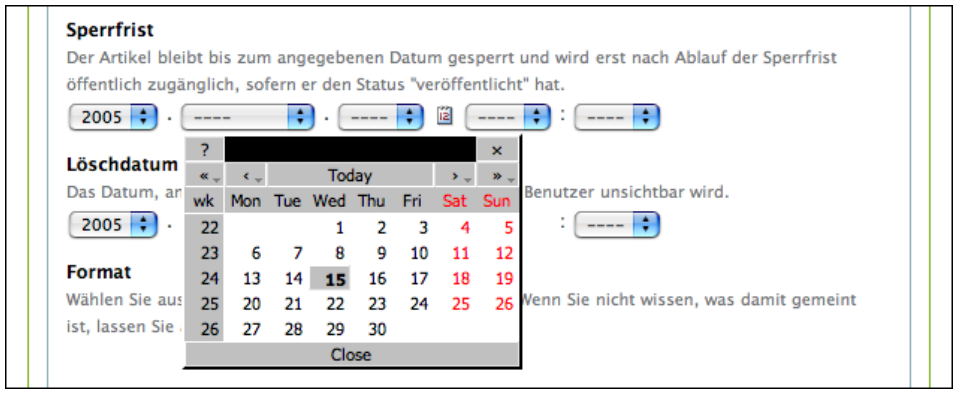


Abbildung 3.14: Eingabe einer Sperrfrist

- **Urheberrechte**: Dies sind Informationen zu den Urheberrechten des Inhalts. Dieses Feld ist üblicherweise leer.
- **Beitragende**: Hierzu gehören die Namen von Leuten außerhalb des Plone-Systems, die etwas zu dem Objekt beigetragen haben. Der Name jeder Person sollte in einer eigenen Zeile stehen.

Nach der Eingabe der Werte in diesem Formular klicken Sie auf SPEICHERN, um die Änderungen zu bestätigen. Wie gesagt: Sie müssen die Werte unter diesem Reiter normalerweise nicht ändern. Die Änderung dieser Werte hängt normalerweise von den Anforderungen an Ihre Site sowie von der Art von Site ab, die Sie entwickeln.

Exkurs: Was sind Sperrfristen und Löschdaten?

Jedes Element im Plone-System verfügt über eine Sperrfrist und ein Löschdatum, wenn die bearbeitende Person das wünscht. Beide sind optional, und wenn Sie die Felder leer lassen, dann werden diese Werte nicht gesetzt.

Ein Beispiel für ein Element mit einer Sperrfrist ist eine Pressemitteilung. In einer idealen Welt würde die Mitteilung in Plone konzipiert, vorbereitet und überprüft. Aber nehmen Sie einmal an, die Mitteilung müsste auf der Website um Mitternacht veröffentlicht werden, was aber genau dann ist, wenn Sie eigentlich schlafen möchten. Das ist kein Problem, wenn Sie der Pressemitteilung eine Sperrfrist bis Mitternacht geben. Bis dahin wird sie weder im Kalender noch im Navigationsteil noch bei Suchvorgängen oder auf Seiten sichtbar sein, die eine Suche als Liste unter dem Reiter NACHRICHTEN benutzen.

Alle, die von der Pressemitteilung wissen, können direkt auf die Seite zugreifen. Sobald die Sperrfrist vorbei ist, erscheint das Element an allen vorher genannten Orten und wird für alle Welt veröffentlicht.

Ähnlich verhält es sich mit Löschdaten. Wenn Sie ein Sonderangebot haben, das an einem bestimmten Tag verfällt, könnten Sie das Löschdatum auf diesen Tag setzen. Nach diesem Datum wird es nicht mehr im Kalender, in der Navigation, in Suchoperationen usw. erscheinen.

Sperrfristen und Löschdaten ändern nicht den Status des Elements im Workflow (siehe Kapitel 8 für weitere Informationen zum Workflow). Sie ändern lediglich, wo es dargestellt wird. Sie können Sperrfristen und Löschdaten auch im STATUS-Reiter setzen, über den Sie im nächsten Abschnitt gleich mehr erfahren.

3.3.5 Veröffentlichen Ihres Dokuments

Wenn ein Dokument erstellt wird, wird ihm ein Anfangszustand namens *Sichtbar* zugewiesen. Inhalte werden standardmäßig nicht automatisch veröffentlicht und aller Welt zugänglich gemacht. Andere können Ihr Dokument sehen, aber es erscheint nicht bei Suchoperationen oder im Navigationsbaum. Dieser Zustand ist nützlich, weil Sie andere Benutzer auf diesen Inhalt hinweisen können, aber da er nicht in der Navigation oder bei Suchvorgängen erscheint, ist er so lange unsichtbar, bis die Benutzer davon erfahren.

Jedes Element in Ihrer Plone-Site hat zu jedem Zeitpunkt einen bestimmten Zustand. Dieser beschreibt dessen Rechte und Rollen innerhalb der Plone-Site. Mit Hilfe von Zuständen kann jedem Element eine andere Sicherheitseinstellung zugewiesen werden. Ein Element kann z.B. manchmal ein oder zwei Wochen brauchen, bis es vorbereitet ist, und es kann dabei in mehreren Versionen vorliegen. Und schließlich möchten Sie den Inhalt veröffentlichen, damit er für alle Benutzer sichtbar wird und in der Navigation und bei der Suche erscheint.

Sie können den Inhalt veröffentlichen, indem Sie das Dropdown-Menü STATUS in der Hauptnavigation oben rechts verwenden (siehe Abbildung 3.15).

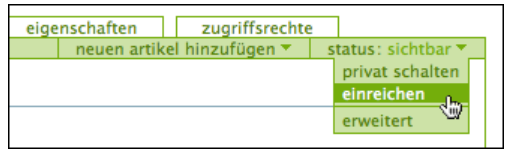


Abbildung 3.15: STATUS-*Dropdown-Menü*

Um ein Objekt zu veröffentlichen, wählen Sie im Dropdown-Menü EINREICHEN. Standardmäßig können Sie Inhalte nicht direkt veröffentlichen, aber Sie können Sie zur Überprüfung einreichen. Dann erhalten sie den Status *Einreichen*. Dies ist ein Zwischenzustand zwischen *Sichtbar* und *Veröffentlicht*. Er erlaubt die Überprüfung von Inhalten durch Benutzer Ihrer Site, die die Rolle eines Prüfers haben, bevor die Inhalte für alle Welt veröffentlicht werden. Nachdem Sie den Inhalt eingereicht haben, werden Sie feststellen, dass der Inhalt im Zustand *Einreichen* ist, wenn Sie in die Box in der oberen rechten Ecke schauen. Sie werden auch bemerken, dass es keinen BEARBEITEN-Reiter mehr gibt, wie in Abbildung 3.16 zu sehen ist.

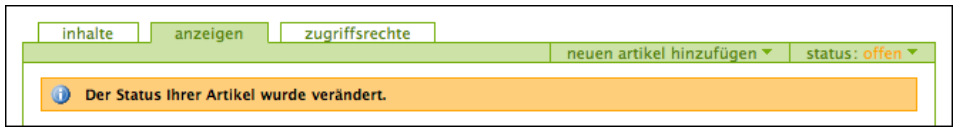


Abbildung 3.16: Der Inhalt wurde eingereicht, der Zustand hat sich auf Offen geändert, und der Bearbeiten-Reiter ist verschwunden.

Hinweis

Wenn Sie als Manager angemeldet sind, werden Sie bemerken, dass es in der Dropdown-Veröffentlichungsliste eine weitere Option namens VERÖFFENTLICHEN gibt. Damit versetzen Sie Inhalte ohne Zwischenschritt direkt in den Zustand *Veröffentlicht*.

In der Workflow-Dropdown-Liste in der oberen rechten Ecke gibt es auch eine Option namens ERWEITERT, die das Statusformular öffnet, mit dem der Status eines Objekts verändert werden kann. Das Formular ist mit jenem identisch, das beim Klick auf den STATUS-Reiter erscheint, und enthält folgende Felder:

- **Sperrfrist**: Dies ist identisch mit dem SPERRFRIST-Feld in den EIGENSCHAFTEN (siehe den Abschnitt »Dokument-Metadaten setzen« weiter oben).
- **Löschdatum**: Dies ist identisch mit dem LÖSCHDATUM-Feld in den EIGENSCHAFTEN (siehe den Abschnitt »Dokument-Metadaten setzen« weiter oben).
- **Kommentare**: Dies sind irgendwelche Kommentare, die Sie zu dieser Zustandsänderung machen möchten, die in der Historie festgehalten werden. Sie könnten z.B. Folgendes schreiben: **Erster Entwurf. Bob, sieh dir bitte den zweiten Absatz an**.

- **Status verändern**: Dies entspricht den verfügbaren Optionen im Dropdown-Menü. Beispieloptionen sind VERÖFFENTLICHEN, EINREICHEN usw. Eine weitere Option, KEINE VERÄNDERUNG, ist für den Fall verfügbar, dass keine Änderung nötig ist.

Wählen Sie den gewünschten Zustandswechsel, und klicken Sie auf SPEICHERN, um dies festzuhalten.

3.3.6 Welche Workflow-Zustände gibt es?

An dieser Stelle fragen Sie sich vielleicht, was dieser *Workflow* überhaupt ist und was die Zustände bedeuten? Wie in Kapitel 8 beschrieben wird, versteht man unter Workflow die Möglichkeit, Inhalten verschiedene Zustände zuzuweisen. Es gibt folgende Standardzustände:

- **Sichtbar**: Neue Inhalte werden im sichtbaren Zustand erzeugt. Alle Benutzer können sichtbare Inhalte über die Suchfunktion finden und direkt darauf zugreifen, indem sie die URL des Objekts besuchen. Sichtbare Inhalte erscheinen nicht im Navigationsbaum, können aber von ihren Besitzern und von Site-Managern bearbeitet werden.
- **Offen**: Offene Inhalte sind Elemente, die von Site-Mitgliedern zur Veröffentlichung eingereicht wurden. Aus Sicht eines Benutzers verhalten sich offene Inhalte wie solche im sichtbaren Zustand. Der Unterschied ist der, dass offene Elemente zur Überprüfung markiert sind. Die Site-Redakteure werden aufgefordert, offene Elemente zu veröffentlichen oder zurückzuweisen. Nur Manager und Redakteure können offene Elemente bearbeiten.
- **Veröffentlicht**: Veröffentlichte Elemente sind für alle Besucher der Site sichtbar. Sie erscheinen in Suchergebnissen sowie im Navigationsbaum. Sie können auch in anderen Bereichen auftauchen, die speziell für diesen Typ existieren (Nachrichten z.B. erscheinen auch dann, wenn Sie auf den NACHRICHTEN-Reiter klicken). Nur Manager können veröffentlichte Elemente bearbeiten, aber ihre Besitzer können sie zur weiteren Bearbeitung zurückziehen (durch das Zurückziehen wird ein Element in den Zustand ÖFFENTLICHER ENTWURF versetzt).
- **Privat**: Elemente im privaten Zustand können nur von ihren Besitzern gesehen und bearbeitet werden – sowie von anderen mit Managerrechten an dem Ordner, in dem sie liegen. Bei anderen Benutzern erscheinen sie nicht in Suchergebnissen oder im Navigationsbaum. Private Elemente können von Managern bearbeitet werden.

3.3.7 Wie werden Inhalte geprüft?

Wenn Sie Redakteur sind, erscheint bei Ihrer ersten Anmeldung in der rechten Spalte Ihrer Homepage eine neue Revisionsliste. Dies ist eine Liste von Elementen, die zur Veröffentlichung eingereicht wurden und von Ihnen oder einem anderen Redakteur geprüft werden müssen (siehe Abbildung 3.17).

Abbildung 3.17: Die Revisionsliste

Diese Revisionsliste erscheint immer dann auf der rechten Seite, wenn Sie sich als Benutzer in der Rolle eines Redakteurs anmelden und es zu prüfende Elemente gibt. In meinem Fall habe ich mich als *admin* angemeldet, d.h. als der Benutzer, der beim Installationsvorgang angelegt wurde. Wenn Ihr Name in der Mitgliederleiste erscheint, wissen Sie, dass Sie angemeldet sind. Die Revisionsliste enthält zu prüfende Elemente; in diesem Fall müssen Sie das Testdokument prüfen. Klicken Sie auf das Dokument, um es zu öffnen. Unmittelbar danach können Sie dreierlei Dinge mit dem Dokument machen:

- **Ablehnen**: Sie lehnen ein Dokument ab, indem Sie im Dropdown-Menü ABLEHNEN wählen. Dadurch wird der Inhalt wieder in den sichtbaren Zustand versetzt, was bedeutet, dass Sie als Redakteur damit nicht glücklich sind. Üblicherweise werden Sie auf ERWEITERT klicken, um das Kommentarformular zu öffnen und eine Begründung dafür zu schreiben, warum Sie es ablehnen.
- **Genehmigen**: Sie genehmigen ein Dokument durch die Wahl von VERÖFFENTLICHEN, was den Inhalt in den Zustand *Veröffentlicht* versetzt. Dabei wird der Inhalt öffentlich verfügbar.
- **Nichts tun**: Indem Sie nichts tun, lassen Sie ein Dokument so, wie es ist. Dabei bleibt der Inhalt weiter in der Schwebe, was manchmal passieren kann, wenn Sie Informationen überprüfen oder mit anderen darüber reden müssen. Irgendwann sollten Sie hierher zurückkommen, um etwas mit diesem Inhalt zu machen, weil er so lange in Ihrer Liste erscheint, bis Sie eine der oben genannten Aktionen ausführen.
- **Bearbeiten**: Bearbeiten Sie das Dokument, und führen Sie dann eine der vorherigen Aktionen darauf aus. Als Redakteur können Sie beliebige Änderungen vornehmen, tun Sie das also, indem Sie auf den Reiter BEARBEITEN klicken.

Sobald Sie den Inhalt einmal aus dem Prüfzustand in den Zustand der Veröffentlichung oder Ablehnung versetzt haben, erscheint er nicht mehr in der Revisionsliste. Natürlich wird dabei angenommen, dass auf Ihrer Site jemand als Redakteur zur Verfügung steht. Normalerweise (aber nicht immer) ist das jener Benutzer, der als Administrator die Plone-Site erstellt hat. In Kapitel 9 beschreibe ich, wie man Benutzer hinzufügt, bearbeitet und einigen Benutzern die Rolle des Redakteurs gibt.

3.3.8 Wie bearbeitet man ein veröffentlichtes Dokument?

Wenn ein Dokument einmal veröffentlicht ist, muss es zurückgezogen werden, damit es erneut bearbeitet werden kann. Wählen Sie dazu im Dropdown-Menü ZURÜCKZIEHEN, wodurch das Element wieder in den sichtbaren Zustand gelangt. Sobald es wieder sichtbar ist, können Sie es wieder bearbeiten und erneut in die Revisionsschlange bringen.

Dieser etwas störende Schritt ist notwendig, um sicherzustellen, dass alle Inhalte diesen Prüfschritt durchlaufen. Sie müssen z.B. sicherstellen, dass alle Bearbeitungsschritte an einer Seite korrekt sind, indem Sie ihren Inhalt prüfen. Benutzer mit der Manager-Rolle können Inhalte jederzeit bearbeiten, d.h., sie können schnell einen Tippfehler korrigieren, ohne diesen Prüfschritt durchzumachen. Dabei nimmt man an, dass Benutzer mit der Manager-Rolle vertrauenswürdig sind! Wenn Sie ein Manager sind, wie er in Kapitel 9 beschrieben wird, können Sie jedes Stück Inhalt anwählen und sehen dort den Reiter BEARBEITEN. Klicken Sie darauf, um Ihre Änderungen vorzunehmen. Dokument:veröffentlichen

3.3.9 Zugriffsrechte an Ihrem Dokument

Hiermit können Sie weitere Rechte an Ihrem Dokument an andere Benutzer oder Benutzergruppen im System vergeben. Dies ist eine Möglichkeit für Fortgeschrittene, die in Kapitel 9 detaillierter behandelt wird.

3.4 Andere Inhaltstypen hinzufügen und ändern

Soeben habe ich im Detail behandelt, wie Dokumente hinzugefügt und bearbeitet werden. Bei allen anderen Inhaltstypen ist es ähnlich. Sie alle verfügen über die gleichen oder ähnliche Aktionen zu ihrer Bearbeitung. Es ändern sich lediglich die Formulare und die Daten darin. In den folgenden Abschnitten werden einige andere dieser Inhaltstypen behandelt. Für alle folgenden Inhaltstypen gilt der gleiche Workflow-Prozess, d.h., sie müssen genauso wie Dokumente veröffentlicht werden.

3.4.1 Bilder erstellen und bearbeiten

Bilder sind grafische Inhalte, die Sie durch die Auswahl von BILD in der Dropdown-Liste hinzufügen. Wenn Sie ein Bild hinzufügen, wechselt der Name des Inhalts zum Namen der Bilddatei. Wenn Sie also ein Bild namens »photo.gif« hinzufügen, kann man in Plone unter `photo.gif` darauf zugreifen. Beim Erstellen und Hochladen eines neuen Bildes können Sie das Bild auf Ihrer Festplatte auswählen, indem Sie auf den Button DATEI AUSWÄHLEN klicken (siehe Abbildung 3.18).

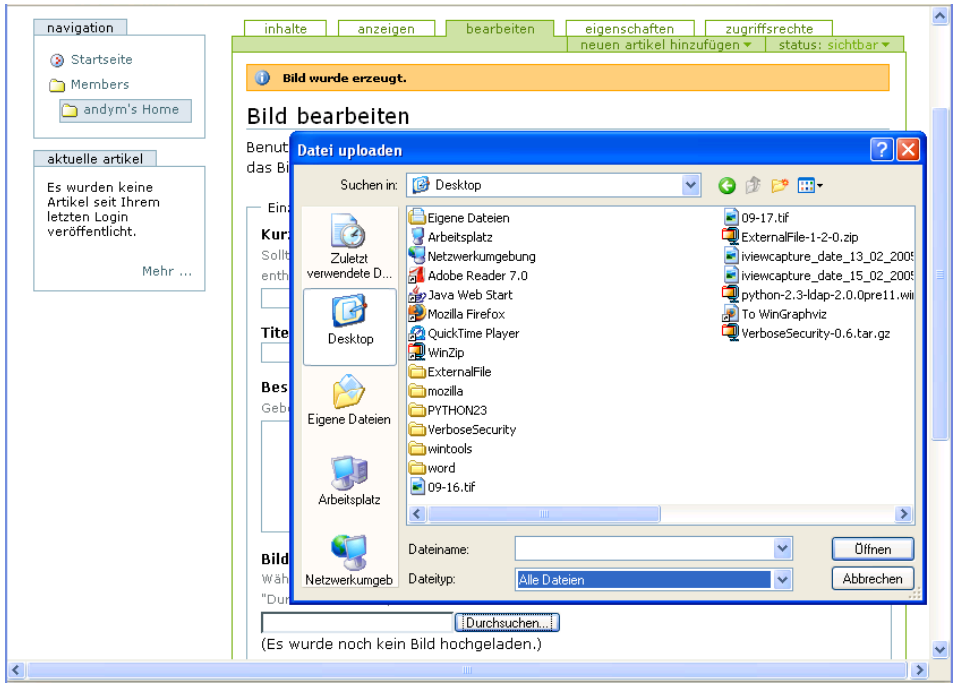


Abbildung 3.18: Ein Bild hochladen

Die Dateinamen von Bildern haben üblicherweise eine Endung, z.B. `.gif`, `.jpg`, `.jpeg`, `.png` oder `.pict`. Unter Plone können Sie Bilder auf einer Webseite darstellen, ohne sie auf Ihren lokalen Rechner herunterladen zu müssen, falls der entsprechende Bildtyp auf dem Webbrowser des Benutzers angezeigt werden kann. Die häufigsten Bildtypen sind `.gif`, `.jpg` und `.png`, die auf fast allen Rechnern angezeigt werden können. Abbildung 3.19 zeigt ein Bild des Plone-Logos.

Ein Bild können Sie nicht direkt bearbeiten. Sie können es aber auf Ihrer Festplatte speichern und dort z.B. mit einem Programm wie Adobe Photoshop oder GIMP (GNU Image Manipulation Program) bearbeiten. Wenn Sie damit fertig sind, können Sie Ihr neues Bild mit einem Klick auf den Reiter BEARBEITEN in

Plone hochladen. Falls Sie sehr viel Bildbearbeitung machen, sollten Sie sich Kapitel 10 anschauen, das ein Werkzeug namens *External Editor* behandelt, mit dem Sie Bilder bearbeiten können, ohne diese hoch- und herunterladen zu müssen.

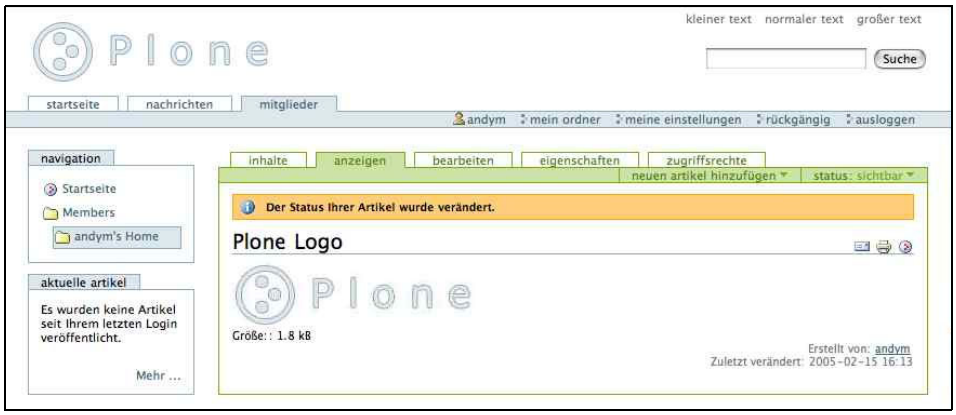


Abbildung 3.19: Anzeigen des Bildes

3.4.2 Dateien hinzufügen und bearbeiten

Eine Datei ist jede beliebige Datei, die Sie von Ihrer lokalen Festplatte hochladen. Dazu wählen Sie DATEI in der Dropdown-Liste. Im Reiter BEARBEITEN sehen Sie den Button DATEI AUSWÄHLEN, mit dem Sie eine Datei von Ihrer Festplatte auswählen können. Das könnte ein beliebiges Element sein, z.B. ein Microsoft Word-Dokument, eine Microsoft Excel-Tabelle, ein ausführbares Programm, ein Adobe Acrobat-Dokument usw. Wenn Sie eine Datei hinzufügen, wechselt in Plone der Name des Elements auf den Namen der hochgeladenen Datei. Wenn Sie also eine Datei namens `book.pdf` hochladen, so ist sie in Plone unter `book.pdf` verfügbar. Abbildung 3.20 zeigt eine Datei mit einfachem Text.

Wenn erkannt wird, dass die Datei Text enthält, wird der Dateiinhalt auf der Webseite angezeigt und kann dort unter dem BEARBEITEN-Reiter editiert werden. Ansonsten kann die Datei auf die lokale Festplatte heruntergeladen werden, was man tun muss, um sie dort zu bearbeiten. Danach kann man sie ins System hochladen. Sie werden bemerken, dass eine Datei auch einen zusätzlichen DOWNLOAD-Reiter hat, mit dem Sie die Datei direkt herunterladen können.

Abbildung 3.20: Hinzufügen einer Datei mit einfachem Text

3.4.3 Termine hinzufügen und bearbeiten

Ein Termin ist etwas, das in der Zukunft passieren wird oder in der Vergangenheit schon passiert ist. In Plone können Sie Termine hinzufügen, die im Kalender erscheinen. Dazu wählen Sie in der Dropdown-Liste TERMIN. Bei Terminen gibt es mehr Angaben als bei den meisten anderen Plone-Objekten. Die meisten davon sind allerdings selbsterklärend (siehe Abbildung 3.21)

Wie üblich ist auch hier TITEL das einzig notwendige Feld, aber wenn Sie möchten, dass der Termin im Kalender erscheint, dann müssen Sie einen Start- und Endzeitpunkt angeben. Termine können mehrere Tage umfassen oder aber in der Vergangenheit liegen, solange nur der Start- vor dem Endzeitpunkt liegt. Um einen Termin einzugeben, wählen Sie die passenden Daten im Dropdown-Menü, oder Sie klicken auf das Datums-Icon, mit dem Sie eine grafische Auswahlmöglichkeit haben.

Nachdem der Termin veröffentlicht ist, erscheint er im Kalender. Wenn Sie die Maus über den Eintrag im Kalender bewegen, erscheinen der Start und das Ende des Termins ebenso wie sein Titel (siehe Abbildung 3.22).

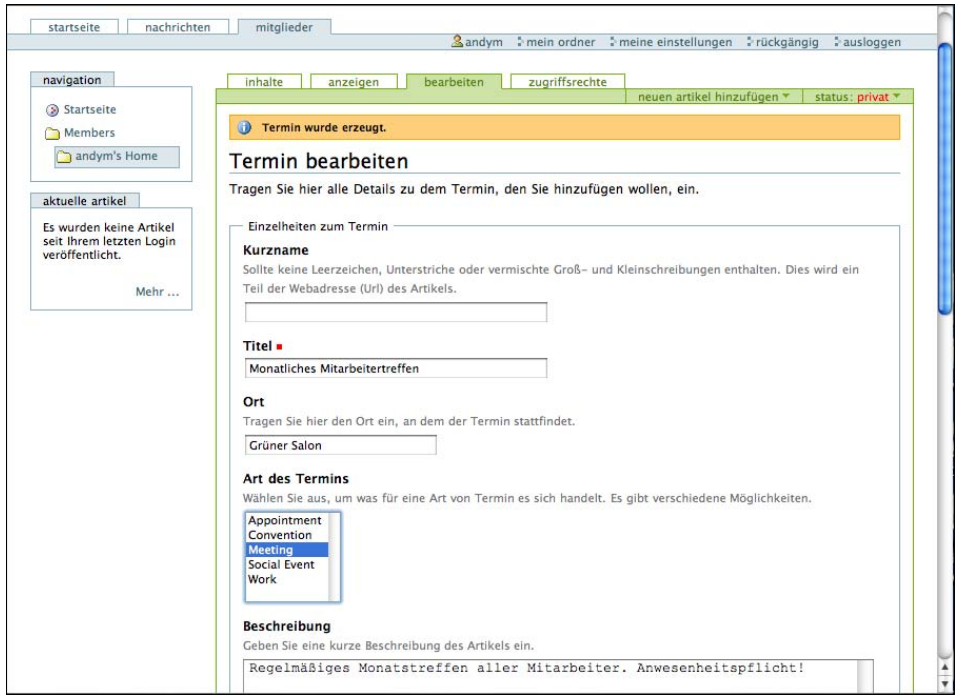


Abbildung 3.21: Hinzufügen eines Termins

Abbildung 3.22: Anzeige von Terminen im Kalender

3.4.4 Links hinzufügen und bearbeiten

Der Inhaltstyp *Link* ist für die Benutzer die wichtigste Art, Links weiterzugeben. Diese URLs können Ressourcen im Internet oder im Intranet sein, eine interne Ressource oder irgendetwas, worauf Benutzer Zugriff haben. Links fügen Sie durch Auswahl von LINK im Dropdown-Menü hinzu.

Wenn Sie einen Link zu einer Ressource im Internet erstellen, sollten Sie bei Ihrem Link das passende Protokoll voranstellen, z.B. `http://`. Wenn ich z.B. eine interessante Seite auf der BBC-Website besuche, die ich an andere weitergeben möchte, könnte ich einen Link darauf erstellen. Der Wert des URL ist der Text in der Adressleiste, z.B. *http://news.bbc.co.uk*, wie in Abbildung 3.23 zu sehen ist.

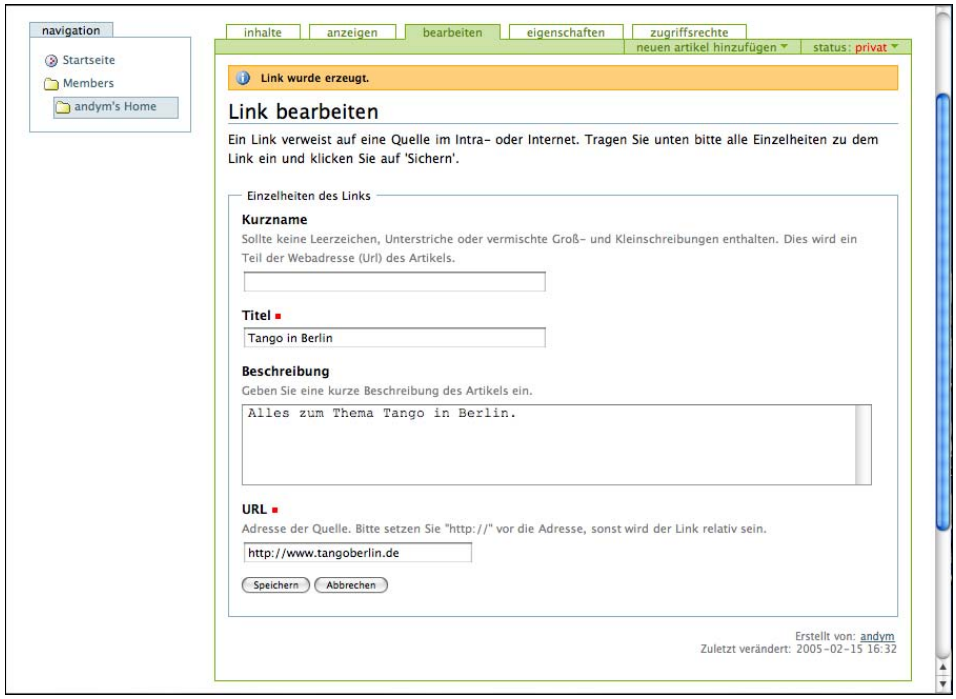


Abbildung 3.23: Hinzufügen eines Links

3.4.5 Nachrichten hinzufügen und bearbeiten

Nachrichten werden auf Websites oft dazu verwendet, dem Leser interessante Neuigkeiten anzuzeigen. Tatsächlich enthält eine Nachricht die gleiche Information wie ein Dokument. Der einzige wirkliche Unterschied ist der, dass eine Nachricht dann angezeigt wird, wenn ein Besucher auf den NACHRICHTEN-Reiter klickt (nachdem die Nachricht veröffentlicht worden ist), was in Abbildung 3.24 zu sehen ist.

Wenn ich eine Webseite schreiben würde, die langfristig von Bedeutung sein soll, z.B. Anfahrtsskizzen zum Büro meiner Firma, würde ich ein Dokument verwenden. Wenn ich hingegen gern eine Seite zu meinem neuen aufregenden Produkt hätte, die Aufmerksamkeit erregen soll, dann würde ich eine Nachricht wählen.

Die Nachricht wäre unter dem Reiter NACHRICHTEN sichtbar, und wenn neuere Nachrichten hinzukommen, würde sie sich langsam nach unten bewegen.

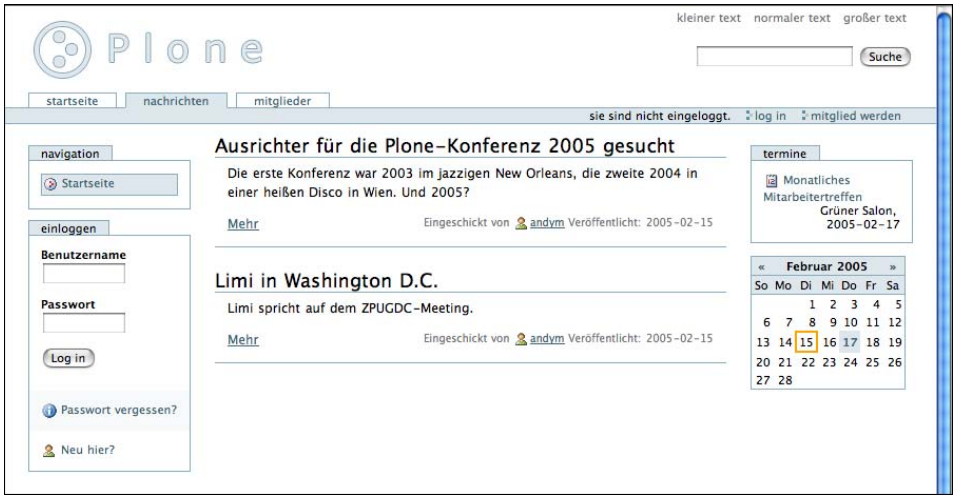


Abbildung 3.24: Eine Liste von Nachrichten

3.5 Inhalte organisieren

Bislang haben Sie gesehen, wie man Inhalte in einer Plone-Site erstellt und bearbeitet, aber ohne eine klare Organisation kann das sehr schnell unübersichtlich werden. Im Wesentlichen haben Sie zwei Möglichkeiten, Ihre Inhalte zu organisieren: Ordner und Themen. Ein *Ordner* ist der einfachste und mächtigste Mechanismus zur Organisation von Inhalten und funktioniert wie ein Ordner bzw. ein Verzeichnis auf der Festplatte eines Rechners. Ein Ordner kann beliebige Inhalte umfassen, Inhalte können zwischen Ordnern hin- und herkopiert und -bewegt werden, und Ordner dürfen natürlich auch andere Ordner enthalten.

Um Inhalte zu organisieren, die über die ganze Site verteilt sind, kann man einen weiterentwickelten und seltener verwendeten Mechanismus in Form von *Themen* verwenden. Ein Thema durchsucht Ihre Website und findet alle Objekte, die auf ein bestimmtes Kriterium passen. Dadurch können Sie viele verschiedene Inhalte zusammenfassen.

3.5.1 Ordner benutzen

Ein Ordner verhält sich wie ein Ordner bzw. ein Verzeichnis auf einer Festplatte, mit dem Unterschied, dass er samt Inhalt innerhalb von Plone existiert. Diese Ordner können Sie genauso benutzen, indem Sie Inhalte gruppieren und in einen

Ordner legen, etwa um sie zu kategorisieren oder um mehr Struktur zu schaffen. Einen Ordner fügen Sie auf Ihrer Site hinzu, indem Sie in der Dropdown-Liste ORDNER wählen. Dadurch wird ein Ordner hinzugefügt und Sie gelangen zum Formular für die Eingabe der Ordnereigenschaften. Ein Ordner hat nur drei einfache Attribute, die der Benutzer bearbeiten kann: Name, Titel und Beschreibung. Diese habe ich bereits für Dokumente beschrieben, und sie unterscheiden sich nicht von denen für Ordner.

Ordner verfügen über zwei grüne Reiter, die leicht unterschiedliche Ansichten bieten: INHALTE und ANZEIGEN. Vielleicht haben Sie sogar schon bemerkt, dass es zu jedem auf der Site erstellten Inhalt einen INHALTE-Reiter gibt. Als Sie z.B. ein Dokument bearbeitet haben, gab es dort auch einen INHALTE-Reiter. Mit diesem Reiter gelangen Sie immer zum Inhalt eines Ordners.

3.5.2 Inhalt eines Ordners anzeigen

Ein Ordner kennt das Konzept einer Standardseite, d.h. einer Seite, die der Benutzer sieht, wenn er einen Ordner anzeigen lässt. Dieses Konzept stammt von Websites, bei denen beim Anzeigen eines Ordners eine Standardseite angezeigt wird, sofern eine vorhanden ist. Der Name dieser Standardseite lautet oftmals `index.htm` oder `index.html`. Falls ein Ordner über eine Standardseite verfügt, wird beim Klick auf den Reiter ANZEIGEN diese Standardseite angezeigt. Wenn der Ordner keine Standardseite hat, wird eine Liste mit dem Inhalt dieses Ordners angezeigt. Bei der Suche nach einer anzuzeigenden Standardseite sucht Plone im Ordner nach einem Inhalt mit einem bestimmten Namen und zeigt diesen an. Der Seitenname lautet üblicherweise `index.html` oder `index_html`, allerdings kann der Site-Administrator diese Namen ändern oder welche hinzufügen.

Mit dieser Inhaltsansicht eines Ordners kann der Benutzer eine Vielzahl von Aufgaben erledigen, z.B. Inhalte verschieben, umbenennen, löschen, veröffentlichen und seine Reihenfolge in der Liste ändern. Wie Abbildung 3.25 zeigt, sehen Sie auch eine einfache Tabelle des Ordnerinhalts. Jede Zeile enthält den Titel des Inhalts (plus Icon), seinen Typ, seine Größe, eine Angabe dazu, wann er zuletzt geändert wurde, seinen aktuellen Workflow-Status sowie Sortierkriterien. Mit den Kästchen links können Sie die gewünschten Einträge auswählen, auf die Sie eine der unten aufgeführten Aktionen ausführen: UMBENENNEN, AUSSCHNEIDEN, KOPIEREN, LÖSCHEN und STATUS ÄNDERN. Diese sind alle recht selbsterklärend, und sie können auf mehrere Objekte gleichzeitig angewendet werden, wenn die entsprechenden Kästchen angekreuzt sind.

Um einen Inhalt z.B. schnell umzubenennen, klicken Sie auf das Kästchen dieses Eintrags und klicken auf UMBENENNEN. Danach gelangen Sie zum UMBENENNEN-Formular, in dem Sie den Titel jedes Listeneintrags ändern können. Klicken Sie auf ALLE UMBENENNEN, damit die Änderung wirksam wird. Mit den Buttons

AUSSCHNEIDEN und KOPIEREN können Sie Inhalte zwischen verschiedenen Ordnern kopieren oder verschieben. Mit dem Button LÖSCHEN können Sie Elemente aus Plone entfernen. Genau wie auf Ihrer Festplatte werden beim Kopieren, Verschieben oder Löschen eines Ordners dessen Inhalte mitkopiert, -verschoben oder -gelöscht.

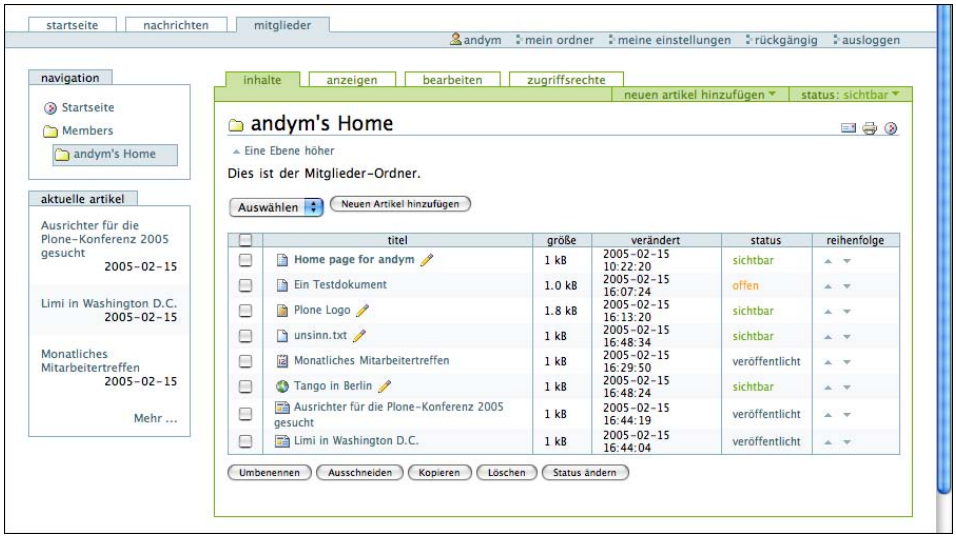


Abbildung 3.25: Inhalt eines Ordners, nachdem einige der vorher beschriebenen Inhaltstypen hinzugefügt worden sind

Eine neue Eigenschaft von Plone 2 besteht darin, die Standardreihenfolge von Ordnerinhalten zu ändern. Standardmäßig werden Ordnerinhalte in der Reihenfolge angezeigt, in der sie erstellt wurden. Falls ein Eintrag besonders wichtig ist und nach oben verschoben werden muss, können Sie das mit den Pfeilen rechts in der Tabelle tun. Die folgenden Eigenschaften erscheinen in Ordnerinhalten nur dann, wenn bestimmte Bedingungen erfüllt sind:

- Falls für den Inhalt ein Löschdatum gesetzt ist, das verstrichen ist, erscheint das Wort *abgelaufen* in Rot neben dem Eintrag.
- Falls auf dem Server External Editor installiert ist, können Sie den Bleistift anklicken, um eine Bearbeitung in External Editor vorzunehmen (siehe Kapitel 10).
- Falls der Inhalt gesperrt ist, erscheint ein Schloss-Icon neben dem Inhalt.

3.5.3 Veröffentlichen eines Ordners

Ordner verfügen über einen wesentlich einfacheren Workflow als Dokumente. Weiter oben in diesem Kapitel haben Sie gesehen, wie man Inhalte veröffentlicht, damit sie öffentlich sichtbar werden, weil Benutzer Inhalte auf diese Weise erzeugen und bearbeiten können, bevor sie allgemein freigeschaltet werden. Ordner sind allerdings insofern anders, als sie weitere Inhalte umfassen, aber selbst keinen Inhalt haben. Aus diesem Grund haben Ordner keinen Zustand *Überprüfen*. Jeder kann private Ordner direkt veröffentlichen oder anlegen, d.h., es gibt drei Zustände: *Privat*, *Sichtbar* und *Veröffentlicht*.

Wählen Sie in der Dropdown-Liste VERÖFFENTLICHEN, nachdem Sie einen Ordner hinzugefügt haben. Danach erscheint er in der Navigation. Gemäß früherer Workflow-Regeln erscheint er erst dann in der Navigation, wenn er veröffentlicht wurde.

3.5.4 Themen verwenden

Mit einem Thema können Sie Inhalte aus verschiedenen Orten der gesamten Plone-Site sammeln und an einem Ort anzeigen. Themen funktionieren über ein Kriterium, das auf alle Objekte zutrifft, die Sie sammeln möchten. Ein Kriterium könnte z.B. sein: alle Bilder oder Nachrichten, bei denen *Plone* im Text vorkommt. Da Themen ein ziemlich komplexer Inhaltstyp sind, können sie zu Beginn nur von Managern erstellt werden. Wenn Sie in der Liste der zu erstellenden Elemente kein Thema sehen, so haben Sie keine Berechtigung dafür.

Um ein Thema hinzuzufügen, wählen Sie im Dropdown-Menü THEMA. Danach können Sie die Schlüsselkriterien unter dem Reiter KRITERIEN eingeben. Die Listen für die Kriterien und ihre Typen ist in den Dropdown-Menüs unten auf der Seite zu sehen. Diese Liste ist ziemlich verwirrend, deswegen versuche ich erst gar nicht, sie hier zu behandeln. Was diese Begriffe bedeuten und wofür sie stehen, basiert leider ganz extrem auf der darunter liegenden Technologie von Katalogindizes und Objektattributen. Aus diesem Grund wird das in Kapitel 11 behandelt.

Um beispielsweise ein Thema zu erstellen, das alle Bilder anzeigt, müssen Sie ein Kriterium hinzufügen, das nach Inhalten sucht, die auf `portal_type` basieren. Wählen Sie dazu den Feldnamen `portal_type` sowie den Kriterientyp `String Criterion`, und klicken Sie dann auf HINZUFÜGEN. Diese Kriterien werden oben auf der Seite hinzugefügt. Geben Sie im Feld neben `portal_type` den Wert **Image** ein, und klicken Sie auf SPEICHERN. Nun haben Sie die Themenkriterien, mit denen alle Bildinhalte angezeigt werden. Wenn Sie zurück zum Reiter ANZEIGEN gehen, können Sie alle Bilder auf der Site sehen.

Themen sind, wie schon erwähnt wurde, sehr komplex, haben eine recht unfreundliche Schnittstelle und sind nur erfahrenen Benutzern zu empfehlen. Viele Leute finden Themen sehr hilfreich, was der Grund dafür ist, dass sie noch immer in Plone existieren. Ein benutzerfreundlicheres System wird aber in Zukunft noch entwickelt werden.

3.6 Inhalte diskutieren und finden

Inhalte in Plone zu erstellen und zu bearbeiten wird wesentlich nützlicher, wenn die Leute diese Inhalte finden und dann auch diskutieren können. Am ehesten finden Benutzer Inhalte über die Suche und Navigation. Plone richtet Suche und Navigation glücklicherweise automatisch für die Benutzer ein, d.h., man kann die erstellten Inhalte auf einfache Weise finden.

3.6.1 Inhalte kommentieren

Feedback von Benutzern ist ein wichtiger Bestandteil jeder Website. Dadurch, dass Benutzer Kommentare hinzufügen können, ermöglichen Sie es ihnen, Feedback zu geben, Tippfehler zu korrigieren oder die Inhalte anderweitig zu diskutieren. In Plone können Sie fast alle Inhalte diskutieren, mit Ausnahme von Ordnern und Themen.

Diskussionen können Sie auf zweierlei Arten ermöglichen. Zum einen kann der Besitzer eines Inhalts (d.h. derjenige, der ihn erstellt hat) die Diskussionsmöglichkeit einschalten, indem er den EIGENSCHAFTEN-Reiter des Objekts anklickt und unter DISKUSSION ERLAUBEN den Punkt EINGESCHALTET wählt, wie in Abbildung 3.26 zu sehen ist. Zum anderen bestimmt die vom Site-Administrator definierte Standardeinstellung die Regelung für diesen Inhaltstyp. Diese Einstellung durch den Administrator wird in Kapitel 10 beschrieben.

Nachdem Diskussionen eingeschaltet sind, klicken Sie auf den Button KOMMENTIEREN, um den Inhalt zu diskutieren. Dazu erscheint dann ein Formular (siehe Abbildung 3.27).

Geben Sie das Stichwort und den Haupttext Ihres Kommentars ein. Der Text wird als einfacher Text eingegeben, d.h., Sie können ihn ganz normal eintippen. Kommentare werden nicht vom Workflow-System erfasst, d.h., sie erscheinen gleich nach ihrer Eingabe. Nachdem ein Kommentar eingegeben wurde, kann man darauf antworten, was eine verkettete Liste von Kommentaren zu einem Inhalt ergibt. Außerdem werden Kommentare im Katalog eingetragen, wodurch in ihnen auch gesucht werden kann.

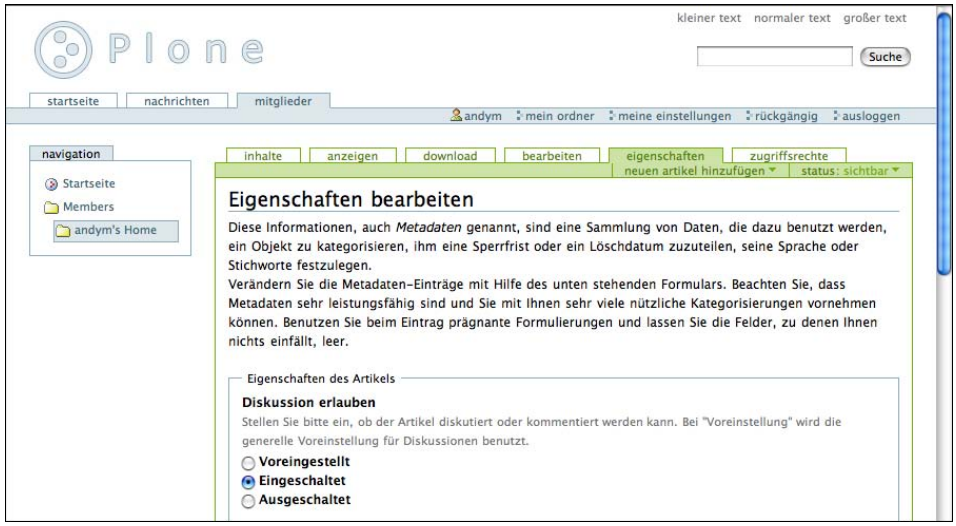


Abbildung 3.26: Diskussionen einschalten

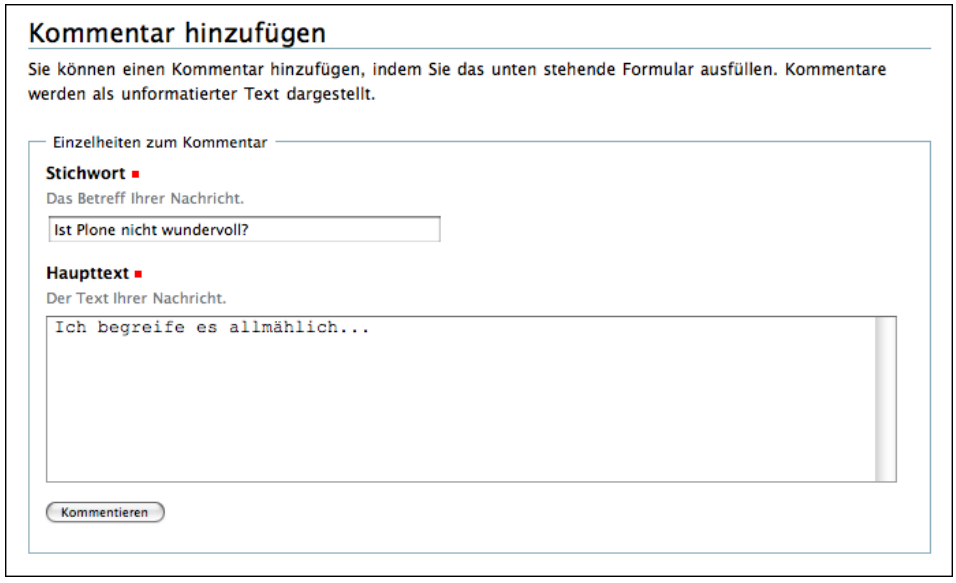


Abbildung 3.27: Kommentar zu einem Inhalt erstellen

Hinweis

Administratoren, die als Manager angemeldet sind, können einzelne Antworten oder gesamte Ketten entfernen. Beim Ausschalten von Antworten werden die Kommentare nicht gelöscht, sie werden lediglich nicht mehr angezeigt. Das heißt, durch das erneute Einschalten von Kommentaren werden vorhandene Kommentare wieder sichtbar.

3.6.2 Nach Inhalten suchen

Plone verfügt über eine mächtige Suchmaschine auf der Basis von Zopes ZCatalog. Mit dieser Suchmaschine kann Inhalt auf mehrere Arten katalogisiert werden, und es kann effizient und schnell danach gesucht werden. Kapitel 10 enthält Details darüber, wie sie funktioniert und abgefragt werden kann.

Wenn ein Benutzer nach Inhalten sucht, werden ihm solche dann angezeigt, wenn sie sich in einem von zwei Zuständen befinden: *Veröffentlicht* oder *Sichtbar*. Ganz oben auf einer Plone-Seite befindet sich ein Suchfeld, mit dem man – wie bei einer Internet-Suchmaschine – sehr leicht eine einfache Textsuche durchführen kann (siehe Abbildung 3.28). Geben Sie z.B. **Tuesday** ein, um alle Inhalte zu finden, die das Wort *Tuesday* enthalten. Dann wird das Ergebnis aller passenden Inhalte angezeigt. Klicken Sie auf einen Titel, um dorthin zu gelangen.

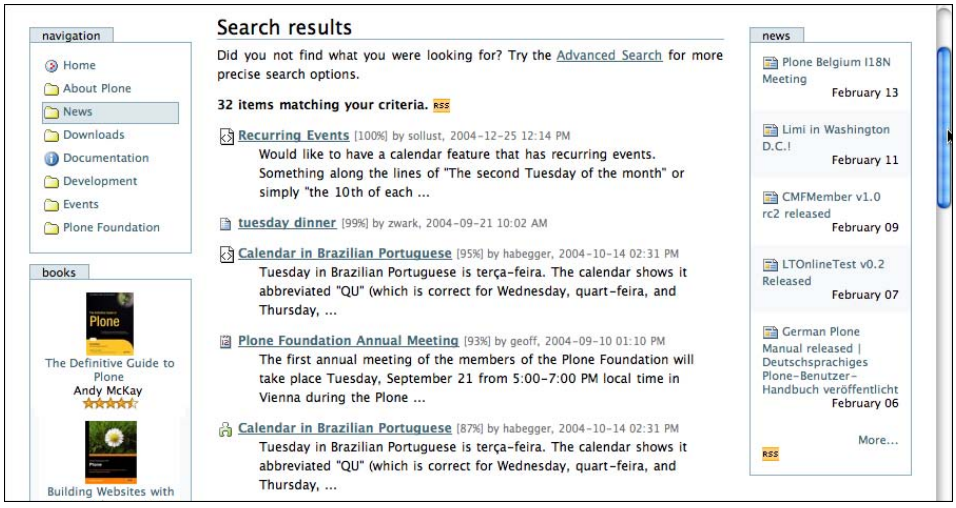


Abbildung 3.28: Eine Suche nach Tuesday auf Plone.org

Diese Suche bietet recht ausgefeilte Möglichkeiten, die denen der meisten Suchmaschinen recht ähnlich sind. Diese einfache Abfrage können Sie um einiges komplexer machen, indem Sie z.B. die folgenden Optionen verwenden:

- **Platzhalter**: Verwenden Sie einen Stern für beliebig viele Buchstaben. So passt z.B. `Dien*` auf `Dienstag` und `Dienstage`. Am Wortanfang können Sie allerdings keinen Stern verwenden.
- **Einzelne Joker**: Verwenden Sie ein Fragezeichen für einen Buchstaben. Beispiel: `Ro?e` passt auf `Rose`, `Robe`, `Rote` usw. Am Wortanfang können Sie jedoch kein Fragezeichen verwenden.
- **And**: Mit `and` geben Sie an, dass die beiden Begriffe rechts und links davon vorhanden sein müssen. Beispiel: `Rom and Dienstag` gibt nur Ergebnisse zurück, in denen beide Wörter im Inhalt vorkommen.
- **Or**: Mit `or` geben Sie an, dass mindestens einer der Begriffe vorkommen muss. Beispiel: `Rom or Dienstag` gibt Ergebnisse zurück, falls eines der Wörter im Inhalt vorkommt.
- **Not**: Mit `not` erhalten Sie Ergebnisse, in denen der Begriff nicht vorkommt (es wird ein `and` als Präfix benötigt). Beispiel: `Willkommen and not Seite` würde passende Seiten zurückgeben, in denen `Willkommen`, aber nicht `Seite` vorkommt.
- **Sätze**: Sätze können Sie in doppelten Anführungszeichen (") setzen, um damit mehrere Wörter nacheinander anzugeben. Beispiel: `"Diese Seite"` passt auf `Diese Seite führt Sie ins Plone Content-Management-System ein.`, aber nicht auf `Diese Startseite von...`.
- **Negierte Sätze**: Sie können einem Satz ein Minuszeichen (-) als Präfix voranstellen. Beispiel: `Start -"Diese Seite"` passt auf alle Seiten, in denen `Start` vorkommt, aber nicht `Diese Seite`.

Hinweis

Bei jeder Suche ist die Groß-/Kleinschreibung nicht von Bedeutung.

Bei großen Sites kann es eine Menge Ergebnisse geben, daher werden nur jeweils 20 auf einmal angezeigt. Um durch die Ergebnisse blättern zu können, erscheinen Navigationsleisten oben und unten auf den Suchergebnisseiten. Die Werte eines Objekts, die bei der Suche verwendet werden, sind dessen Titel, Beschreibung

und Haupttext (falls der Inhaltstyp einen hat, das trifft z.B. auf Nachrichten und Dokumente zu).

3.6.3 Durchführen einer erweiterten Suche

Sie können die Suchergebnisse dadurch einschränken, dass Sie eine erweiterte Suche durchführen, die bei den Suchergebnissen einer Standardsuche verfügbar ist. Auf alten Plone-Sites gelangten die Benutzer mit einem SUCHEN-Reiter dorthin. Wenn Sie möchten, können Sie diesen reaktivieren, was in Kapitel 4 beschrieben wird. Mit Hilfe des Formulars ERWEITERTE SUCHE kann der Benutzer mit einer Reihe von Attributen nach Inhalten suchen, darunter Titel, Stichworte, Beschreibung, Revisionsstatus, Erstellungsdatum, Inhaltstyp und sogar Autor – neben dem Suchtext, der auch bei der schnellen Suche mit dem Feld in der oberen rechten Ecke verwendet wird (siehe Abbildung 3.29).

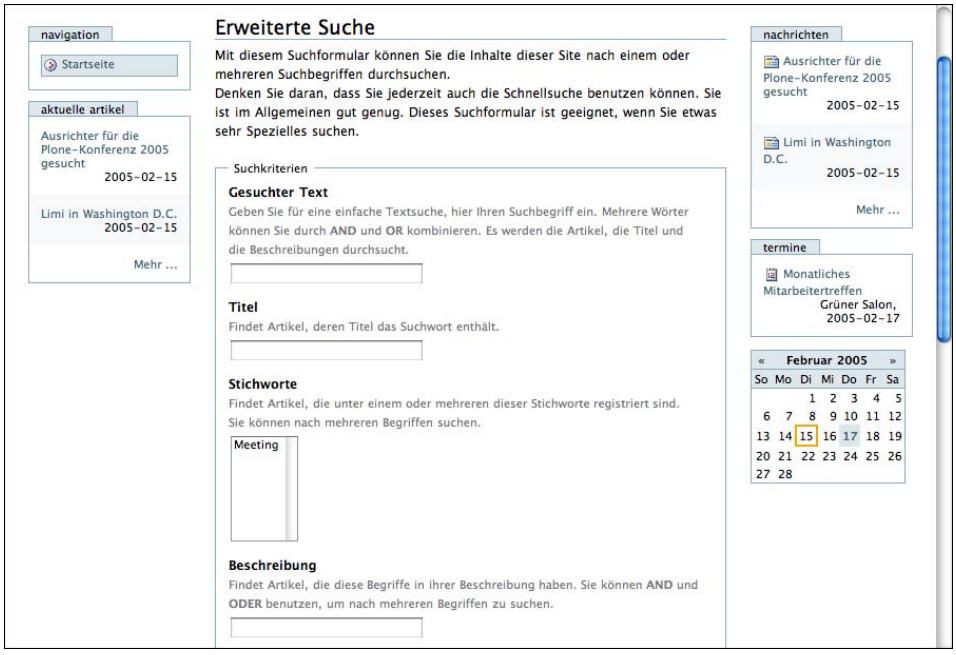


Abbildung 3.29: Erweiterte Suche

Obwohl die Angabe im Suchtextfeld sowohl im Titel als auch in der Beschreibung gesucht wird, möchten Sie evtl. nur im Titel oder nur in der Beschreibung suchen. Aus diesem Grund gibt es diese Felder im Formular ERWEITERTE SUCHE. Joker, Platzhalter und alle weiteren Optionen einer erweiterten Suche können Sie bei der Suche in Titeln und Beschreibungen nicht verwenden. Jedes Suchergebnis

passt in der Eingabe (sofern vorhanden) auf alle Felder, und das Ergebnis ist die Schnittmenge aller Begriffe.

3.7 Beispiel: Erstellen der Website zum Plone-Buch

Um Ihnen ein Beispiel für eine Plone-Site und eine Reihe einzelner Beispiele zu geben, habe ich für dieses Buch eine Website eingerichtet. Es ist eine Plone-Site mit einigen wenigen Änderungen. Bei der Behandlung der einzelnen Buchteile werde ich auf diese Site verweisen und neue Eigenschaften hinzufügen, wie sie im Buch behandelt werden, darunter neue Templates, Skins usw. Die Website zu diesem Buch finden Sie unter *http://plone-book.agmweb.ca*. Zu Beginn habe ich sie auf einem Windows-Server erstellt, wie in Kapitel 2 beschrieben ist. Später habe ich sie jedoch auf Linux übertragen.

Diese Site hat folgende Ziele:

- Sie bietet Leuten einen Ort, an dem sie Informationen zum Buch erhalten und darüber, wo sie es kaufen können.
- Sie bietet einfachen Zugang zur Software, die in diesem Buch benutzt wird.
- Sie bietet Code-Beispiele und ermöglicht den Benutzern, die Beispiele im Buch auszuprobieren.
- Sie enthält Errata oder sonstige Probleme, die nach der Publikation gefunden werden.

Nach der Einrichtung einer Plone-Site habe ich folgende einfache Order- und Seitenstruktur erstellt:

```
Home
 |_ Software
 |_ Chapters
    |_ Chapter 1
    |_ Chapter 2
    ...
```

Dazu habe ich mich als jener Benutzer angemeldet, der vom Installationsprogramm angelegt wurde. In meinem Fall ist das der Benutzer admin. Nach der Anmeldung bin ich auf die Homepage gegangen, habe auf den BEARBEITEN-Reiter geklickt und habe etwas Text für die Homepage geschrieben. Dann habe ich Links zu den Ordnern `Chapters` und `Software` hinzugefügt. Und schließlich habe ich auf den INHALTE-Reiter geklickt und zwei Ordner hinzugefügt, wie in Abbildung 3.30 zu sehen ist.

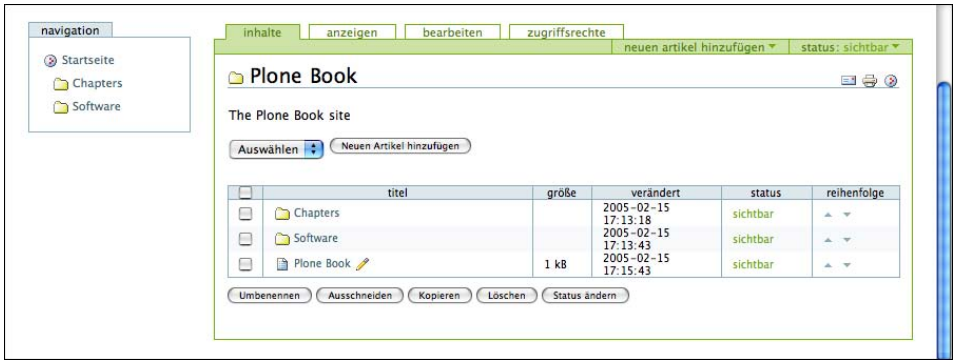


Abbildung 3.30: Der Ordnerinhalt mit meiner Homepage und den neuen Ordnern

Anschließend bin ich in den Ordner `Chapters` gegangen und habe angefangen, für jedes Kapitel einen Ordner zu erstellen. Da ich keine Standardseite erstellt habe, erstellt Plone von sich aus eine Liste mit allen Kapiteln. Die Beschreibung eines Kapitels besteht jeweils aus dem Kapitelnamen (z.B. *Einführung in Plone*), und der Kurzname besteht aus der Kapitelnummer – dadurch bleiben meine URLs hübsch kurz (z.B. `/Chapters/3`). Ich habe alles im sichtbaren Zustand belassen, damit man anschließend sofort Inhalte erstellen kann.

4 Einfache Anpassungen vornehmen

Nachdem Sie nun wissen, wie Sie Inhalte erstellen und bearbeiten, werden Sie anfangen wollen, Ihre Site anzupassen. In diesem Kapitel erfahren Sie, wie Sie mit den Möglichkeiten eines Systemadministrators einfache Anpassungen in Plone vornehmen können. Zu diesem Zweck muss ein Benutzer mit Manager-Rechten angemeldet sein (siehe dazu auch Kapitel 2).

All diese Anpassungen sind Optionen bei der Konfiguration, die Sie über das Web einstellen können. Anstatt sie alle im Detail einzeln zu erklären, soll dieses Kapitel Ihnen einen Überblick über viele Themenbereiche geben und erklären, wie man gewisse Aufgaben löst. Währenddessen sollen Sie einen Blick unter die Haube werfen können. Diese Themen werden dann im weiteren Verlauf des Buches noch erweitert und erklärt.

Am Anfang bringt es am meisten, ins Plone-Control Panel (den Konfigurationsbereich von Plone) zu schauen. Alle Teile einer Plone-Site sind so entworfen, dass sie sich leicht ändern und anpassen lassen. Die blauen Reiter, die Sie am oberen Rand der Seite sehen können, sind leicht hinzuzufügen und zu entfernen. Andere Beispiele hierfür sind die Kästen in der linken und rechten Spalte, die auch *Portlets* genannt werden. Plone enthält mehrere Portlets, und Sie können ganz leicht angeben, wo diese angezeigt werden sollen.

Am Ende dieses Kapitels erfahren Sie, wie Sie auch Cascading Style Sheets (CSS) und Bilder in Plone anpassen können. In einer Plone-Site wird alles durch CSS beinflusst. Tatsächlich werden Sie in diesem Kapitel sehen, dass alle Farben, alle Positionsangaben und viele der Bilder, die Sie sehen können, dadurch bestimmt werden. Wenn Sie in der Lage sind, CSS-Code zu verändern, dann können Sie fast das komplette Look-and-Feel einer Plone-Site ändern. Alle in diesem Kapitel behandelten Optionen zeigen Ihnen, welch große Gestaltungsmöglichkeiten Sie in Ihrer Plone-Site haben.

4.1 Sites verwalten

Als Erstes sollten sich Systemadministratoren das Plone-Control Panel anschauen. Hiermit hat man Zugriff auf einige der Verwaltungsfunktionen einer Site, z.B. auf den Namen und die Beschreibung Ihrer Plone-Site, die Benutzer- und Gruppenverwaltung sowie auf irgendwelche eventuellen Fehler in Ihrer Site.

Der Begriff *Control Panel* wird sehr häufig benutzt, daher sollten Sie ihn nicht mit dem Control Panel im Zope Management Interface (ZMI) verwechseln, das die unteren Schichten der ZMI-Optionen anzeigt. Das Plone-Control Panel wird kontinuierlich weiterentwickelt, um eine benutzerfreundlichere Schnittstelle zu den Funktionen im ZMI zu bieten. Das da Projekt sehr aktiv ist, kann man nur schwer voraussagen, welche Funktionalität in Zukunft vorhanden sein wird. Daher empfehle ich, dass Sie einfach ins Control Panel gehen und sich anschauen, welche Funktionen momentan verfügbar sind. Wenn Sie Ihre Aufgabe dort nicht lösen können, müssen Sie ins ZMI gehen.

Um auf das Control Panel zuzugreifen, melden Sie sich bei Plone als Benutzer mit Manager-Rechten an. Wenn Sie keinen solchen Benutzer haben, aber selbst Site-Administrator sind, sollten Sie schnell in Kapitel 9 nachsehen, wie Sie das machen. Wenn Sie kein Site-Administrator sind, aber einen solchen Zugriff haben möchten, sollten Sie Ihren Site-Administrator darum bitten. Um zum Control Panel zu gelangen, klicken Sie auf PLONE KONFIGURATION oben auf der Seite (siehe Abbildung 4.1).

Abbildung 4.1: Zugriff auf das Control Panel

Danach wird das Control Panel geöffnet (siehe Abbildung 4.2).

Im Control Panel sind folgende Funktionen verfügbar:

- **Produkte hinzufügen/löschen**: Mit einem Klick auf diesen Link können Sie die Installation von Produkten automatisieren (das wird detailliert in Kapitel 10 behandelt).
- **Fehlerprotokoll**: Mit diesem Link gelangen Sie zum Fehlerprotokoll der Plone-Site.
- **E-Mail Einstellungen**: Hiermit können Sie den SMTP-Server (Simple Mail Transfer Protocol) ändern, mit dem Plone E-Mails verschickt.

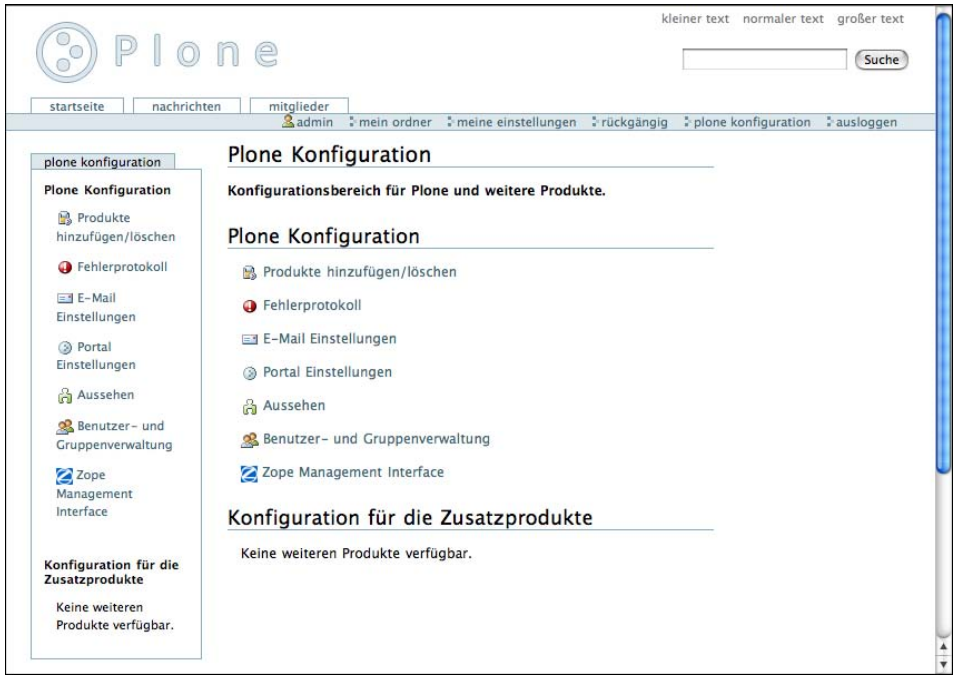


Abbildung 4.2: Das Plone-Control Panel

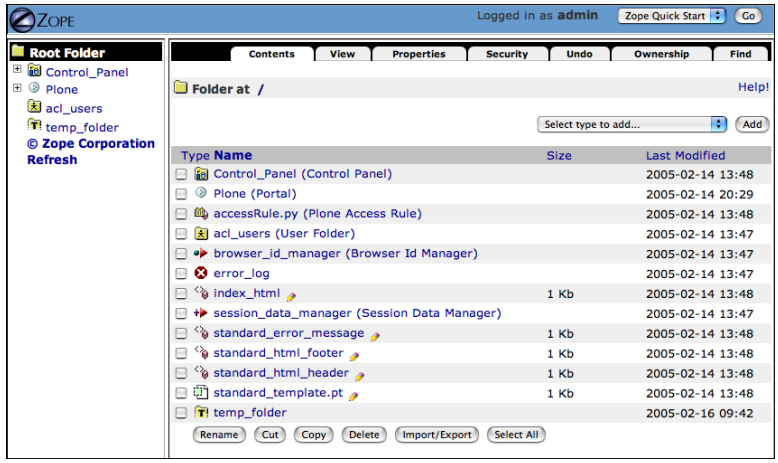
- **Portal Einstellungen**: Hiermit können Sie Portal-Einstellungen ändern (siehe den Abschnitt »Titel, Beschreibung und E-Mail-Adressen ändern« in diesem Kapitel).
- **Aussehen**: Hiermit können Sie die aktuelle Skin (in der deutschen Lokalisierung auch Aussehen genannt) ändern (siehe Kapitel 7).
- **Benutzer- und Gruppenverwaltung**: Hiermit können Sie Benutzer und Gruppen verändern (siehe Kapitel 8).
- **Zope Management Interface**: Mit diesem Link gelangen Sie zum ZMI.

Im weiteren Verlauf dieses Buchs beziehe ich mich auf das Plone-Control Panel, wann immer die jeweilige Eigenschaft dort verfügbar ist. Ansonsten wird das ZMI verwendet, um Eigenschaften zu verändern.

Exkurs: Das ZMI verwenden

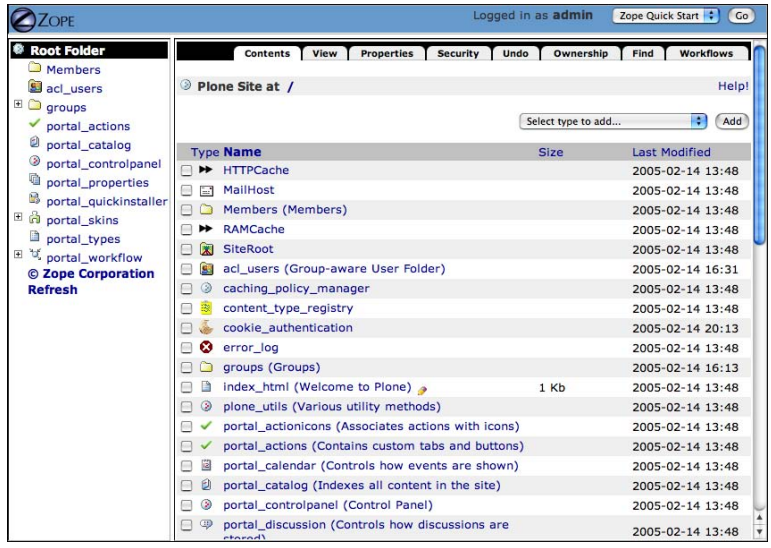
Das ZMI ist die grundlegende Schnittstelle, mit der Sie Zugriff auf die darunter liegende Zope-Schnittstelle von Plone haben. Vor Plone war das ZMI die wichtigste Methode, um auf eine Zope-Site zuzugreifen und sie und ihren Inhalt zu bearbeiten und zu verwalten. Das ZMI war ursprünglich die Webschnittstelle für das Content-Management-System. Natürlich ist Zope kein solches Fertig-CMS wie Plone, sondern eine Anwendung, die unter einem System wie Plone läuft. Nach einigem Herumspielen mit dem ZMI werden Sie sehen, warum es als Schnittstelle zu einem CMS ungeeignet ist.

Eines bietet das ZMI aber doch, nämlich eine Schnittstelle zur darunter liegenden Infrastruktur von Plone und Zope. Viele der in diesem Kapitel genannten grundlegenden Eigenschaften finden Sie über Plone, aber irgendwann müssen Sie das ZMI doch benutzen. Wenn Sie noch nicht ins ZMI gegangen sind, werden Sie sehen, dass es einige einfache Wege dorthin gibt. Der einfachste ist, sich als Benutzer mit Manager-Rechten anzumelden und auf PLONE KONFIGURATION und dann auf ZOPE MANAGEMENT INTERFACE zu klicken. Sie werden feststellen, dass die Adresse des ZMI der URL Ihrer Plone-Site mit einem */manage* am Ende ist. Das ZMI für Ihre Plone-Site sollte etwa wie folgt aussehen:



Eventuell haben Sie ein Problem mit dem Virtual Hosting, was bei den Windows- und Mac-Installationsprogrammen vorkommen soll. *Virtual Hosting* bezeichnet die Möglichkeit, die Plone-Site und nicht die Wurzel Ihrer Zope-Instanz als Wurzelobjekt zu benutzen. Weitere Angaben zum Virtual Hosting finden Sie in Kapitel 10. Um also zur Wurzel zu gelangen, müssen Sie auf den Management-Port zugreifen. Unter Windows wählen Sie START – PLONE – PLONE – MANAGE ROOT. Sie werden feststellen, dass die Adresse der Site auf *http://localhost:8080/manage* eingestellt wird. Weitergehende Informationen zum Virtual Hosting bei Ihrer Installation finden Sie in der entsprechenden Dokumentation.

Sie müssen aus zwei Gründen zur Wurzel Ihrer Zope-Installation gelangen können: Erstens müssen Sie zum Zope-Control Panel kommen können, und zweitens müssen Sie zur Wurzel Ihre Plone-Site gelangen können, um Plone-Sites zu erstellen, umzubenennen und zu kopieren. Im Zope-Control Panel erhalten Sie Zugriff auf Datenbankinformationen und auf Produkte und andere Add-Ons (diesen Zugriff benötigen Sie in Kapitel 10), wie Sie hier sehen können:



Tipp

Bei der Arbeit mit dem ZMI finde ich es sehr hilfreich, zwei verschiedene Browser gleichzeitig geöffnet zu haben. Ich zum Beispiel verwende Mozilla und Firefox. Nebenbei bemerkt empfiehlt es für Site-Administratoren immer, zwei verschiedene Browser zu haben, damit man testen kann, ob Änderungen in mehr als einem Browser funktionieren.

4.1.1 Titel, Beschreibung und E-Mail-Adressen ändern

Titel, Beschreibung und E-Mail-Adressen werden als Objekteigenschaften in einer Plone-Site gespeichert. Auf diese Felder können Sie zugreifen, indem Sie im Plone-Control Panel auf PORTAL EINSTELLUNGEN klicken (siehe Abbildung 4.3).

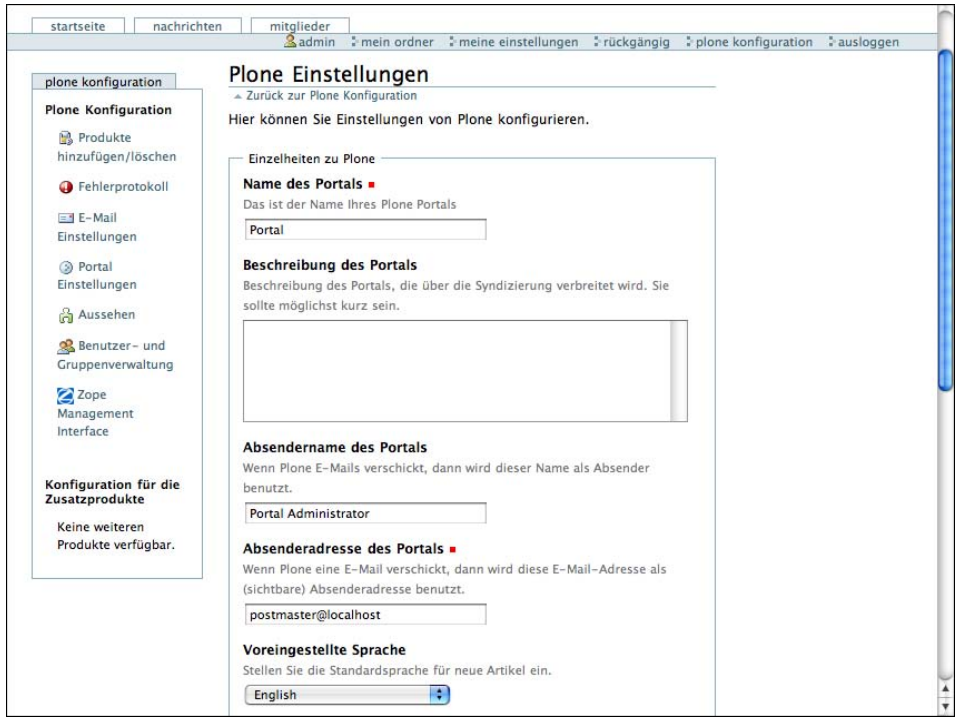


Abbildung 4.3: Portal-Einstellungen

Folgende Portal-Einstellungen sind vorhanden:

- **Name des Portals**: Dies ist der Name der Site, der im Titel von Browsern, der Pfadnavigation, der Navigation, E-Mails usw. erscheint. Der voreingestellte Wert lautet Portal.
- **Beschreibung des Portals**: Dies ist die Beschreibung des Portals, die im Moment nur bei der Syndizierung verwendet wird.
- **Absendername des Portals**: Dieses Feld wird in mehreren Zusammenhängen verwendet, z.B. bei einem verlorenen Passwort oder bei der Funktion »Einem-Freund-empfehlen«. Plone verschickt E-Mails unter diesem Namen, dessen Vorgabewert PORTAL ADMINISTRATOR lautet.
- **Absenderadresse des Portals**: Unter dieser Adresse verschickt Plone seine E-Mails. Die Voreinstellung lautet POSTMASTER@LOCALHOST.
- **Voreingestellte Sprache**: Dies ist die Standardsprache, die in den Eigenschaften eines Objekts verwendet wird.
- **Passworteigenschaften**: Neue Benutzer haben zwei Möglichkeiten: Entweder wählen sie ein Passwort selbst oder sie bekommen eines per E-Mail geschickt. Zwar müssen sie in beiden Fällen ein Passwort angeben, aber im zweiten Fall ist es schwerer, Scheinkonten einzurichten.
- **Externe Editoren ermöglichen**: Dies erlaubt es, externe Editoren, also ein fortgeschrittenes Bearbeitungswerkzeug, zu benutzen. Dazu muss das Produkt External Editor auf dem Rechner des Benutzers installiert sein. Kapitel 10 behandelt dies im Detail.

Nach der Auswahl der gewünschten Optionen klicken Sie auf SPEICHERN, damit die Änderungen sofort wirksam werden.

4.1.2 Einen Mail-Server einrichten

Plone verschickt E-Mails mit Hilfe des Objekts *MailHost*, das eine Schnittstelle zum SMTP-Server bietet und es dem Entwickler ermöglicht, Formulare und Werkzeuge zu schreiben, mit denen E-Mails verschickt werden können. Diese Einstellung wird bei der Funktion »Einem-Freund-empfehlen« und zum Verschicken eines vergessenen Passworts verwendet.

Die Standardkonfiguration ist ein Mailserver auf dem lokalen Rechner auf Port 25. Falls der SMTP-Server sich anderswo im Netzwerk befindet, können Sie auf das Formular zugreifen, indem Sie auf PLONE KONFIGURATION und E-MAIL-EINSTELLUNGEN klicken und dann den Server und den Port entsprechend ändern. In meinem Netzwerk befindet sich der Mailserver auf *monty.clearwind.ca* auf Port 1025, d.h., ich stelle den Server ein, wie es in Abbildung 4.4 zu sehen ist. In den

meisten Fällen jedoch müssen Sie hier nichts ändern (solange man den Server nicht unter Windows betreibt, da hier standardmäßig kein SMTP-Server vorausgesetzt werden kann).

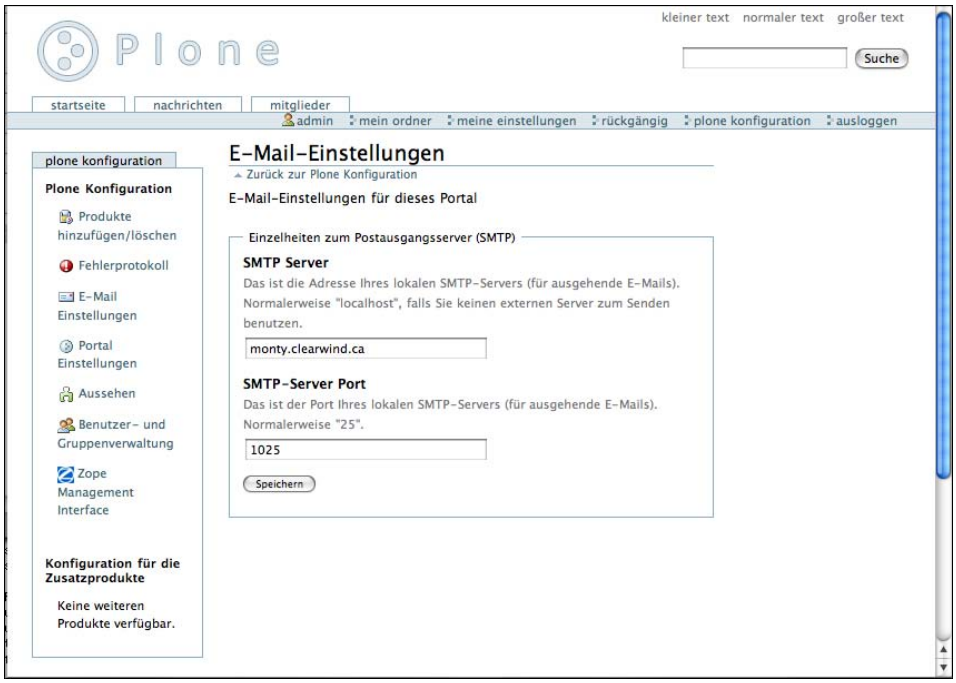


Abbildung 4.4: Einrichten des Mailservers

Hinweis

Das Objekt `MailHost` ist ein Zope-Objekt, auf das man mit dem ZMI zugreifen kann. Dieses Objekt kann momentan noch keine Identifizierung mit dem Server vornehmen. Wenn das nötig sein sollte, müssen Sie die Einstellungen auf dem Server ändern.

4.1.3 Fehlermeldungen protokollieren

Im Fehlerprotokoll werden Fehler festgehalten, die in einer Plone-Site möglicherweise auftreten. Dazu gehören Fehler wie Page Not Found (404), Autorisierungsfehler usw. Es dient nicht dazu, Fehler in Formularen abzufangen. Wenn z.B. jemand in einem Feld keinen Wert eingibt, wo einer eingegeben werden muss, so wird das hier nicht festgehalten. Das ist kein Fehler, denn es wird vom Validie-

rungs-Framework abgefangen. Dieses Fehlerprotokoll soll nur mögliche interne Server-Fehler abfangen.

Klicken Sie in der Plone-Schnittstelle auf PLONE KONFIGURATION und dann auf FEHLERPROTOKOLL, um die Fehler auf der Plone-Site zu sehen. Klicken Sie auf den Fehler in der Liste (sofern es eine gibt), um den Fehler zu sehen. Abbildung 4.5 zeigt einen Fehler, der beim fehlerhaften Ausfüllen des Formulars für die E-Mail-Einstellungen auftrat. Es ist eine lange Seite, die einen kompletten Python-Traceback sowie die Eingangsabfrage enthält.

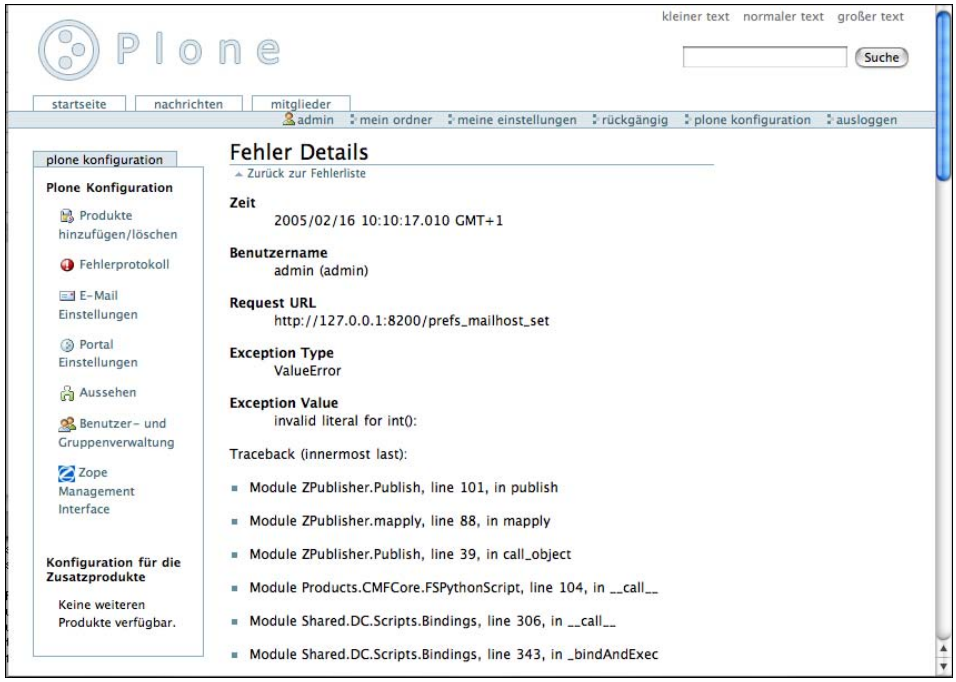


Abbildung 4.5: Ein Fehlerbeispiel

Im Fehlerlisten-Formular sehen Sie folgende Einstellungen:

- **Anzahl der Fehler, die gespeichert werden**: Dies ist die Anzahl der auf dem Bildschirm anzuzeigenden Fehler. Der Standardwert dafür beträgt 20.
- **Fehler ins Event-Logfile kopieren**: Hiermit werden alle Fehler in eine Fehlerprotokolldatei kopiert. Ohne diese Kopie wird nichts dauerhaft über diese Fehler festgehalten. Standardmäßig ist dies ausgewählt.
- **Fehlertypen ignoriert**: Dies ist eine Liste von Fehlertypen, die ignoriert werden sollen (einer pro Zeile). Voreingestellt sind *Unauthorized*, *NotFound* und *Redirect*.

Jeden Fehler können Sie protokollieren und auf dem Schirm anzeigen. Das bedeutet: Wenn ein Benutzer Ihre Site besucht und es tritt ein Fehler auf, dann können Sie im Fehlerprotokoll nachsehen, was passiert ist. Die Fehler bestehen aus den drei Komponenten Fehlertyp, Fehlerwert (der String, der erklärt, wann ein Fehler auftritt) und Traceback. Die beiden ersten werden dem Benutzer auf der Standardfehlerseite angezeigt (siehe Abbildung 4.6).

Abbildung 4.6: Ein Beispiel für eine Fehlermeldung

Wenn ein Benutzer einen Fehler meldet, enthält der Bericht oft eine Nachricht mit dem Fehlernamen und einem Fehlerwert darin. Wenn der Benutzer nichts tun darf und ein Unauthorized-Fehler oder ein Page Not Found (404) auftritt, dann bekommen Sie eine speziell angepasste Fehlerseite statt einer Standardseite angezeigt (siehe Abbildung 4.6). Folgende Standardfehlertypen kommen vor:

- **Unauthorized**: Dieser Fehler tritt ein, wenn ein Benutzer kein Recht hat, eine Funktion auszuführen.
- **NotFound**: Dieser Fehler tritt ein, wenn das Element, auf das ein Benutzer zugreifen möchte, nicht existiert.
- **Redirect**: Dies ist ein Fehler, der eine HTTP-Weiterleitung (Hypertext Transfer Protocol) auslösen kann.
- **AttributeError**: Dieser Fehler wird ausgelöst, wenn ein Objekt ein Attribut nicht besitzt.
- **ValueError**: Dieser Fehler tritt auf, wenn ein gegebener Wert nicht korrekt ist und vom Validierungs- oder einem anderen Framework nicht abgefangen wird.

4.2 Das Look-and-Feel von Plone anpassen

Die folgenden Abschnitte beschreiben weitere mögliche Anpassungen. Fast alle davon erfordern den Zugriff auf das ZMI.

4.2.1 Mehr über Portlets

Auf einer Plone-Site sehen Sie standardmäßig drei Spalten, links, in der Mitte und rechts. Die mittlere Spalte enthält den Inhalt des gerade angezeigten Objekts. Hier befindet sich die meiste Benutzerfunktionalität für das Hinzufügen, Bearbeiten, Formularerstellen usw. Die beiden Spalten links und rechts enthalten eine Reihe von Kästen, die Informationen anzeigen. Jeder davon ist ein so genanntes *Portlet*. Eine Variable bestimmt, welche Portlets zu einem gewissen Zeitpunkt angezeigt werden. Am besten versteht man diese Portlets, wenn man sich die Standard-Portlets anschaut, die auf einer Plone-Site vorhanden sind. Die Parameter für die Portlets finden Sie im Portal-Objekt. Darauf können Sie zugreifen, wenn Sie ins ZMI gehen und sicherstellen, dass Sie in der Plone-Wurzel-Site sind, und auf den PROPERTIES-Reiter klicken. Dann wird eine Liste von Eigenschaften geöffnet, darunter auch *left_slots*, *right_slots* und *document_action_slots* (siehe Abbildung 4.7).

Hinweis

In früheren Versionen von Plone wurden Portlets auch *Slots* genannt. Das ist jedoch ein häufig gebrauchter Begriff, der auch bei Seiten-Templates verwendet wird. Deswegen wurde er in Version 2 in *Portlets* geändert. An manchen Stellen im Code oder im Text wird vielleicht noch *Slots* verwendet. An diesen Stellen bedeutet der Begriff *Slots* dasselbe wie *Portlets*.

Die *left_slots*- bzw. *right_slots*-Eigenschaften gelten für Portlets jeweils im linken bzw. rechten Teil der Seite. Die Portlets werden von oben nach unten in der Reihenfolge angezeigt, in der sie in diesen Eigenschaften aufgelistet werden. Allerdings verfügen die meisten Portlets über Code, der sicherstellt, dass sie nur dann angezeigt werden, wenn es Sinn macht. Ein Login-Portlet macht z.B. keinen Sinn, wenn der Benutzer schon angemeldet ist. In dem Fall ist das Login-Portlet zwar in der Portlet-Liste enthalten, wird aber nur bei Bedarf angezeigt.

Alle Portlet-Werte sind tatsächlich spezielle Werte, nämlich TALES-Pfadausdrücke (Template Attribute Languages Expression Syntax), die in Kapitel 5 detailliert behandelt werden. Site-Entwickler können eigene Portlets zu einer Site hinzufügen, indem sie einfache Makros und Seiten-Templates erstellen. Folgende Standard-Portlets gibt es:

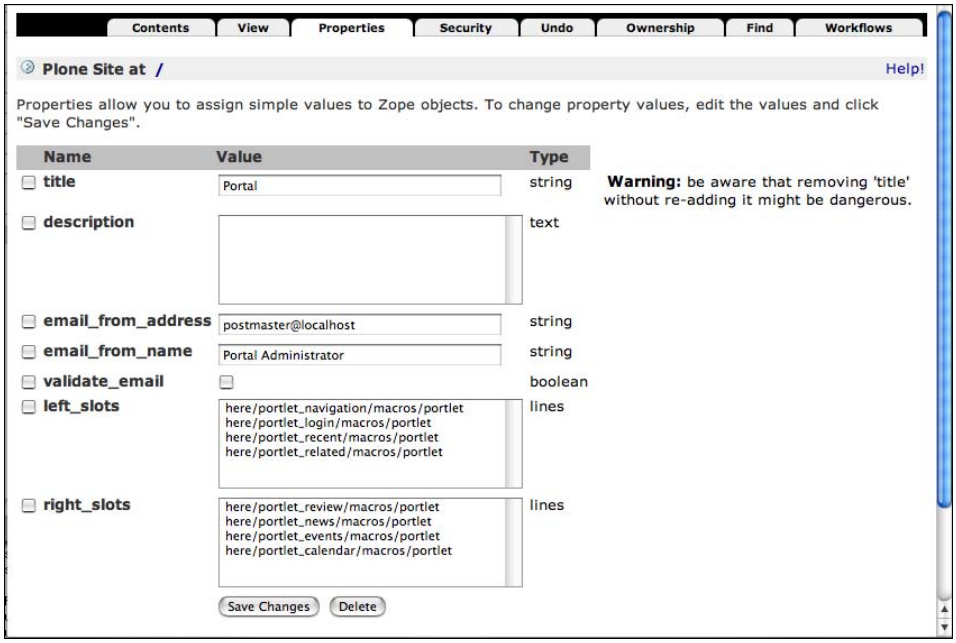


Abbildung 4.7: Standardeigenschaften von Portlets

- **left_slots**: Dazu gehören Portlets zur Navigation, zum Login und ähnliche.
- **right_slots**: Dazu gehören Portlets wie die Revisionsliste, Nachrichten, Termine, Aktuelle Artikel und der Kalender. Alle verfügbaren Portlets werden nicht standardmäßig in Plone konfiguriert. Die folgenden Abschnitte beschreiben Portlet-Slots in Plone. Jeder Abschnitt beschreibt ein Portlet und zeigt, wie es aussieht. Dann gebe ich den Pfadausdruck an, den Sie brauchen, um es zur *slots*-Eigenschaft hinzuzufügen, so dass es in Ihrer Plone-Site erscheint.

Um etwa links das Kalender-Portlet anzuzeigen, geben Sie in der *left_slots*-Eigenschaft **here/portlet_calendar/macros/portlet** ein und klicken auf SAVE CHANGES. Wenn Sie es aus der *right_slots*-Eigenschaft entfernen möchten, können Sie die gleiche Zeile aus der *right_slots*-Eigenschaft entfernen und wieder auf SAVE CHANGES klicken.

Kalender

Das Kalender-Portlet ist eines der Standard-Portlets. Es zeigt rechts auf einer Plone-Seite einen Kalender an. Dieses Portlet zeigt veröffentlichte Termine für den jeweiligen Monat in einem kleinen Kalender an. Das Kalender-Portlet erscheint auch dann, wenn es keine Termine gibt. Mit dem Werkzeug `portal_calendar` können Sie den Kalender im ZMI weiter konfigurieren (siehe Abbildung 4.8).

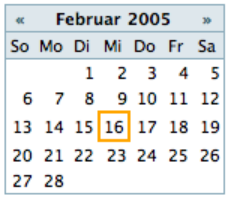


Abbildung 4.8: Kalender-Portlet

Der Ausdruck, der hinzugefügt werden muss, lautet *here/portlet_calendar/macros/portlet*.

Termine

Das Termine-Portlet zeigt eine Liste veröffentlichter kommender Termine an. Wenn Sie diesen Eintrag in der Portlet-Liste haben, wird das Portlet nur dann angezeigt, wenn es solche Termine gibt (siehe Abbildung 4.9).

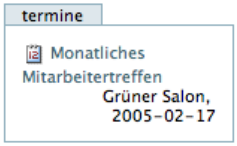


Abbildung 4.9: Termine-Portlet

Der Ausdruck, der hinzugefügt werden muss, lautet *here/portlet_events/macros/portlet*.

Favoriten

In der oberen linken Ecke eines Plone-Dokuments sehen Sie ein Plone-Icon. Benutzer können darauf klicken, um einen Favoriten hinzuzufügen. Ein *Favorit* ähnelt einem Bookmark oder Link auf die Seite, die Sie aufrufen möchten. Dieser Favorit wird jedoch auf der Plone-Site gespeichert. Abbildung 4.10 zeigt das Icon, mit dem ein Favorit hinzugefügt wird.

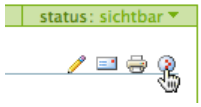


Abbildung 4.10: Das Icon zum Erstellen eines Favoriten

Favoriten werden zum Benutzerordner eines Benutzers hinzugefügt und werden im Favoriten-Portlet angezeigt, zusammen mit einem Link, der dazu dient, sie zu organisieren (siehe Abbildung 4.11). Die gezeigten Favoriten sind solche, die der Benutzer gespeichert hat. Das heißt, selbst wenn Sie diesen Eintrag in Ihrer Portlet-Liste haben, wird das Portlet nur dann angezeigt, wenn Sie auch wirklich Favoriten haben.

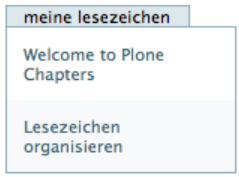


Abbildung 4.11: Favoriten-Portlet

Der Ausdruck, der hinzugefügt werden muss, lautet *here/portlet_favorites/macros/portlet*.

Anmeldung

Das Einloggen-Portlet zeigt das Anmeldeformular an, über das sich ein Benutzer mit seinem Benutzernamen und Passwort anmelden kann. Wenn er sein Passwort vergessen hat, hat er die Möglichkeit, es sich per E-Mail schicken zu lassen. Auch wenn dieses Portlet in der Portlet-Liste ist, wird es nur dann angezeigt, falls der Benutzer noch nicht angemeldet ist (siehe Abbildung 4.12).

Abbildung 4.12: Einloggen-Portlet

Der Ausdruck, der hinzugefügt werden muss, lautet *here/portlet_login/macros/portlet*.

Navigation

Das Navigation-Portlet zeigt einen einfachen Baum aller Ordner an der aktuellen Position in Baumform an. Es bietet ein mächtiges und einfaches Navigationswerkzeug. Das Navigation-Portlet ist extrem anpassbar. Sie können es mit einem Klick auf PORTAL_PROPERTIES und NAVTREE_PROPERTIES im ZMI ändern, was im Abschnitt »Das Navigation-Portlet ändern« weiter unten behandelt wird (siehe Abbildung 4.13).

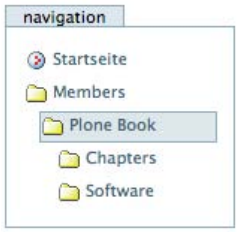


Abbildung 4.13: Navigation-Portlet

Der Ausdruck, der hinzugefügt werden muss, lautet *here/portlet_navigation/macros/portlet*.

Nachrichten

Das Nachrichten-Portlet zeigt die Liste der letzten Nachrichten mit Links darauf an (siehe Abbildung 4.14). Auch dann, wenn Sie dieses Portlet in der Portlet-Liste haben, wird es nur dann angezeigt, wenn irgendwelche Nachrichten veröffentlicht wurden. Die Nachrichten einer Site sind auch über einen Klick im NACHRICHTEN-Reiter verfügbar.

Abbildung 4.14: Nachrichten-Portlet

Der Ausdruck, der hinzugefügt werden muss, lautet *here/portlet_news/macros/portlet*.

Aktuelle Artikel

Das Aktuelle Artikel-Portlet listet die zuletzt auf der Site veröffentlichten Artikel seit dem Zeitpunkt auf, an dem Sie zum letzten Mal angemeldet waren (siehe Abbildung 4.15). Auch wenn es keine solchen Artikel gibt, wird es angezeigt.

Abbildung 4.15: Aktuelle Artikel-Portlet

Der Ausdruck, der hinzugefügt werden muss, lautet *here/portlet_recent/macros/portlet*.

Dazu passend

Das Dazu-passend-Portlet zeigt eine Liste von Artikeln an, die zu dem Artikel passen, den Sie gerade sehen, was anhand der Stichwörter dazu bestimmt wird. Wenn ein passender Artikel ein Link auf eine andere Website ist, wird er in einer eigenen Liste externer Ressourcen angezeigt. Auch dann, wenn dieses Portlet in der Portlet-Liste ist, wird es nur dann angezeigt, wenn es passende Artikel gibt (siehe Abbildung 4.16).

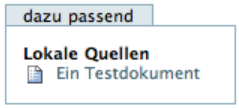


Abbildung 4.16: Dazu passend-Portlet

Der Ausdruck, der hinzugefügt werden muss, lautet *here/portlet_related/macros/portlet*.

Revisionsliste

Das Revisionsliste-Portlet zeigt eine Liste von Artikeln an, die im Zustand *Überprüfen* sind und darauf warten, geprüft zu werden. Es wird nur dann angezeigt, wenn der angemeldete Benutzer die Rolle des Redakteurs hat und es zu prüfende Artikel gibt.

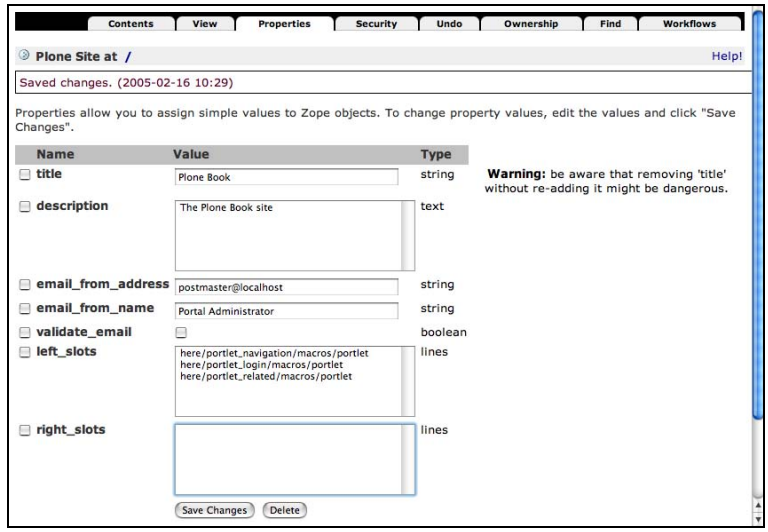
Abbildung 4.17: Revisionsliste-Portlet

Der Ausdruck, der hinzugefügt werden muss, lautet *here/portlet_review/macros/portlet*.

Exkurs: Buch-Website: Slots ändern

Für die Website zum Buch machen die meisten der Slots auf der rechten Seite keinen Sinn. Zu diesem Buch gibt es keine Termine, d.h., die Kalender- und Termine-Slots rechts sind nicht notwendig. Ich rechne zwar mit neuen Dingen, die noch zur Site hinzugefügt werden, aber es werden sehr wenige sein, sobald die Site einmal fertig ist. Also habe ich mich entschlossen, alle rechten Slots auf meiner Site zu entfernen. Das habe ich so gemacht, dass ich im ZMI zum Portal-Wurzelobjekt gegangen bin und auf das *Eigenschaften*-Objekt geklickt habe. Dann habe ich alle Inhalte in der Eigenschaft *right_slots* gelöscht. Die anderen Portlets, Navigation, Login und Dazu passend, die normalerweise links vorkommen, sind für mich alle von Nutzen, daher habe ich sie behalten.

So sehen im Moment die Portlet-Eigenschaften der Plone-Buchsite aus:



Verschiedene Portlets in verschiedenen Teilen Ihrer Site

Die Zope-Datenbank, auf der Plone basiert, verfügt über eine Eigenschaft namens *Akquisition*. In der einfachsten Form bedeutet das, dass bei der Suche nach einem Objekt wie *right_slots* Plone das nächstgelegene Objekt findet, das diese Eigenschaft enthält. Das heißt, dass Plone bei der Suche nach Portlets, die in der rechten Spalte angezeigt werden sollen, normalerweise das Wurzelobjekt findet und diese Portlets anzeigt.

Auf diese Weise können Sie die Eigenschaften im Portal-Wurzelobjekt ändern, um die ganze Site zu ändern. Sie werden vielleicht bemerken, dass es keinen Kalender gibt, wenn Sie auf den Link MEIN ORDNER klicken und zu Ihrem persönlichen Ordner gehen. Wenn Sie im ZMI auf MEMBERS und dann auf PROPERTIES klicken, werden Sie einen Eintrag für *right_slots* sehen. Für diesen Ordner besteht er aus einer leeren Liste. Wenn die Plone-Site nach einem Wert für die rechts anzuzeigenden Portlets sucht, bewegt sie sich in der Ordnerhierarchie so lange nach oben, bis sie den Ordner `Members` erreicht. Dort findet sie den Wert *right_slots* und benutzt ihn. Da der Wert von *right_slot* im Ordner `Members` leer ist, wenn Sie Inhalte aus `Members` sehen, wird der rechte Slot leer sein.

Durch das Hinzufügen und Entfernen von Ordnereigenschaften mit Hilfe des ZMI können Site-Administratoren genau anpassen, welche Portlets auf ihren Sites erscheinen. Zum Glück ist der PROPERTIES-Reiter einigermaßen einfach. Sie wählen einfach den Artikel im ZMI und klicken dann auf PROPERTIES. Um eine Eigenschaft zu den linken oder rechten Slots hinzuzufügen, benutzen Sie das untere ADD-Formular und vergewissern sich, dass der Eigenschaftstyp *lines* lautet.

4.2.2 Das Navigation-Portlet ändern

Von allen behandelten Portlets ist das Navigation-Portlet das wahrscheinlich nützlichste und am meisten nachgefragte. Wie können Sie also speziell dieses Navigation-Portlet und die Art, wie es angezeigt wird, ändern? Das Navigation-Portlet ist eine Liste von aktuellen Ordnern und Dokumenten im Navigation-Slot. Den Navigation-Slot können Sie dadurch ändern, dass Sie den Code ändern, allerdings können Sie viele Änderungen auch im ZMI vornehmen. Am Wichtigsten ist, sich daran zu erinneren, dass nur veröffentlichte Objekte im Navigationsbaum von Mitgliedern und anonymen Benutzern gesehen werden. Um die Eigenschaften des Navigationsbaums zu ändern, klicken Sie auf PORTAL_PROPERTIES und dann auf NAVTREE_PROPERTIES im ZMI.

Die folgende Liste ist ein Ausschnitt der verfügbaren Optionen:

- **showMyUserFolderOnly**: Dies zeigt nur den Benutzerordner des angemeldeten Benutzers an. Wenn der `Mitglieder`-Ordner in der Navigation angezeigt

wird, werden nicht alle `Mitglieder`-Ordner angezeigt. Dies ist standardmäßig ausgewählt.

- **showFolderishSiblingsOnly**: Wenn diese Option gewählt ist, werden nur Ordner in übergeordneten Ordnern angezeigt, sonst wird der ganze Inhalt angezeigt. Dies ist standardmäßig aus gewählt.
- **showFolderishChildrenOnly**: Wenn diese Option bei einem Ordner gewählt ist, werden nur Ordner im gleichen Ordner angezeigt statt alle anderen Inhaltstypen auch. Wird diese Option nicht gewählt, dann werden alle Inhalte des gerade angezeigten Ordners gezeigt. Diese Option ist standardmäßig aus gewählt.
- **roleSeeUnpublishedContent**: Wie bereits erwähnt wurde, wird ein Inhalt nur dann angezeigt, wenn er für Mitglieder und anonyme Benutzer veröffentlicht ist. Um nicht veröffentlichte Inhalte anzuzeigen, müssen Sie weitere Rollen zu dieser Liste hinzufügen, jeweils eine Rolle pro Zeile. Falls der Benutzer nicht sowieso schon Zugriff darauf hat, ist das nicht wünschenswert.
- **croppingLength**: Diese Option bestimmt, wie viele Zeichen der Namen im Navigationsbaum angezeigt werden. Der Standardwert hierfür ist 256.
- **idsnotToList**: Dies ist eine Liste der Artikel-IDs, die nicht angezeigt werden sollen. Setzen Sie jede ID in jeweils eine eigene Zeile. Standardmäßig ist diese Liste leer.

Exkurs: Buch-Website: Navigationsbaum ändern

Bei den meisten Websites bevorzuge ich einen umfangreicheren Navigationsbaum als den standardmäßig verfügbaren. Also bin ich zu den Optionen für den Navigationsbaum gegangen und habe *showFolderishChildrenOnly* und *showFolderishSiblingsOnly* ausgeschaltet. Dadurch wurden die Inhalte hübsch angezeigt. Hier ist z.B. der Software-Ordner mit nur einigen gewählen Artikeln:

Nehmen Sie Änderungen an dieser Liste vor, und klicken Sie dann auf SAVE CHANGES. Die Reihenfolge der Artikel im Navigationsbaum wird von ihrer Reihenfolge im Ordnerinhalt-Formular bestimmt. Wie Sie in Kapitel 3 gesehen haben, können die Benutzer diese Reihenfolge nach Belieben mit den Pfeilen nach oben bzw. unten ändern.

4.2.3 Datumsformate ändern

In allen Portlets und auf der gesamten Site stellt Plone Datumsangaben konsistent in einem Format dar, das intern bearbeitet werden kann. Immer, wenn in Plone ein Datum angezeigt wird, wird eines von zwei Formaten aufgerufen. Diese Formate finden Sie im ZMI, indem Sie auf PORTAL_PROPERTIES und SITE_PROPERTIES klicken. Diese Formate lauten:

- **localTimeFormat**: Dies ist das Zeit-Format für Datumsangaben, die in Plone in einem Kurzformat erscheinen sollen.
- **localLongTimeFormat**: Dies ist das Zeit-Format für Datumsangaben, die in einem Langformat mit Sekundenangaben erscheinen sollen.

Das Datumsformat basiert auf dem time-Modul von Python. Die Referenz zu den Formaten finden Sie unter *http://www.python.org/doc/current/lib/module-time.html*. Beim Kurzformat lautet der Standardwert `%Y-%m-%d`, d.h. `Jahr-Monat-Tag` als Dezimalzahlen (z.B. `2003-10-26`). Beim Langformat lautet der Standardwert `%Y-%m-%d %I:%M %p`, d.h. `Jahr-Monat-Tag Stunden:Minuten am/pm` (z.B `2003-10-26 07:32 PM`).

Hier ist eine kurze Zusammenfassung der verfügbaren Optionen:

- `%a`: Abgekürzter Wochentagsname der aktuellen Lokalisierung (z.B. `Mon`)
- `%A`: Voller Wochentagsname der aktuellen Lokalisierung (z.B. `Montag`)
- `%b`: Abgekürzter Monatsname der aktuellen Lokalisierung (z.B. `Jan`)
- `%B`: Voller Monatsname der aktuellen Lokalisierung (z.B. `Januar`)
- `%d`: Tag des Monats als Dezimalzahl
- `%H`: Stunde (max. 24) als Dezimalzahl
- `%I`: Stunde (max. 12) als Dezimalzahl
- `%m`: Monat als Dezimalzahl
- `%M`: Minute als Dezimalzahl
- `%S`: Sekunde als Dezimalzahl
- `%y`: Jahr ohne Jahrhundert als Dezimalzahl
- `%Y`: Jahr mit Jahrhundert als Dezimalzahl

Wenn der Wochentagsname im Kurzformat enthalten sein soll, können Sie das einfach dadurch erreichen, dass Sie das Kurzformat auf `%A, %b. %d, %y` ändern. Dadurch entsteht z.B. `Donnerstag, Okt. 24, 02`. Diese Datumsangaben werden in den Kästen links und rechts auf dem Bildschirm, in Suchergebnissen, in den Verfasserangaben usw. verwendet.

4.2.4 Stichwörter und Terminarten hinzufügen

Mit einem der Werkzeuge in Plone, *portal_metadata*, kann der Site-Administrator einige Metadaten-Elemente definieren. Plone verwendet die mit *portal_metadata* definierten Metadaten an verschiedenen Stellen.

Wenn Sie z.B. einen Termin hinzufügen, erhalten Sie eine Liste möglicher Terminarten. Diese Liste können Sie erweitern, wenn Sie im ZMI auf *portal_metadata* klicken und dann auf *elements* und *subject*. Dann sehen Sie eine Wortliste (Vocabulary) für Termine mit den Themen für diesen Inhaltstyp. Diese Liste mit je einem Eintrag pro Zeile kann man ganz einfach erweitern oder bearbeiten, um die relevanten Terminarten zur Verfügung zu haben. Diese Terminarten erscheinen dann in den Formularen zum Hinzufügen und Bearbeiten von Terminen.

Mit *portal_metadata* kann man auch die auf einer Site verfügbaren Stichwörter angeben. In dem Formular unter *portal_metadata/elements/subject* sehen Sie auch ein Vocabulary-Formular für den Inhaltstyp <*default*>. Wenn Sie auf dieser Seite Einträge im VOCABULARY-Feld hinzufügen und auf UPDATE klicken, fügen Sie diese Einträge zur Liste der verfügbaren Stichwörter für *alle* Inhaltstypen hinzu.

Wenn Sie möchten, dass Stichwörter nur für, sagen wir, Dokumente erscheinen, dann benutzen Sie das Hinzufügen-Formular unten auf der Seite. Wählen Sie einen Inhaltstyp, und fügen Sie Wörter in die Vocabulary-Liste ein, und zwar eines pro Zeile. Daraus werden dann die einzigen Stichworte, die ein Benutzer für diesen Inhalt auswählen kann.

Wenn Sie als Benutzer mit Manager- oder Redakteursrechten angemeldet sind, erscheint, wenn Sie den EIGENSCHAFTEN-Reiter eines Objekts in Plone anklicken, ein Kasten namens NEUES STICHWORT, in dem neue Stichwörter spontan hinzugefügt werden können. Diese erscheinen nicht in der Vocabulary-Liste von *portal_metadata*, aber in allen Inhaltstypen, die andere Benutzer eingeben können.

4.2.5 Standardseite ändern

Wie in Kapitel 3 beschrieben wurde, erscheint die Standardseite eines Ordners, wenn ein Benutzer diesen Ordner anzeigt und diese Seite existiert. In älteren Versionen von Zope und Plone lautete der Name dieser Standardseite `index_html`. Diesen können Sie sehr oft auf Plone-Sites sehen, wo in Website-Adressen oft-

mals `index_html` am Ende steht. Wenn Sie daraus eine Dateinamenserweiterung machen, die weiter verbreitet ist und leichter erkannt wird, z.B. index.html, dann ist es auch einfacher, die Seite mit vorhandenen Editoren und Website-Werkzeugen zu bearbeiten.

In Plone können Sie eine Liste von Seiten definieren, nach denen gesucht wird, wenn eine Standardseite angezeigt werden soll (siehe Abbildung 4.18). Diese Standardseiten heißen `index_html`, `index.html`, `index.htm` und `FrontPage`. Diese Liste setzen Sie in `site_properties/portal_properties/default_page property`, mit je einem Namen pro Zeile. Bei der Suche nach der Standardseite sucht Plone nach jeder Seite aus dieser Liste, wobei es vorne anfängt und so lange weitersucht, bis es eine passende findet. Wenn Sie den Wert für nur einen Ordner ändern möchten, können Sie mit dem ZMI auf den Ordner zugreifen, den Reiter PROPERTIES anklicken und dann eine neue Listeneigenschaft namens *default_page* hinzufügen.

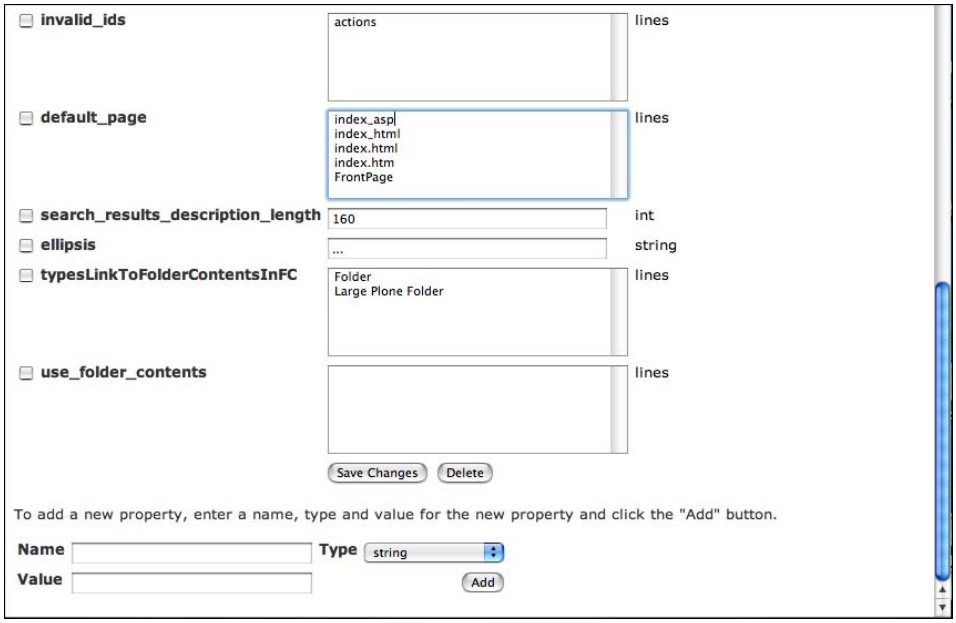


Abbildung 4.18: Einstellen von index.asp als erste Standardseite

Exkurs: Wie bekommen Sie in der Nachrichtenliste einen Eintrag auf die Standardseite?

Um zu begreifen, wie das genau funktioniert, muss man ziemlich viel von der darunter liegenden Maschinerie verstehen. Gehen Sie vorläufig einfach zu Ihrer Portal-Wurzel, und klicken Sie auf den Reiter PROPERTIES. Dann gehen Sie zum unteren Teil der Seite und füllen das Formular zum Hinzufügen neuer Eigenschaften mit folgenden Angaben aus. Klicken Sie anschließend auf den ADD-Button:

1. Im Name-Feld: **default_page**
2. Im Value-Feld: **news**
3. Im Type-Feld: **lines**

Kehren Sie nun zu Ihrer Plone-Site zurück. Statt der Standard-Homepage sehen Sie nun die Nachrichtenseite. Der NACHRICHTEN-Reiter zeigt ebenfalls weiterhin Nachrichten an, aber in den folgenden Abschnitten zeige ich Ihnen, wie Sie das abstellen.

4.2.6 Reiter der Site ändern

In einer Plone-Site beziehen sich verschiedene Reiter auf verschiedene Site-Abschnitte oder -Teile. Die Verwendung von Reitern ist ein sehr gebräuchliches Mittel beim Design von Websites und wird auch auf den Sites von Amazon, MSN und auf Plone-Sites praktiziert.

Es gibt zwei Arten von Reitern: *Portalreiter* und *Inhaltsreiter*. Portalreiter sind blau und erscheinen oben auf der Plone-Site. Die Standardreiter heißen STARTSEITE, NACHRICHTEN und MITGLIEDER. In den folgenden Abschnitten sehen Sie, wie Sie diese Reiter anpassen können. Inhaltsreiter sind grün und erscheinen, wenn ein Artikel bearbeitet werden kann. Wie ihr Name schon sagt, beziehen sich Inhaltsreiter auf den Inhalt. Kapitel 11 zeigt, wie man diese Reiter ändern kann. Die Reiter auf einer Plone-Site entstehen aus einer Reihe von Aktionen. Das heißt, Sie müssen einen kleinen allgemeinen Abstecher zu Aktionen machen, um zu verstehen, wie Sie diese Reiter verändern können.

Einführung in Aktionen

In Plone können gewisse Personen bestimmte Aufgaben zu verschiedenen Zeiten und an verschiedenen Orten der Site ausführen. Diese Aufgaben werden *Aktionen* genannt. Plone übersetzt sie in Reiter, Links und andere Elemente. Aktionen sind eine hochgradig konfigurierbare Art, um Navigationselemente für eine Site zu erstellen.

Jede Aktion verfügt über folgende Eigenschaften, die im ZMI konfiguriert werden können. Wo sie genau konfiguriert werden, hängt davon ab, wo die Aktion gespeichert ist. Hier ist eine Liste von Eigenschaften für eine Standardaktion:

Name: Dies ist ein benutzerfreundlicher Name für die Aktion. Er wird in der Benutzerschnittstelle oft verwendet. Wenn z.B. die Aktion als Reiter benutzt wird, entspricht dieser Wert dem Text im Reiter.

- **Id**: Dies ist eine eindeutige ID für die Aktion.
- **Actions**: Dies ist die auszuführende Aktion. Wenn die Aktion z.B. als Reiter definiert wird, wird diese Aktion beim späteren Klick auf diesen Reiter ausgeführt. Dieses Feld enthält einen TALES-Ausdruck (siehe weitere Informationen in Kapitel 5).
- **Condition**: Dies ist die Bedingung, die erfüllt sein muss, damit die Aktion ausgeführt wird. Wenn die Aktion z.B. als Reiter genutzt wird und diese Bedingung ist erfüllt, dann erscheint der Reiter. Dieses Feld enthält einen TALES-Ausdruck (siehe Kapitel 5).
- **Permission**: Dies sind die Rechte, die der Benutzer haben muss, damit er diese Aktion ausführen kann. Diese Rechte müssen gegeben sein, damit die Aktion genutzt werden kann (siehe Kapitel 9 für weitere Informationen zum Thema Sicherheit).
- **Category**: Hiermit werden die Aktionen kategorisiert. In Plone werden Aktionen damit voneinander unterschieden, d.h., sie werden in verschiedenen Abschnitten der Benutzerschnittstelle verwendet. Bei Portalreitern lautet der Kategoriewert *portal_tabs*.
- **Visible**: Das bestimmt, ob die Kategorie aktiv ist. Der Begriff *visible* wird verwendet, weil Aktionen normalerweise mit visuellen Elementen gekoppelt sind.

4.2.7 Einführung in die obersten Reiter

In den folgenden Abschnitten werden Sie als Beispiel die Portalreiter auf zwei verschiedene Weisen ändern. Sie werden den Reiter der Startseite so ändern, dass er *Willkommen* lautet, und Sie werden den MITGLIEDER-Reiter nach links neben den NACHRICHTEN-Reiter verlegen. Die Aktionen für die Portalreiter werden in dem Werkzeug `portal_actions` gespeichert. Um sie also zu ändern, klicken Sie im ZMI auf PORTAL_ACTIONS. Wie Sie in Abbildung 4.19 sehen, wird dabei eine lange Liste von standardmäßig vorhandenen Portalaktionen geöffnet. Manche dieser Aktionen werden Ihnen bekannt vorkommen, da sie für Teile der Plone-Site stehen.

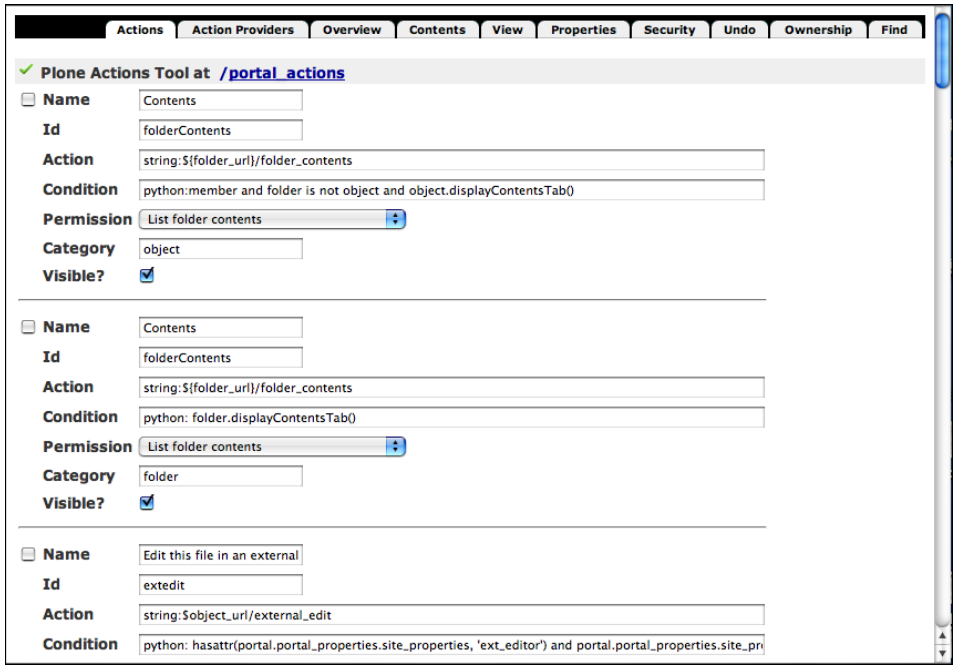


Abbildung 4.19: Die Portal-Aktionen für Ihre Plone-Site

Scrollen Sie durch diese Aktionen, bis Sie den Eintrag HOME finden, und ändern Sie das Namensfeld auf **Willkommen**. Scrollen Sie dann hinunter zum Seitenende, und klicken Sie auf SAVE. Zurück in der Plone-Schnittstelle werden Sie bemerken, dass der Reiter nun *Willkommen* heißt.

Die Reihenfolge der Reiter von links nach rechts auf der Seite wird durch die Reihenfolge von oben nach unten in der Liste der Aktionen bestimmt. Um einen Reiter zu verschieben, müssen Sie ihn in der Liste markieren und ans Seitenende zu den Buttons MOVE UP und MOVE DOWN scrollen. Es ist etwas umständlich, aber durch wiederholtes Markieren der Aktionen und Benutzen der Rauf- und Runter-Buttons können Sie die Reihenfolge ändern. Führen Sie das durch, und Sie werden bemerken, dass die Reiter nun in einer anderen Reihenfolge in Ihrer Plone-Site erscheinen.

Warum erscheint der Text in Kleinbuchstaben?

Plone ändert mit einem Stylesheet die Schreibweise von vielen Dingen, z.B. Reitern, in Kleinbuchstaben. Um das abzustellen, können Sie das Stylesheet ändern, was später in diesem Kapitel im Abschnitt »Bilder und CSS ändern« beschrieben wird.

Exkurs: Buch-Website: Einen neuen Reiter hinzufügen

Durch das Hinzufügen oder Entfernen eines Reiters in den Portalreitern entsteht eine nette Navigationshilfe. Deswegen werden Sie an dieser Stelle einen Reiter hinzufügen, der auf den `Software`-Ordner verweist, und außerdem werden Sie die Reiter NACHRICHTEN und MITGLIEDER entfernen (die auf meiner Site unnütz sind). Gehen Sie zurück ins ZMI, und klicken Sie auf PORTAL_ACTIONS. Scrollen Sie zum Seitenende bis zum HINZUFÜGEN-Formular. Ich habe das Formular mit folgenden Werten ausgefüllt:

- Geben Sie im Name-Feld **Software** ein.
- Geben Sie im ID-Feld **software_tab** ein.
- Geben Sie im Action-Feld **string:$portal_url/Software** ein.
- Geben Sie im Permission-Feld **View** ein.
- Geben Sie im Category-Feld **portal_tabs** ein.
- Geben Sie im Visible-Feld **selected** ein.

Weiterhin habe ich die Aktionen für Nachrichten und Mitglieder gefunden und dort VISIBLE ausgeschaltet (und natürlich auf SAVE geklickt). Zurück in der Plone-Schnittstelle sehen Sie nun die neuen Reiter. Der Schlüsselwert hierbei ist *Action*, ein TALES-Ausdruck. Diese Ausdrücke werden in Kapitel 5 ausführlich beschrieben. Der *Action*-Wert enthält den URL des Ordners, auf den Sie zeigen, in meinem Fall `Software`, und ist in der Wurzel meiner Plone-Site abgelegt. Daher lautet der Ausdruck `string:$portal_url/Software`. <)>

Icons für ein Dokument ändern

Wenn Sie eine Liste von Links oder Optionen in einer Plone-Site betrachten, ist es sehr wahrscheinlich, dass diese Liste von einer Serie von Aktionen produziert wird. Und wenn es keine Aktionen sind, dann ist es Code; aber viele der wesentlichen Eigenschaften der Plone-Schnittstelle werden aus den Einstellungen im ZMI dynamisch generiert. Zwei weitere Beispiele für Aktionen sind Dokument- und Site-Aktionen.

Die Site-Aktionen erscheinen in der oberen rechten Ecke als Links zum Ändern der Schriftgröße. Diese Links könnten alles Mögliche sein, aber zufällig verweisen sie auf einige clientseitige Scriptfunktionen. Auch diese Links werden in *portal_actions* konfiguriert und sind nichts weiter als Aktionen, die zu einer anderen Kategorie gehören. Wenn Sie sich die Aktionen in *portal_actions* anschauen, werden Sie am unteren Seitenende diese drei Aktionen sehen. Sie haben die Kate-

gorie *site_actions*. Wenn Sie sie entfernen möchten, müssen Sie lediglich die VISIBLE-Option ausschalten. Die Icons stammen aus dem Werkzeug *portal_actionicons*, das lediglich ein weiteres einfaches Werkzeug ist, das das Icon auf die Aktion abbildet. Unter *portal_actionicons* sehen Sie eine Übereinstimmung bei *normal_text* für *site_actions*, das auf ein Icon passt (siehe Abbildung 4.20).

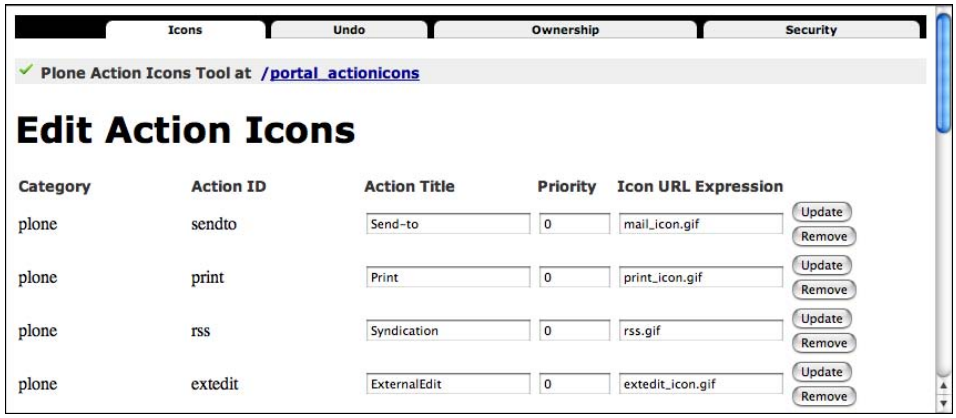


Abbildung 4.20: Site-Aktionen

Analog dazu befinden sich Dokumentationen in *portal_actions*. Sie haben die Kategorie *document_actions*. Auch hier können Sie die Reihenfolge, Icons und den Text ändern und Icons in der Schnittstelle hinzufügen oder entfernen, indem Sie diese Aktionen bearbeiten.

4.2.8 Bilder und CSS ändern

Das Look-and-Feel einer Plone-Site ist ein umfangreiches Thema, das drei Kapitel für sich in Anspruch nimmt, Kapitel 5 bis 7. Die folgenden Abschnitte behandeln die Grundlagen und zeigen, wie man schnell ein paar Änderungen vornimmt. Sie erklären aber nicht, wie alles funktioniert.

Eine *Skin* besteht aus einer Menge von CSS, Bildern, Templates und Scripts, die zusammen ein Look-and-Feel für den Benutzer bilden. Eine Skin dient dazu, das Aussehen und damit das Look-and-Feel einer Website ändern zu können, ohne den Inhalt ändern zu müssen.

Skins ändern

Sie können die Standard-Skin einer Site ändern, indem Sie das Formular EINSTELLUNGEN DES AUSSEHENS benutzen, zu dem Sie vom Control Panel aus gelangen. Eine Plone-Site können Sie auf einige verschiede Weisen darstellen, indem Sie verschiedene Farben, Stylesheets und Templates auf einer Site benutzen.

Dieses Formular enthält folgende Optionen:

- **Voreingestelltes Aussehen**: Dies ist die Standard-Skin, wenn Benutzer auf die Site zugreifen. Es gibt nur eine standardmäßig vorgegebene Skin, nämlich Plone Default.
- **Flexibilität des Aussehens**: Damit bestimmen Sie, ob es Benutzern erlaubt ist, ihre eigene Skin auszuwählen. Wenn ja, dann kann ein Benutzer in seinen Voreinstellungen eine neue Skin wählen. Standardmäßig ist diese Option gewählt.
- **Wirkungsdauer des Cookies für das Aussehen**: Wenn ein Benutzer eine Skin wählen kann, dann sollten Sie diese Option wählen, damit das entsprechende Cookie beliebig lange gültig ist. Damit wird ein Benutzer immer diese Skin sehen, wenn er sich bei der Site anmeldet. Diese Option ist standardmäßig nicht gewählt.

Nehmen Sie die gewünschten Änderungen vor, und klicken Sie auf SPEICHERN, um diese zu bestätigen. Um die Performance auf der Site zu verbessern, können Sie ein Caching von Bildern und Stylesheets verwenden. Um sicherzustellen, dass Sie die neue Skin auch wirklich so sehen, wie sie sein soll, löschen Sie den Cache Ihres Browsers (beim Internet Explorer drücken Sie dazu Strg + F5).

Ein anderes Logo einstellen

Das Logo einer Plone-Site zu ändern und statt des Plone-Logos ein anderes zu verwenden, ist eine leichte Übung, aber die einzelnen Schritte können ein wenig verwirrend sein. Daher sollten Sie sie sorgfältig befolgen.

Zuerst greifen Sie auf das ZMI zu, klicken auf PORTAL_SKINS, dann auf PLONE_IMAGES und schließlich auf LOGO.JPG. Danach wird die Seite zu diesem Objekt geöffnet. Sie sollte in etwa so aussehen wie in Abbildung 4.21.

Abbildung 4.21: Das Standard-Logo

Dieses Objekt stellt das Logo dar, wie es in Zope benutzt wird. In Abbildung 4.21 können Sie deutlich Informationen zum Bild sehen, seine Größe, seinen Typ und seinen Platz im Dateisystem. In der Seitenmitte befindet sich der CUSTOMIZE-Button, den Sie nun anklicken. Dabei wird eine Kopie des Objekts namens `logo.jpg` im `custom`-Ordner erzeugt (siehe Abbildung 4.22).

Hinweis

Wenn Sie an dieser Stelle eine Fehlermeldung wegen einer fehlerhaften Anfrage erhalten, gehen Sie zurück zu `portal_skins/custom`. Sie werden ein Objekt namens `logo.jpg` sehen. Klicken Sie auf dieses Objekt. Es kann nur ein Objekt namens `logo.jpg` im `custom`-Ordner geben, und die Fehlermeldung warnt Sie davor, dass diese Prozedur bereits ausgeführt worden ist. Falls Sie das ursprüngliche Objekt anpassen möchten (mit anderen Worten, wenn Sie diese Schritte wiederholen möchten), müssen Sie das Objekt innerhalb von `custom` löschen.

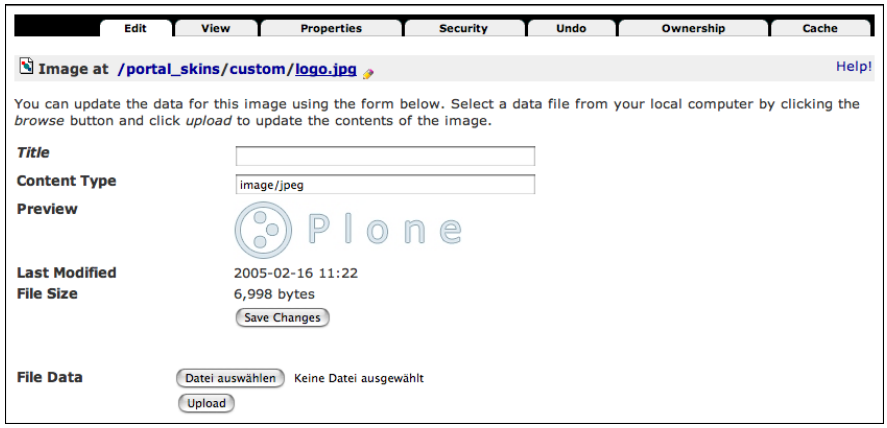


Abbildung 4.22: Das angepasste Bild

Diese Seite sieht vielleicht ähnlich wie die vorherige Seite in Abbildung 4.21 aus, aber es gibt eine Reihe von Unterschieden. Erstens: Wenn Sie in die obere linke Ecke der Seite schauen, werden Sie bemerken, dass sich der *meta_type* und der Ort dieses Objekts verändert haben. Nun sind Sie sind nicht mehr in `portal_skins/plone_images/logo.jpg`, sondern Sie sind in `portal_skins/custom/logo.jpg`. Zweitens können Sie nun einen BROWSE-Button sehen, mit dem Sie ein Bild auswählen und hochladen können, d.h., Sie können dieses Bild ändern. Klicken Sie auf diesen Button, um Ihr neues Bild zu finden, und klicken Sie auf SAVE, um die Änderung zu bestätigen. In Abbildung 4.23 füge ich als Beispiel ein kanadisches Plone-Logo hinzu.

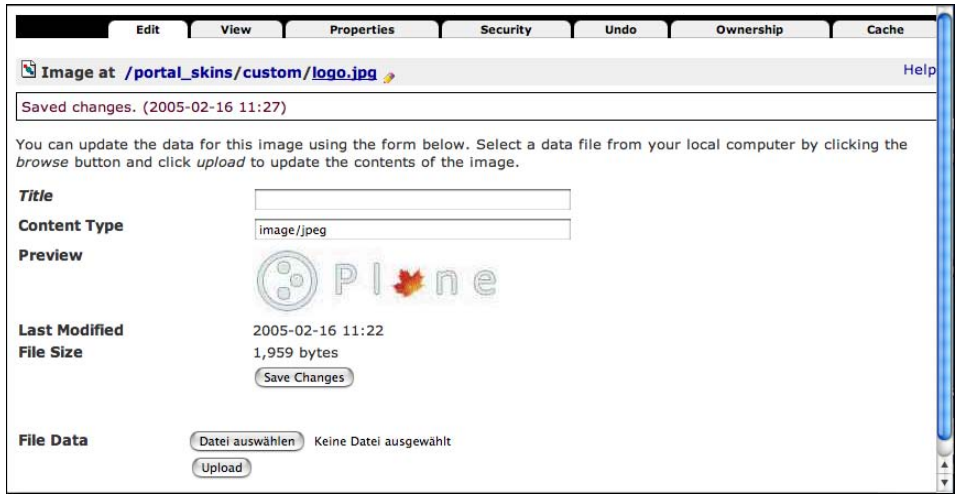


Abbildung 4.23: Das kanadische Plone-Logo

Wenn Sie nun zur Plone-Schnittstelle zurückgehen, sehen Sie, dass sich das Bild verändert hat. Um sicherzugehen, dass Sie das neue Bild sehen, sollten Sie den Cache Ihres Browsers löschen (beim Internet Explorer drücken Sie dazu Strg+F5).

Was tun Sie, wenn Ihr Bild nicht im JPEG-Format vorliegt?

Zope bestimmt den MIME-Typ (Multipurpose Internet Mail Extensions) nicht aus der Dateierweiterung, sondern aus dem Inhalt. Daher können Sie ein GIF-Bild unter `logo.jpg` hochladen, und es funktioniert trotzdem, weil der korrekte MIME-Typ `image/gif` angewendet wird. Um die Verwirrung in Grenzen zu halten, möchten Sie das Bild aber vielleicht doch `logo.gif` oder `logo.png` nennen.

CSS-Code ändern

CSS bestimmt einen Großteil des Look-and-Feel Ihrer Website, darunter die Reiter, Bilder, Kästen und das allgemeine Layout. Die Tatsache, dass Plones CSS durchgehend angepasst werden kann, bedeutet, dass die Benutzer viele Aspekte einer Site mit ein paar Stylesheets vollkommen umgestalten können.

Noch einmal: Kapitel 7 beschreibt, was die einzelnen Elemente bewirken. In diesem Abschnitt zeige ich Ihnen nur schnell, wie Sie den CSS-Code einer Plone-Site ändern können. Zuerst gehen Sie ins ZMI. Dort klicken Sie auf PORTAL_SKINS, dann auf PLONE_STYLES und schließlich auf PLONECUSTOM.CSS. Danach erscheint eine Seite zu diesem Objekt. Dieses Stylesheet ist ziemlich einfach, ja, es ist sogar leer. Plone macht nämlich Gebrauch von der Kaskadierungseigenschaft von CSS. Da der HTML-Code für Plone zuerst `plone.css` importiert und dann `ploneCustom.css`, überschreiben alle Änderungen in letzterem das Standard-Stylesheet.

Warum ist das eine gute Idee? Weil es bedeutet, dass Sie kleine inkrementelle Änderungen an `ploneCustom.css` vornehmen können, ohne das Kern-Stylesheet kaputtzumachen oder zu verändern.

Um also das Objekt `ploneCustom.css` anzupassen, klicken Sie auf PORTAL_SKINS, dann auf PLONE_STYLES und auf PLONECUSTOM.CSS. Danach klicken Sie auf den CUSTOMIZE-Button. Wieder wurde dieses Objekt angepasst, und Sie bemerken, dass Sie nicht mehr unter `portal_skins/plone_styles/ploneCustom.css`, sondern nun unter `portal_skins/custom/ploneCustom.css` sind. Da Dateiobjekte nun über das Web bearbeitet werden können, können Sie das Stylesheet direkt über das Web bearbeiten.

Ändern Sie als Beispiel den Hintergrund so, dass er ein Bild in der Mitte enthält (das ist nicht unbedingt die beste Benutzerschnittstelle, aber ein klares Beispiel dafür, wie Sie den CSS-Code anpassen können). Zuerst müssen Sie ein Bild in Plone hochladen. Dazu klicken Sie auf PORTAL_SKINS, auf CUSTOM und dann auf den ADD-Button, und schließlich wählen Sie ein Bild aus, wie in Abbildung 4.24 zu sehen ist.

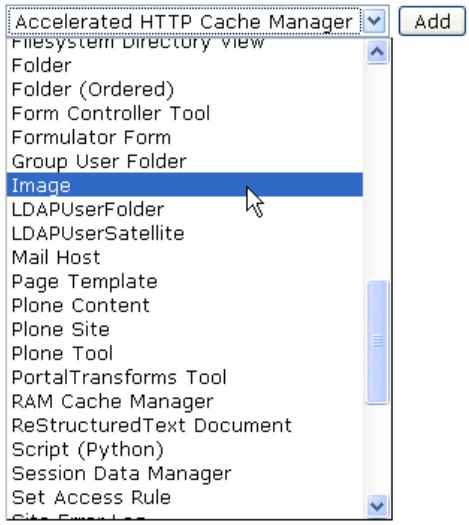


Abbildung 4.24: Ein neues Bild hinzufügen

Als Datei habe ich ein Bild gewählt, das ich im Web gefunden habe (und das auch auf der Website zum Plone-Buch verfügbar ist), aber Sie können ein beliebiges Bild nehmen. Vergewissern Sie sich, dass die ID des Bildes `background.gif` ist, wie in Abbildung 4.25 zu sehen ist.

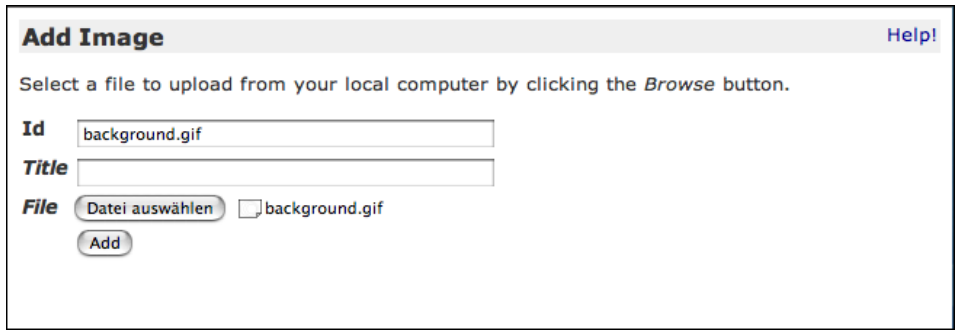


Abbildung 4.25: Prüfen des neuen Bildes

Nun müssen Sie den CSS-Code ändern, damit er auf das neue Bild zeigt. Den CSS-Code haben Sie schon angepasst, gehen Sie also zu `portal_skins/custom/ploneCustom.css` zurück, und ändern Sie den Text von

```
/* DELETE THIS LINE AND PUT YOUR CUSTOM STUFF HERE */
```

in Folgendes:

```
body {
   background-image: url(background.jpg);
   background-repeat: no-repeat;
   background-position: center;

}
```

Klicken Sie auf SAVE CHANGES, um die Änderungen an dieser Datei nicht zu verlieren. Gehen Sie dann zur Plone-Schnittstelle zurück. Wenn alles gut ging, sollten Sie das neue Bild sehen (siehe Abbildung 4.26).

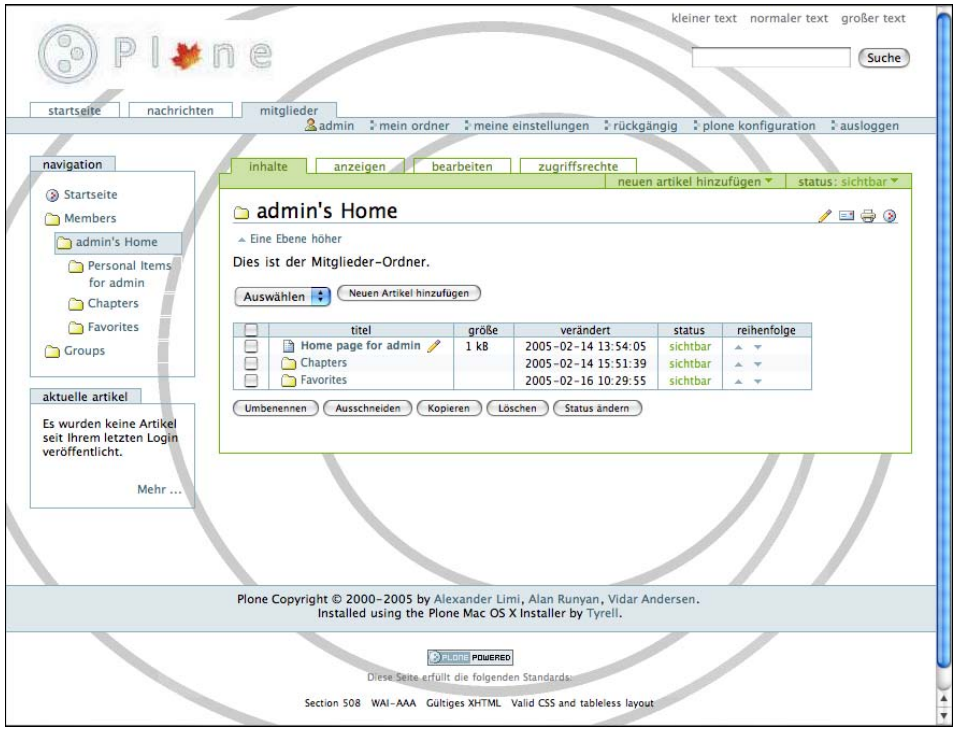


Abbildung 4.26: Das neue Hintergrundbild

5 Einführung in einfaches Plone-Templating

Plone kombiniert drei Technologien bei der Erstellung einer Webseite. Python und Seiten-Templates erzeugen HTML, das an den Browser gesendet wird. Dort wird mit einigen CSS-Stylesheets die hübsche Seite dargestellt, mit der Sie nunmehr vertraut sind. In diesem Kapitel und in Kapitel 6 stellen die beiden ersten Elemente, der Python-Code und die Seiten-Templates, den Hauptgegenstand der Betrachtung dar.

Um zu verstehen, wie man ein Plone-Template erzeugt und bearbeitet, muss man zunächst einiges über die wesentlichen zugrunde liegenden Konzepte lernen. Manche davon sind für Plone besonders einmalig, und obwohl sie große Vorteile haben, braucht man einige Zeit, um sich daran zu gewöhnen.

In diesem Kapitel beginne ich damit, dass ich das Objekt-Publishing behandle. Ich erkläre, wie man mit Objekten innerhalb von Plone interagiert. Danach erkläre ich, wie man Ausdrücke zusammensetzt. Sobald Sie mit diesen beiden Konzepten vertraut sind, beschreibe ich, wie Plone-Seiten wirklich zusammengestellt werden. Am Ende dieses Kapitels werden Sie eine neue Seite in Ihrer Plone-Site erstellen, die jene Techniken verwendet, die Sie bislang kennen gelernt haben.

Dieses Kapitel macht etwas mehr Sinn, wenn Sie mit Python schon etwas vertrauter sind. Ich werde aber durchgehend das Konzept hinter dem Code erklären, d.h., auch wenn Sie kein Python verstehen, sollten Sie damit zurechtkommen. Der Rest des Buchs verweist auf TALES- (Template Attribute Language Expression Syntax) und Script(Python)-Objekte, d.h., Sie sollten sich in diesem Kapitel die Zeit nehmen, sich mit ihnen vertraut zu machen. Aber Sie sollten bereits einen guten Start haben: TALES habe ich schon im vorigen Kapitel eingeführt, weil es bei der Erzeugung von Portlets und Aktionen benutzt wird.

5.1 Die zugrunde liegende Templating-Maschinerie

Ein direktes Eintauchen in die Arbeitsweise von Plone-Templates würde Sie vermutlich ein wenig verwirren, daher gehe ich erst einmal die Templating-Maschinerie durch, auf der sie aufbauen. Idealerweise wäre das etwas, worüber Sie sich keine Sorgen machen müssten, aber in der Praxis habe ich festgestellt, dass es die erste Hürde ist, über die man stolpert, wenn man anfängt, Plone zu benutzen.

Plone ist insofern einmalig, als alles in Plone ein Objekt ist. Seien Sie beruhigt, wenn Sie mit dem Konzept von Objekten nicht so recht vertraut sind. Es steckt nicht viel dahinter: Ein *Objekt* ist einfach ein Ding, das ein bestimmtes Verhalten hat. Jedes Objekt verfügt über Methoden, die Sie aufrufen können. Eine Computer-Maus könnte beispielsweise Methoden namens `bewegen`, `klicken` und `rechtsklicken` haben.

Ein Dokument in Plone ist ein Objekt eines bestimmten Typs. Das heißt nichts weiter, als dass das Dokument nicht einfach nur aus statischem Text besteht, sondern ein bisschen komplizierter und weitaus nützlicher ist. In Plone hat ein Dokument z.B. eine Methode namens `description`, die Ihnen eine Beschreibung liefert, die der Benutzer hinzugefügt hat. Bei der Benutzung des Templating-Systems werden Sie sehen, dass alles ein Objekt ist. Zuerst werden Sie aber ein paar Grundprinzipien beim Objekt-Publishing kennen lernen.

5.1.1 Einführung in das Objekt-Publishing

In Plone veröffentlichen Sie eigentlich Objekte, die sich in Zope befinden. Die meisten davon sind Objekte, die persistent in der Objektdatenbank gespeichert sind. Das Konzept ist komplizierter als bei den üblichen CGI-Umgebungen (Common Gateway Interface), in denen Sie ein Script ausführen und ihm eine Reihe von Anfragevariablen übergeben. In Plone ist alles ein Objekt, in Zope ist alles ein Objekt, und auch in Python ist alles ein Objekt. Bis jetzt habe ich es vermieden, das Wort *Objekt* zu verwenden, und habe stattdessen Wörter wie *Template*, *Script* und *Eintrag* benutzt, aber all das sind wirklich Objekte, die sich lediglich unterschiedlich verhalten.

Wenn Sie von Plone einen URL (Uniform Resource Locator) abfragen, wird ein Objekt aus der Umgebung aufgerufen. Das macht Plone so, dass es den URL in einen Pfad übersetzt. Das heißt: Falls der URL `/Plone/login_form` lautet, versucht Plone, aus diesem URL einen Pfad zu bestimmen und all die Objekte darauf in der Datenbank zu finden. Dabei findet es das Objekt `Plone` und darin dann ein Objekt `login_form`. Dieses Absuchen des Pfads wird auch *Traversierung* genannt. Im Grunde genommen traversiert Zope diese Objekte und ruft das letzte Objekt im Pfad auf.

Wenn Zope das Objekt `login_form` aufruft, wird dieses Objekt in seinem Kontext ausgeführt. Den Begriff *Kontext* werden Sie im Zusammenhang mit Plone oft hören. Es ist lediglich der aktuelle Kontext des ausgeführten Objekts. In diesem Fall ist es `/Plone`. Der Kontext verändert sich ständig, während Sie sich in einer Plone-Site bewegen. Falls Sie den URL `/Plone/Members/login_form` in einem Browser aufgerufen haben, wäre der Kontext `/Plone/Members`.

Wie oben erwähnt wurde, können Sie per *Traversierung* mit Hilfe eines Programms genauso auf Objekte in Plone zugreifen, wie Sie es mit einem URL tun. Das ähnelt dem Zugriff auf Dateien in einem Dateisystem. Das heißt: Wenn Sie unter Windows auf ein Bild in `Eigene Dateien` zugreifen möchten, würden Sie ein Verzeichnis eingeben wie **C:\Dokumente und Einstellungen\andym\Eigene Dateien\Mein Portrait.jpg**. Auf ein Objekt in Plone könnten Sie zugreifen, indem Sie **Members/andy/Mein Portrait.jpg** eingeben. Das würde auch funktionieren, sofern Sie eine Reihe von Ordnern und Objekten wie folgt hätten:

```
Members
  |_ andy
     |_  Mein Portrait.jpg
```

In der Dateisystemversion gehen Sie auf der Festplatte des Rechners Verzeichnis für Verzeichnis durch. In Plone geschieht das Gleiche, nur ist es so, dass `Members` und `andy` Objekte sind.

Ein Unterschied besteht darin, dass Zope die Schreibweise beachtet. Unter Windows können Sie **Mein Portrait.jpg** oder **mein portrait.jpg** eingeben. In Plone wird das aber nicht funktionieren, denn Sie müssen die gleiche Schreibweise wie in der Objekt-ID verwenden. Aus diesem Grund empfiehlt es sich, in allen URLs Kleinbuchstaben zu verwenden, damit Ihre Benutzer weniger Fehler machen können.

Plone und Zope haben eine Besonderheit namens *Akquisition* zu diesem Publishing-System hinzugefügt. Das Konzept dahinter ist das des Enthaltenseins: Objekte befinden sich in anderen Objekten, die als *Container* bezeichnet werden. Im vorigen Beispiel ist das Objekt `andy` ein Container innerhalb des `Members`-Containers innerhalb des Plone-Site-Containers (der sich seinerseits innerhalb des Zope-Application-Containers befindet).

In einer normalen objektorientierten Umgebung erbt ein Objekt Verhalten von seinem Elternobjekt. In Plone und Zope erbt ein Objekt auch Verhalten von seinem Container. Dabei wandert ein Objekt durch eine Container-Hierarchie, um herauszufinden, wo es dieses Verhalten findet.

Nehmen Sie z.B. den Zugriff auf `Members/andy/Mein Portrait.jpg`. Was geschähe, wenn das Objekt `Ein Bild.jpg` nicht im Ordner `andy` vorhanden wäre, sondern

weiter oben in der Hierarchie? Nun, dank Akquisition würde es für Sie gefunden. Betrachten Sie die folgende Hierarchie:

```
Members
   |_ andy
   |_ Mein Portrait.jpg
```

In diesem Fall würde Plone, wenn Sie die URL ausführen, diese bis `andy` traversieren und dann versuchen, dort `Mein Portrait.jpg` zu finden, was aber in dem Container nicht existiert. Daher würde es in der Container-Hierarchie suchen, also im Ordner `Members`, wo es `Mein Portrait.jpg` findet und zurückgibt. Als Ergebnis sieht der Benutzer das Bild wie sonst auch.

Wenn Sie das jedoch mit dem vorigen Beispiel vergleichen, in dem das Bild im Ordner `andy` enthalten war, würden Sie folgende Unterschiede feststellen:

- Erstens ist der Kontext der gleiche, auch wenn sich das Objekt an einem anderen Ort befindet. Der Kontext basiert auf dem Ort, von dem aus das Objekt aufgerufen wird.
- Zweitens ist der Container ein anderer, und der Container von `Mein Portrait.jpg` ist nun ein anderer, nämlich `Members` und nicht `andy`.

Worauf will ich also hinaus? Nun, jetzt können Sie ein Objekt in die Wurzel einer Plone-Site setzen, und ein beliebiges anderes Objekt kann darauf zugreifen, weil ersteres mit Hilfe der Akquisition gefunden wird.

Auch wenn das alles vermutlich Sinn macht, sollten Sie wissen, dass Akquisition ziemlich kompliziert werden kann, besonders bei der Suche in der Kontexthierarchie (die vorkommen kann). Wenn Sie gern mehr darüber lernen möchten, können Sie eine hervorragende Abhandlung zum Thema Akquisition von Jim Fulton, dem Systemarchitekten von Zope, unter *http://www.zope.org/Members/jim/Info/IPC8/AcquisitionAlgebra/index.html* lesen.

5.1.2 Einführung in Template-Ausdrücke

Bevor Sie ins Zope Page Templates-System eintauchen, müssen Sie TALES verstehen. Oft müssen Sie in einer Anwendung Ausdrücke schreiben, die dynamisch ausgewertet werden können. Das sind keine Scripts, sondern *einzeilige* einfache Ausdrücke, die etwas Einfaches mit einer Zeile Code machen können.

Ein Ausdruck wird mit einer Reihe von lokalen Variablen ausgewertet, die an ihn übergeben werden. Diese Variablen werden durch das bestimmt, was den Ausdruck aufruft. Vom Workflow wird eine Reihe Variablen übergeben, und das Zope Page Templates-System übergibt eine Reihe weiterer. Vorläufig werde ich

Beispiele verwenden, die einen `context` haben. Wie Sie sich hoffentlich erinnern, ist `context` der Kontext, in dem ein Objekt aufgerufen wird.

Bislang haben Sie schon einige TALES-Ausdrücke wie `string:${portal_url}/Software` gesehen. Dies ist aber nur ein Beispiel aus einem weiten Spektrum möglicher Ausdrücke. Die Hauptanwendung von TALES liegt bei den Zope Page Templates, dem System, mit dem Plone HTML generiert. Auch wenn der Name vermuten lässt, dass es nur in Templates zu gebrauchen ist, verwenden viele Werkzeuge in Plone diese Syntax bei einfachen Ausdrücken, z.B. Aktionen, Workflow und Sicherheit. Es gibt verschiedene Arten von Ausdrücken, die ich einzeln durchgehen werde.

Pfadausdrücke verwenden

Der Pfadausdruck ist der Standardausdruck, der auch am meisten verwendet wird. Anders als alle anderen Ausdrücke benötigt er kein Präfix, um den Typ des Ausdrucks anzugeben. Der Ausdruck umfasst einen oder mehrere Pfade. Die Pfade sind mit einem senkrechten Strich (|) voneinander getrennt. Jeder Pfad besteht aus einer Reihe von Variablen, die mit Schrägstrichen (/) voneinander getrennt sind. Hier sind einige einfache Beispiele:

```
context/message
context/ordnerA/title
context/Members/andy/Mein Portrait.jpg
```

Bei der Auswertung des Ausdrucks wird der Pfad an den Schrägstrichen aufgeteilt. Dann wird ausgehend vom linken Wert versucht, das Objekt, die Methode oder den Wert zu finden. Dieses Objekt wird dann auf den aktuellen Stack gelegt, und es geht mit dem nächsten Wert weiter. Das wird so lange wiederholt, bis das Ende des Ausdrucks erreicht ist oder kein passender Wert gefunden wird. Falls das gefundene Objekt ein Python-Dictionary oder ein Abbildungsobjekt ist, wird dieser Wert des Dictionarys aufgerufen. Eine nette Eigenschaft von Pfadausdrücken ist, dass / das einzige reservierte Zeichen ist. Das heißt, Namen dürfen Leerzeichen und Punkte enthalten und können dennoch ausgewertet werden.

Wenn das Ende erreicht ist, wird dieses Objekt aufgerufen, sofern das möglich ist. Handelt es sich um ein nicht aufrufbares Objekt, wird der String-Wert des Objekts geholt und zurückgegeben. Falls bei dieser Suche zu irgendeinem Zeitpunkt ein Fehler auftritt (am häufigsten passiert es, dass das verlangte Attribut nicht existiert), wird mit dem nächsten Alternativausdruck weitergemacht, sofern einer vorhanden ist. Einen Alternativausdruck können Sie nach einem senkrechten Strich angeben.

Beispiel:

```
context/ordnerA/title|context/ordnerB/title
```

Im vorigen Beispiel wird der Titel von `ordnerA` ausgegeben, wenn er existiert, oder der Titel von `ordnerB`, falls der erste nicht existiert. Dieser Vorgang wird für jeden Ausdruck so lange wiederholt, bis es keine weiteren Ausdrücke gibt oder einer davon erfolgreich ausgewertet werden kann.

Not-Ausdrücke verwenden

Ein negierter Ausdruck hat das Präfix `not:` und invertiert einfach die Auswertung des TALES-Ausdrucks, der dem Präfix folgt. Da das Zope Page Templates-System keine `if`-Anweisung kennt, können Sie hiermit das Gegenteil einer vorherigen Bedingung testen.

Beispiel:

```
not: context/message|nothing
```

Nocall-Ausdrücke verwenden

Wenn ein Pfadausdruck das letzte Element im Pfad erreicht, wird dieses, wenn möglich, standardmäßig aufgerufen. Das Präfix `nocall:` verhindert genau das. Ein Nocall-Ausdruck wird in Plone selten verwendet, hat aber gelegentlich seinen Zweck. Sie können ihn z.B. dafür benutzen, auf ein anderes Objekt zu verweisen, es aber nicht auszugeben.

Beispiel:

```
nocall: context/einBild
```

String-Ausdrücke verwenden

Mit String-Ausdrücken können Sie Text und Variablen in einem Ausdruck mischen. Alle String-Ausdrücke beginnen mit dem Präfix `string:`. Dies ist eine nützliche und recht häufig benutzte Funktion. Der Text darf alles enthalten, was in einem Attribut erlaubt ist, d.h. im Wesentlichen alphanumerische Zeichen und Leerzeichen. In Text können Variablen enthalten sein, denen ein Dollar-Zeichen (`$`) vorangestellt sein muss. Hier einige Beispiele:

```
string: Dies ist ein langer String
string: Dies ist der $title
```

Im letzten Beispiel wird die Variable `$title` ausgewertet. Die Variable kann ein beliebiger Pfadausdruck sein. Wenn sie `/` enthält, muss die Variable in `{}` verpackt werden, um den Anfang und das Ende des Ausdrucks deutlich zu machen.

Beispiel:

```
string: Dies ist der ${context/einBild/title}.
```

Falls ein Dollar-Zeichen im Text geschützt werden muss, setzen Sie ein weiteres Dollar-Zeichen vor das zu schützende Dollar-Zeichen.

Beispiel:

```
string: In $$US kostet es ${context/meinDing/kosten}.
```

Python-Ausdrücke verwenden

Python-Ausdrücke werten eine Zeile Python-Code aus. Alle Python-Ausdrücke beginnen mit dem Präfix `python:` und enthalten eine Zeile Python-Code.

Beispiel:

```
python: 1 + 2
```

Der Python-Code wird unter dem gleichen Sicherheitsmodell ausgewertet, das auch ein Script(Python)-Objekt benutzt, wie in Kapitel 6 dargestellt wird. Aus diesen Gründen sollte der Python-Code einfach und auf Präsentationsfunktionen beschränkt sein, z.B. auf die Formatierung von Strings und Zahlen oder auf das Aufstellen einfacher Bedingungen.

Weiterhin gilt, dass fast alle anderen erwähnten TALES-Ausdrücke in Python verpackt und aufgerufen werden können. Folgendes sind die entsprechenden Ausdrücke:

- **path(string)**: Wertet einen Pfadausdruck aus.
- **string(string)**: Wertet einen String-Ausdruck aus.
- **exists(string)**: Wertet einen String-Ausdruck aus.
- **nocall(string)**: Wertet einen Nocall-Ausdruck aus.

Beispielsweise ist folgender Code:

```
python: path('context/Members')
```

äquivalent zu folgendem:

```
context/Members
```

Einige Funktionen wurden auch hinzugefügt, um Entwicklern ihre Arbeit bequemer zu machen. Die Funktion *test* erwartet drei Parameter: eine auszuwertende Anweisung und die Bedingungen für *true* und *false*. Die Anweisung wird ausgewertet, und es wird der entsprechende Wert zurückgegeben.

Beispiel:

```
python: test(1 - 1, 0, 1)
```

Die Funktion `same_type` erwartet zwei Parameter und stellt fest, ob sie gleich sind. Beispiel:

```
python: same_type(irgendwas, '')
```

Manche Entwickler raten davon ab, Python-Code im Zope Page Templates-System zu benutzen, weil das bedeutet, dass Logik zu den Präsentations-Templates hinzugefügt wird. Für Entwickler ist es oftmals hilfreich, sich bei jedem hinzugefügten Python-Happen zu fragen, ob dieser Code nicht besser ausgegliedert und in ein separates Script(Python)-Objekt verlagert werden sollte. Das heißt nicht, dass Sie jedes Stück Python-Code ausgliedern sollen, sondern nur, dass Sie darüber nachdenken sollten, bevor Sie etwas hinzufügen.

Hinweis

In Kapitel 4 haben Sie eine Aktion zum Verweisen auf den Software-Teil der Site in Form eines Portalreiters hinzugefügt. In jener Aktion hatten Sie im String-Ausdruck `string: ${portal_url}/Software` hinzugefügt. Das macht jetzt vielleicht etwas mehr Sinn, nachdem ich die Variable `portal_url` erklärt habe. Dies ist der URL Ihres Portals, der variieren kann, falls Sie virtuelles Hosting verwenden. Das wird mit Hilfe von Akquisition gemacht, um das Objekt `portal_url` zu erhalten und das Ergebnis in den String einzufügen. Als Ergebnis erhalten Sie immer einen absoluten Link auf den Ordner `Software`.

Hinweis

Ich habe öfter beobachtet, wie Neulinge Python und Strings mischen. Alle Ausdrücke sind verschieden, d.h., Sie können keine pfadähnlichen Ausdrücke in einen Python-Ausdruck setzen. So macht z.B. der Ausdruck `python: here/Members + "/danae"` keinen Sinn. Der gesamte Ausdruck wird als Python-Code interpretiert, d.h., Plone versucht, `here` von `Members` zu trennen, und Sie erhalten einen Fehler. Das ist die ideale Gelegenheit, einen String-Ausdruck zu verwenden (in dem Sie Variablen ersetzen können). Das heißt, die Variable enthält einen Pfadausdruck. Sie könnten also `string: ${here/Members}/danae` benutzen.

5.2 Das Zope-Page Templates System verwenden

Nun da Sie das Object-Publishing und Ausdrücke verstehen, können Sie sich aufs Eingemachte im System stürzen, auf Zope Page Templates. Das ist das Templating-System, mit dem Plone HTML generiert.

Es sind sehr viele Systeme zum Generieren von HTML verfügbar, und zu den bekannteren gehören JavaServer Pages, Active Server Pages und PHP. Den Benutzern dieser anderen Systeme sei gesagt: Das Zope Page Templates-System sieht zu Beginn sehr merkwürdig aus, aber Sie werden schnell sehen, dass es ein sehr mächtiges System ist.

Das einfachste Template sieht ungefähr wie folgt aus:

```
<p tal:content="here/message">Der Titel</p>
```

Wenn der Wert von `message` gleich `Hello, World!` wäre, dann wäre Folgendes eine Ausgabe bei der Darstellung des Templates:

```
<p>Hello, World!</p>
```

Im Moment überspringe ich einige der Feinheiten und zeige, was hier passiert ist. Ein normaler Absatz wurde in HTML geschrieben, aber der Inhalt dieses Absatzes ist nicht der Text, der in der Ausgabe gezeigt wird. Zu dem öffnenden Tag des Absatzes wurde das Attribut `tal:content` hinzugefügt, und der Ausdruck `here/message` wurde diesem Attribut zugewiesen. Als Absatzinhalt wurde aber der Wert der Variablen `message` ausgegeben (in diesem Fall, `Hello, World!`).

Das Template wird zur Laufzeit ausgewertet, und das Attribut `tal:content` wird aufgerufen. Der `tal`-Teil steht für Template Attribute Language, eine Sprache, die eine Reihe von Befehlen enthält, z.B. `content`. All diese Befehle werden Sie später noch sehen. Mit ihnen können Sie fast alles tun, was Sie mit HTML-Tags tun möchten. Sie können Schleifen erzeugen, Tags ändern, Attribute ändern, Tags entfernen usw. Bevor das Template ausgeführt wird, wird es als gültiges XHTML (Extensible HTML) angezeigt und wird in einem Editor als Absatz mit diesem Text angezeigt.

All diese Page Templates sind gültiges XHTML. XHTML ist ein Standard für HTML und ist gültiger XML-Code (Extensible Markup Language). Das bedeutet, Sie müssen sich an folgende Regeln halten:

- Alle Tags müssen in Kleinbuchstaben geschrieben sein.
- Attribute müssen immer in Anführungszeichen stehen (z.B. `<input type="checkbox" checked="1" />`).
- Leere Elemente müssen terminiert werden (z.B. `<br />`, nicht `<br>`).

Um eine Seite als XHTML zu definieren, müssen Sie eine DOCTYPE-Deklaration angeben und den XML-Namespace benutzen, der im `html`-Tag gesetzt ist. Plone verwendet folgende Deklaration oben auf jeder Seite:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
```

Weitere Informationen zur XHTML-Spezifikation finden Sie unter *http://www.w3.org/TR/xhtml1/#xhtml*.

Exkurs: Noch ein System zur HTML-Generierung?

In den Anfangsjahren des Webs waren Programmierer die maßgeblichen Erzeuger von HTML. Diese Programmierer haben schnell Systeme zusammengewürfelt, die HTML mit Hilfe eines Programms generierten, damit sie wieder ihre eigentlichen Jobs machen konnten. Mit solchen Werkzeugen wie den CGI-Modulen in Perl konnten Programmierer komplizierten serverseitigen Code für ihre Inhalte schreiben.

Sehr bald wurden Inhalte aber von allen möglichen Leuten erstellt, und dieser Vorgang musste vereinfacht werden. Zu diesem Zweck entstand eine Welle von Sprachen, die Code schützen sollten. Diese Sprachen verwendeten eine spezielle Art von HTML-Auszeichnung, die verarbeitet wurde, um eine Ausgabe zu produzieren. Wie schon erwähnt wurde, sind Active Server Pages und JavaServer Pages einige der verbreitetsten, und es basieren ganze Sprachen wie PHP auf diesem Konzept. Zope folgte diesem Trend mit DTML (Document Template Markup Language).

Diese Systeme nehmen HTML und vermischen es mit eigenen Tags wie `<% .. %>` oder `<dtml-... />`. Dieses Konzept war beliebt, weil es leicht zu verstehen war, und Benutzer mit grundlegenden HTML-Kenntnissen begriffen schnell, was die paar neuen Tags leisten sollten. Designer konnten den Inhalt solcher Tags ignorieren und den Programmierern überlassen. Programmierer konnten die relevanten Codeteile ändern, ohne den Inhalt zu ruinieren.

Allerdings weisen solche Systeme folgende Probleme auf:

- Die HTML-Templates skalieren mitunter nur sehr schwer, wenn immer mehr Inhalte zum Script hinzukommen. Die Seiten werden schnell riesig lang und schwierig zu verwalten.

- Logik und Inhalt sind nicht hübsch voneinander getrennt. Mit manchen dieser Systeme können sie zwar getrennt werden, aber die Möglichkeit, beliebiges HTML mit einem Stück Programmcode zu vermengen, ist zu verlockend. Oftmals wird aus Inhalt, Präsentation und Logik ein großer verwickelter Klumpen.
- Seiten können nicht einfach bearbeitet werden. Oftmals enthalten Seiten oder Templates eine Bemerkung der Art »diesen Teil nicht ändern...«, weil bei einer Änderung der Code drumherum nicht mehr funktioniert. WYSIWYG-Editoren (What You See Is What You Get) kann man so einstellen, dass sie einige Tags nicht ändern, aber sie können leicht andere kaputtmachen. In großen Organisationen müssen Benutzer mit verschiedenen Rechten alle die gleiche Seite ändern dürfen.
- Es kann sehr schwer sein, ein Standardergebnis zu sehen. Nehmen Sie z.B. eine Datenbankabfrage, deren Ergebnis in einer Tabelle dargestellt wird. Wie soll ein Designer sehen, wie das aussehen würde, ohne den Code auszuführen?

Aus diesen Gründen wurde das Zope Page Templates-System erfunden. Page Templates stellen einen neuen Ansatz dar: Anstatt eine weitere Methode zum Schutz von Code zu erfinden, wird der Code zu den vorhandenen Tag-Attributen hinzugefügt. Das Zope Page Templates-System ist nicht nur frei verfügbar und Open Source, es kommt auch ohne Zope aus! Im Moment existieren Versionen dieses Systems für Python, Perl und Java.

5.2.1 Einführung in Page Templates und Inhalt

Wie Sie nun wissen, ist Plone ein Content-Management-System, bei dem Benutzer über das Web Inhalte zur Plone-Site hinzufügen können. Diese Inhaltsobjekte werden innerhalb von Plone gespeichert und dann mit Hilfe von Page Templates für alle Welt dargestellt.

Kommen wir zum früheren Beispiel des Zugriffs auf `/Members/andy/Mein Portrait.jpg` zurück. Ich werde nun beschreiben, was mit dem Inhalt in Plone wirklich passiert. Zuerst findet Plone das Objekt `Mein Portrait.jpg` und ruft es auf. Es wird deswegen aufgerufen, weil keine spezielle Methode auf dem Objekt aufgerufen wird. Wenn ein Inhaltstyp aufgerufen wird, wird ein bestimmtes Template gefunden und dargestellt. Der Kontext für dieses Template wird das Bild sein, auf das Sie zugreifen möchten, und das Template ist jenes für dieses Bild.

Wenn eine andere Aktion auf dem Bild aufgerufen würde, z.B. `/Members/andy/MeinPortrait/image_edit`, dann würde die Aktion `image_edit` für dieses Objekt gesucht, und das entsprechende Template würde zurückgegeben werden. Wie das funktioniert, beschreibe ich in Kapitel 11 im Detail.

In allen Templates in Plone werden Sie also eine Referenz zu `here` oder `context` sehen. Das ist der Kontext des Inhalts, auf den zugegriffen wird. In einem Template können Sie nun `context/irgendwas oder wasanderes` schreiben, und das wird das `irgendwas oder wasanderes` sein, das relativ zu dem Stück Inhalt und nicht zum Template gesucht wird. Jetzt werden Sie Ihr erstes Template in Plone erstellen.

5.2.2 Erstellen Ihres ersten Page Templates

Die Standardmethode, ein Page Template zu erstellen, ist das Zope Management Interface (ZMI). Weil das bedeutet, das Template im Textbereich eines Webbrowsers zu bearbeiten, ist das ZMI leider auch die unangenehmste Methode, die man als Entwickler benutzen kann. Verglichen mit den meisten Editoren, bietet dieser Textbereich sehr wenige Funktionen. Es fehlen Eigenschaften wie Zeilennummern, Syntax-Hervorhebung usw. In Kapitel 10 zeige ich Ihnen, wie Sie mit dem External Editor Inhalte bearbeiten können. Damit können Sie den Inhalt der Website in lokalen Editoren wie Macromedia Dreamweaver oder Emacs bearbeiten. In Kapitel 6 zeige ich Ihnen, wie Sie Plone dazu bringen, Page Templates als Dateien von der Festplatte zu lesen, und dann können Sie ein beliebiges Werkzeug dafür benutzen.

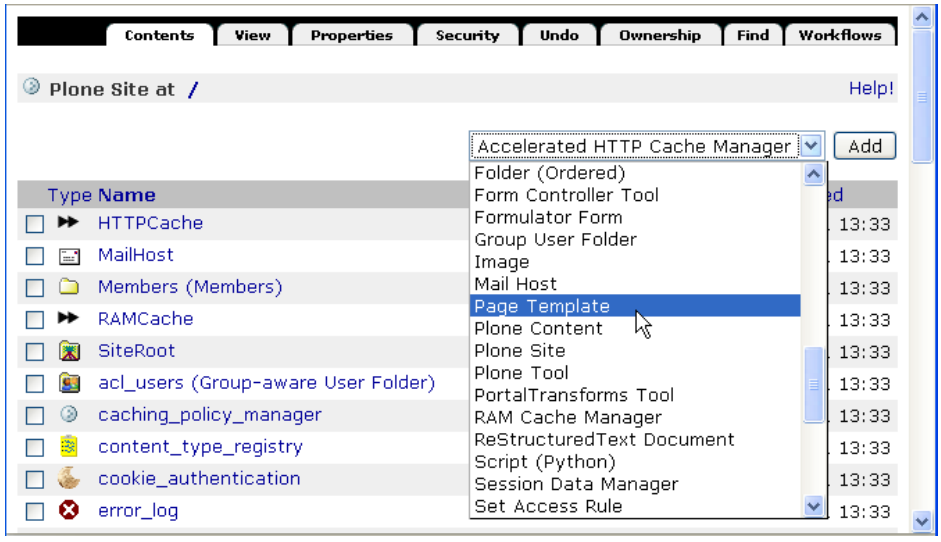


Abbildung 5.1: Wählen der Page Template-Option

Um ein Template zu erstellen, gehen Sie ins ZMI, klicken erst auf PORTAL_SKINS, dann auf CUSTOM und wählen dann PAGE TEMPLATE im Dropdown-Menü (siehe Abbildung 5.1). Klicken Sie auf ADD, und anschließend sehen Sie die Seite in Abbildung 5.2.

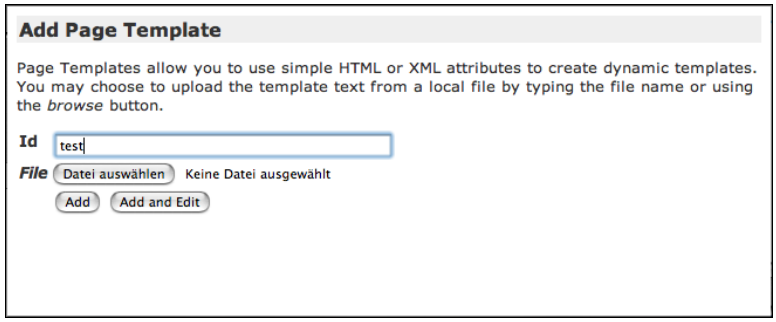


Abbildung 5.2: Hinzufügen eines Page Templates

Geben Sie **test** als ID des Page Templates ein. Klicken Sie dann auf ADD und auf den EDIT-Button, mit dem Sie zum Management-Schirm gelangen (siehe Abbildung 5.3). Dann können Sie dieses Template über das Web bearbeiten, indem Sie den Textbereich benutzen und auf SAVE CHANGES klicken, um die Änderungen zu speichern.

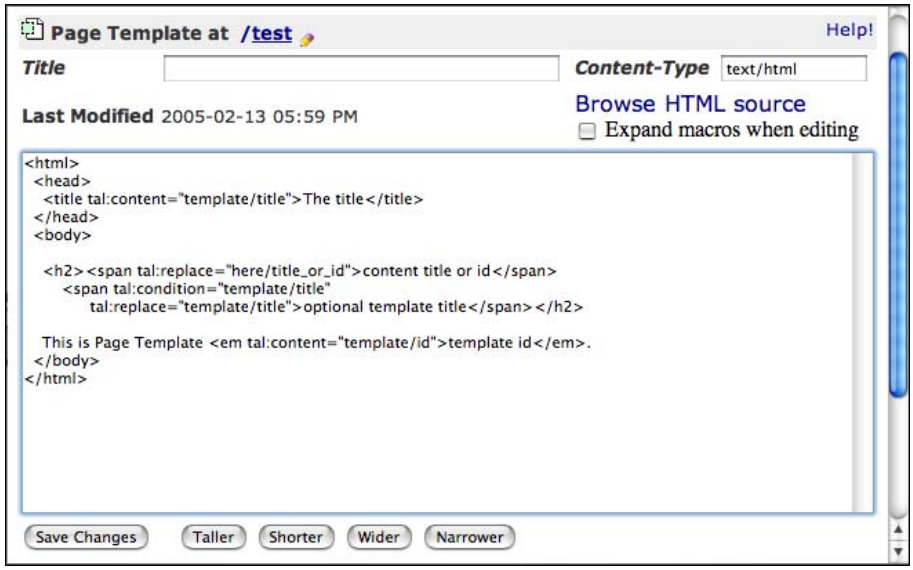


Abbildung 5.3: Bearbeiten eines Page Templates

Hinweis

Vor Zope 2.7 durchliefen alle Page Templates die Variable `here`, die äquivalent zu `context` ist. Wenn Sie `here` in irgendeinem Stück Code in einem Page Template sehen, bedeutet das `context`. Die neue Variable `context` wurde hinzugefügt, um klarer zu sein und die Page Templates mit Script(Python)-Objekten zu harmonisieren.

Nach einem Klick auf SAVE CHANGES wird das Page Template kompiliert. Falls Sie Fehler im Template gemacht haben sollten, sehen Sie diese oben auf der Seite hervorgehoben. Abbildung 5.4 zeigt einen Fehler bei einem `h1`-Tag an, das nicht geschlossen ist. (Wie zuvor erwähnt wurde, müssen Page Templates gültiges XHTML sein.)

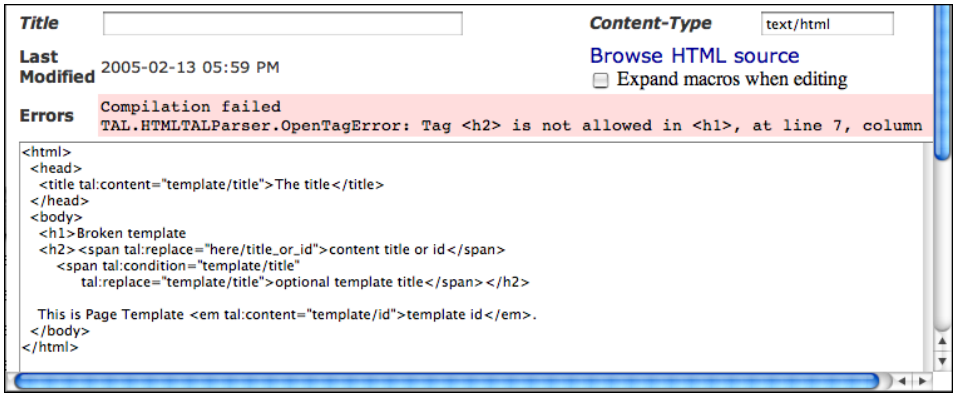


Abbildung 5.4: Page Template-Fehler

Nachdem Sie das Page Template erfolgreich gespeichert haben, können Sie auf den TEST-Reiter klicken, um den dargestellten Wert des Templates zu sehen. In Abbildung 5.5 sehen Sie, dass die Überschrift durch die ID des Templates ersetzt wurde und dass der Hauptabsatz nun die ID des Templates enthält.

Der Management-Bildschirm eines Page Templates enthält auch die folgenden wichtigen Eigenschaften:

- **Title**: Dies ist der Titel dieses Templates, der optional ist. Wenn Sie diesen im vorigen Beispiel ändern, z.B. nachdem Sie auf TEST geklickt haben, werden Sie feststellen, dass sich das HTML im Ergebnis verändert hat.
- **Content-Type**: Dies ist der Inhaltstyp dieses Templates, der normalerweise `text/html` ist.

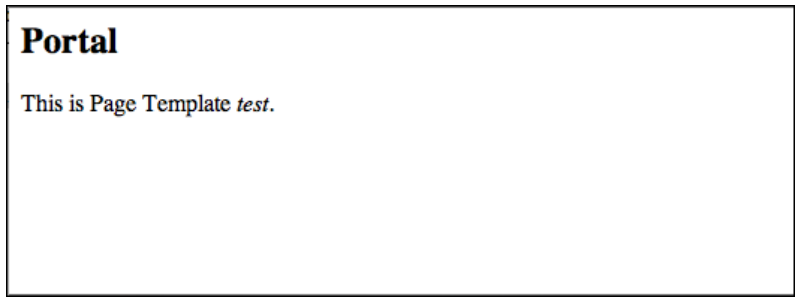


Abbildung 5.5: Erzeugen der Seite

- **Browse HTML source**: Hiermit wird das Template als unverarbeitetes HTML dargestellt. So würde das Template erscheinen, wenn es in einem HTML-Editor geladen würde.
- **Test**: Hiermit wird das Template verarbeitet und dargestellt.
- **Expand macros when editing**: Bei angekreuztem Kontrollkästchen wird versucht, Makros zu expandieren. Ich empfehle, es meistens nicht angekreuzt zu lassen. Makros sind eine Eigenschaft für Fortgeschrittene und werden in Kapitel 6 behandelt.

Nun, da Sie ein Page Template erzeugt haben, werden Sie ein paar Änderungen daran vornehmen. Dabei werden die Themen demonstriert, die bisher in diesem Kapitel behandelt worden sind. Wenn Sie z.B. möchten, dass Ihr Page Template 1+2 berechnet, könnten Sie folgende Zeile zu Ihrem Page Template hinzufügen:

```
<p>1+2 = <em tal:content="python: 1+2" /></p>
```

Klicken Sie dann auf den TEST-Reiter, um zu sehen, ob es funktioniert. Sie sollten Folgendes sehen:

```
1+2 = 3
```

Um ein Beispiel für eine Pfadtraversierung zu sehen, geben Sie das Logo Ihrer Plone-Site aus. Sie können einen Ausdruck im Logo Ihrer Plone-Site einfügen, indem Sie Folgendes zu Ihrem Page Template hinzufügen:

```
<p tal:replace="structure context/logo.jpg" />
```

Hierdurch wird das passende HTML für das Bild erzeugt und auf der Seite angezeigt.

5.3 Die Grundsyntax von Page Templates

Nachdem Sie nun gesehen haben, wie man ein Page Template erstellt, werde ich dessen grundlegende Syntax erklären. Die Syntax von Page Templates lässt sich in einige verschiedene Bestandteile gliedern, die ich in den folgenden Abschnitten behandeln werde.

5.3.1 Einführung in eingebaute Variablen

Die Syntax von Ausdrücken kennen Sie bereits, also werden Sie nun Variablen kennen lernen, die Ausdrücken übergeben werden, wenn Sie ein Page Template darstellen. Alles nun Folgende spielt sich im Kontext eines Zugriffs auf das Bild `Ein Bild.jpg` im Ordner `Members/andy` mit dem URL `/Members/andy/Ein Bild.jpg` ab:

- **container**: Dies ist der Container, in dem sich das Template befindet. Bei Plone ist das normalerweise der Ordner `portal_skins`. Sie sollten es vermeiden, einen Container zu verwenden, weil `portal_skins` unerwartete Änderungen am Container bewirken kann (z.B. eine Referenz auf den Ordner `andy`).
- **context**: Dies ist der Kontext, in dem das Template ausgeführt wird. In Plone ist das das angezeigte Objekt, falls Sie ein Portalobjekt anzeigen (z.B. eine Referenz auf das Objekt `Ein Bild.jpg`).
- **default**: Manche Anweisungen haben ein spezielles Standardverhalten. Darauf wird bei allen Anweisungen hingewiesen, und diese Variable ist ein Zeiger auf dieses Verhalten.
- **here**: Dies ist äquivalent zu `context`.
- **loop**: Dies ist äquivalent zu `repeat`.
- **modules**: Dies ist ein Container für importierte Module. So ist z.B. `modules/string/atoi` die Funktion `atoi` aus Pythons string-Modul. Hierunter fallen alle Module, die man gefahrlos ins Zope Page Templates-System importieren kann. Weitere Informationen finden Sie im Abschnitt »Plone mit Python scripten« in Kapitel 6.
- **nothing**: Dies ist das Äquivalent zu `None` in Python.
- **options**: Dies sind die Optionen, die ans Template übergeben werden, was dann passiert, wenn das Template von einem Script oder einer anderen Methode und nicht über das Web aufgerufen wird.
- **repeat**: Dies ist das wiederholte Element (siehe das Element `tal:repeat` im Abschnitt »Einführung in die Syntax von TAL-Anweisungen« in diesem Kapitel).

- **request**: Dies ist die eingehende Anfrage des Clients (alle Werte der eingehenden Anfrage sind mit Hilfe des folgenden Kontext-Testscripts sichtbar). Alle `GET`- und `POST`-Parameter werden zwecks leichteren Zugriffs mit dem Python-Modul `marshall` in ein Dictionary serialisiert. Hier sind einige Beispiele:

```
request/HTTP_USER_AGENT # der Browser des Benutzers
request/REMOTE_ADDRR  # der Browser des Benutzers
request/someMessage   # der Wert einer Meldung im Abfrage-String
```

- **root**: Dies ist das Zope-Wurzelobjekt. Mit `root/Control_Panel` erhalten Sie beispielsweise das Control Panel für Zope.
- **template**: Dies ist das aufgerufene Template. Mit `template/id` erhalten Sie z.B. die ID des dargestellten Templates.
- **traverse_subpath**: Dies enthält eine Liste von Elementen, die noch traversiert werden müssen. Es handelt sich um eine Variable für Fortgeschrittene, die Sie besser erst dann verwenden sollten, wenn Sie Traversierung und Akquisition verstanden haben.
- **user**: Dies ist das aktuelle Benutzerobjekt. `user/getUserName` ist der Benutzername des aktuellen Benutzers.
- **CONTEXTS**: Dies ist eine Liste mit den meisten dieser Werte.

Hinweis

Mit Ausnahme von `CONTEXTS` können all diese Variablen in einer `tal:define`-Anweisung neu definiert werden, wenn der Benutzer das möchte. Das ist allerdings nicht ratsam, weil das für jemanden, der den Code verwendet, recht verwirrend sein kann.

Das Page Template `test_context` zeigt die Werte all dieser Variablen sowie den Ort einiger Objekte (siehe Listing 5.1). Es kann hilfreich bei der Fehlersuche und beim Bestimmen von Variablenwerten sein. Fügen Sie es als Page Template namens `test_context` hinzu, und klicken Sie dann auf TEST, um die Ergebnisse zu sehen.

Listing 5.1: test_context

```
<html>
  <head />
  <body>
    <h1>Debug information</h1>
  <h2>CONTEXTS</h2>
```

```
  <ul>
    <tal:block
        tal:repeat="item CONTEXTS">
    <li
        tal:condition="python: item != 'request'"
        tal:define="context CONTEXTS;">
            <b tal:content="item" />
            <span tal:replace="python: context[item]" />
    </li>
    </tal:block>
  </ul>
  <h2>REQUEST</h2>
  <p tal:replace="structure request" />
  </body>
</html>
```

Das Page Template `test_context` produziert die Ausgabe, die Sie in Abbildung 5.6 sehen.

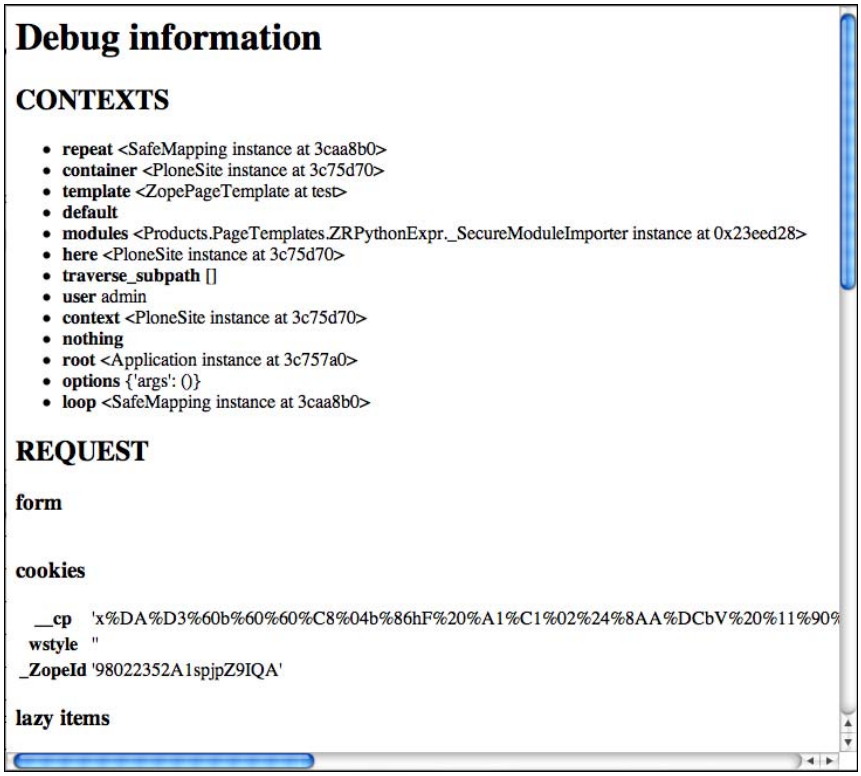


Abbildung 5.6: Ein Beispiel für alle Standardvariablen in einem Script

5.3.2 Einführung in die Syntax von TAL-Anweisungen

Die Sprache TAL (Template Attribute Language) bietet alle grundlegenden Bausteine für eine dynamische Präsentation. TAL definiert acht Anweisungen: `attributes`, `condition`, `content`, `define`, `omit-tag`, `on-error`, `repeat` und `replace`.

Da Page Templates gültiges XML sind, müssen alle TAL-Attribute in Kleinbuchstaben geschrieben werden. Außerdem darf jedes Element eine Anweisung nur einmal enthalten. In den folgenden Beispielen habe ich in den Elementen Zeilenumbrüche eingefügt, um die Lesbarkeit zu verbessern. Das ist absolut gültiger Code und kommt recht häufig im Plone-Quellcode vor. Dennoch ist es optional und wird nicht verlangt.

tal:attributes: Element-Attribute ändern

Mit `tal:attributes` können Sie ein oder mehrere Attribute eines Elements ersetzen. Eine Anweisung enthält die zu ändernden Attribute, die mit einem Leerzeichen von der Anweisung getrennt werden. Beispiel:

```
<a href="#"
   tal:attributes="href context/absolute_url">
   Link hierher
</a>
```

Hierbei wird das Attribut `href` des Links auf das Ergebis von `here/absolute_url` geändert. Das Attribut `href` wurde für dieses Element bereits definiert, d.h., wenn ein Design-Programm diese Seite öffnet, sieht es ein gültiges Element (auch, wenn der Link vielleicht so lange keinen Sinn macht, bis die Seite verarbeitet worden ist). Es folgen einige Beispielausgaben:

```
<a href="http://plone.org/Members/andy/book">Link hierher</a>
```

Da jedes Element mehrere Attribute haben kann, erlaubt `tal:attributes` die Änderung eines oder mehrerer Attribute gleichzeitig, indem mehrere Anweisungen verwendet werden. Um mehrere Attribute gleichzeitig zu ändern, trennen Sie die Anweisungen mit einem Semikolon (;). Falls das Attribut oder die Anweisung ein Semikolon enthält, können Sie dieses mit einem weiteren Semikolon direkt davor schützen (;;). Tun Sie Folgendes, um beispielsweise `href` und `title` im Element zu ändern:

```
<a href="#"
   tal:attributes="href context/absolute_url;
      title context/title_or_id">Link</a>
```

Die Ausgabe dieses Beispiels ist folgende:

```
<a href="http://plone.org/Members/andy/book" title="Plone Book">Link</a>
```

Die Tags `tal:attributes` und `tal:replace` schließen einander aus, weil `replace` das Element entfernt. Das Zope Page Templates-System stellt fest, wenn beide verwendet werden, gibt dann eine Warnung aus und ignoriert den `tal:attributes`-Teil. Falls der Ausdruck zu `default` ausgewertet wird, wird keine Änderung vorgenommen. Beispiel:

```
<a href="#"
    tal:attributes="href
        python:request.get('message', default)">
    Link</a>
```

In diesem Beispiel verwende ich die Funktion `get` auf dem Objekt `request`. Falls die eingehende Anfrage nach der Seite die Variable `message` enthält, wird der erste Wert verwendet, d.h. `message`. Wenn die Variable `message` nicht vorhanden ist, wird der zweite Wert, `default`, benutzt. Daher erfolgt allein durch die Übergabe des Parameters `message` eine Änderung.

tal:condition: Bedingungen auswerten

Mit der Anweisung `tal:condition` kann eine Bedingung getestet werden, bevor das Element dargestellt wird. Beispiel:

```
<p tal:condition="request/message">
    Es gibt eine Message
</p>
<p tal:condition="not: request/message">
    Keine Message
</p>
```

Hierbei wird der Absatz mit dem Text `message` nur dann dargestellt, wenn die Variable `request` ein Attribut hat und dieses zu `True` ausgewertet wird. Der Test einer Bedingung ist jedoch sinnlos, falls das Gegenteil der Bedingung nicht getestet werden kann. Dies leistet der negierte Ausdruck. Das Präfix `not:` invertiert die Anweisung, d.h., `not: request/message` wird dann zu True ausgewertet, falls die Anfragevariable `message` zu `False` ausgewertet wird.

In TAL wird Folgendes zu `False` ausgewertet:

- Die Zahl Null
- Alle Fließkomma- oder komplexen Zahlen, die als null ausgewertet werden (z.B. `0.0`)
- Strings der Länge null (z.B. `""`)
- Eine leere Liste oder ein leeres Tupel
- Ein leeres Dictionary

- Der Wert `None` in Python
- Der Wert `nothing` in TALES

Folgendes wird zu `True` ausgewertet:

- Der Wert `default`
- Alle Zahlen, die von null verschieden sind
- Strings, die nicht leer sind
- Strings, die nicht nur aus Leerzeichen bestehen (z.B. `"  "`)
- Alles andere

tal:content: Text hinzufügen

Die Anweisung `tal:content` in einem Page Template ist die wahrscheinlich am häufigsten verwendete Anweisung. Sie ist auch eine der einfachsten, da sie den Inhalt eines Elements durch den angegebenen Wert ersetzt. Beispiel:

```
<i tal:content="context/title_or_id">Ein Titel</i>
```

Die Ausgabe des Beispiels lautet:

```
<i>Ein Titel</i>
```

Hierbei wird der Text `Ein Titel` durch den Wert des Ausdrucks `context/title_or_id` ersetzt. Falls der zu ersetzende Text HTML-Elemente enthält, werden diese geschützt. Der zu ersetzende Text wird standardmäßig HTML-geschützt. Das Präfix `structure` erlaubt, dass HTML eingegeben wird, ohne dass die Elemente geschützt werden. Beispiel:

```
<i tal:content="structure here/title_or_id">HTML nicht schützen</i>
```

Falls das Element mit den Attributen `tal:content` weitere Elemente enthält, werden auch diese Elemente alle ersetzt. Die Tags `tal:content` und `tal:replace` schließen sich gegenseitig aus und können nicht beide im gleichen Element vorkommen, ohne dass ein Fehler dabei auftritt. Falls der Wert `default` lautet, bleibt der Inhalt unverändert.

tal:define: Variablen definieren

Mit der Anweisung `tal:define` können im Template Variablen erzeugt und wiederverwendet werden. Beispiel:

```
<p tal:define="title here/title_or_id">
    ... <i tal:content="title">Ein Titel</i> ...
</p>
```

In diesem Beispiel wird die Variable `title` erzeugt, und es wird ihr das Ergebnis von `here/title_or_id` zugewiesen. Später wird die Variable in der Anweisung `tal:content` benutzt. Standardmäßig wird diese Variable nur im lokalen Geltungsbereich des aktuellen Elements erzeugt. Im vorigen Beispiel können daher nur Elemente im Absatz die Variable `title` benutzen. Sie können die Variable an einer beliebigen Stelle in der Anweisung neu definieren oder in anderen Elementen so oft wiederverwenden, wie Sie möchten.

Um eine global benutzbare Variable zu erstellen, verwenden Sie das Präfix `global`. Dadurch wird der Zugriff auf die Variable von überall im Template ermöglicht, nicht nur aus dem definierenden Element. Beispiel:

```
<p tal:define="global title string:Foo bar">
    ... <i tal:content="title">Ein Titel</i> ...
</p>
<i tal:content="title">Immer noch ein Titel</i>
```

Plone bietet eine große Anzahl solcher globaler Definitionen, damit die Benutzer diese in eigenen Scripts einfach verwenden können. Wie bei allen solchen Definitionen gilt, dass sie sich in Zukunft verändern können, d.h., Sie sollten sie mit Bedacht einsetzen. Diese `define`-Anweisungen bedeuten, dass eine große Menge an globalen Variablen verfügbar sind. Um beispielsweise an den Titel Ihrer Plone-Site zu gelangen, können Sie einfach Folgendes aufrufen:

```
<p tal:content="portal_title" />
```

Diese `define`-Anweisungen finden Sie im ZMI, wenn Sie erst auf PORTAL_SKINS klicken, dann auf PLONE_TEMPLATES und schließlich auf GLOBAL_DEFINES. Eine vollständige Liste aller Definitionen samt Erklärung finden Sie in Anhang A.

tal:omit-tag: Elemente entfernen

Die Anweisung `tal:omit-tag` ist etwas ungewöhnlich. Mit ihr kann ein Tag entfernt werden. Da das Zope Page Templates-System den Gebrauch von HTML-Tags vorschreibt, benötigen komplizierte Seiten oftmals eine Menge Elemente und können dazu führen, dass Extra-Tags hinzugefügt werden. Bei dieser Anweisung wird das Tag entfernt, so dass nur sein Inhalt stehen bleibt. Beispiel:

```
<p tal:omit-tag="">Irgendein Text</p>
```

Die Ausgabe lautet:

```
Irgendein Text
```

In diesem Beispiel wird der Text `Irgendein Text` ausgegeben, aber der Tag drumherum wird nicht dargestellt. Die Anweisung `tal:omit-tag` hat einen Ausdruck

als optionales Argument. Wenn dieser zu `False` ausgewertet wird, findet die Auslassung nicht statt. Folgende Anweisung z.B. tut nichts:

```
<p tal:omit-tag="nothing">Irgendein Text</p>
```

Eine Alternative zum Einsatz von `tal:omit-tag` ist der eines `tal`-Namespace, wie im Abschnitt »Nützliche Hinweise« in Kapitel 6 beschrieben wird.

tal:on-error: Fehler behandeln

Die Anweisung `tal:on-error` bietet die Möglichkeit, Fehler zu behandeln. Sie verhält sich eher wie `tal:content`, weil sie bewirkt, dass der Inhalt des Tags ersetzt wird. Aber sie wird nur dann ausgeführt, wenn ein Fehler auftritt.

Hier ist ein Beispiel:

```
<p  tal:content="request/message"
    tal:on-error="string: Keine Message">Message</p>
```

Wenn es hier einen Fehler bei der Auswertung des Ausdrucks `request/message` gibt, dann wird das Attribut `on-error` aktiviert. Es bewirkt, dass der Inhalt des Tags durch den Text `Keine Message` ersetzt wird.

Leider ist die Anweisung `on-error` sehr beschränkt. Das Tag kann nicht zwischen verschiedenen Fehlern unterscheiden und erlaubt nur einen einzigen Ausdruck, der ausgewertet und benutzt werden kann. Diese Einschränkung ist gewollt, damit das Tag nicht übermäßig verwendet wird. Eine Fehlerbehandlung ist wirklich sehr viel besser in der Logik Ihrer Anwendung aufgehoben.

Zum Glück können Sie bei allen Ausdrücken Alternativen in der Anweisung angeben, falls der erste Anweisungsteil weder zu `True` noch zu `False` ausgewertet wird, d.h., wenn ein Fehler auftritt. Alle Alternativen werden durch einen senkrechten Strich (`|`) voneinander getrennt, und es können mehrere Alternativen in einer Anweisung vorkommen. Wenn Sie sich auf Variablen in eintreffenden Anfragen verlassen, sollten Sie am Ende immer `|nothing` hinzufügen, um sicherzugehen, dass kein Attributfehler auftritt.

Beispiel:

```
<p
  tal:content="request/message"
  tal:condition="request/message|nothing">
    Es gibt eine Message
</p>
<p tal:condition="not: request/message|nothing">
    Keine Message
</p>
```

Dieses zweite Beispiel ist etwas wortreicher, aber aus mehreren Gründen empfehlenswert:

- Der Designer sieht die positive *und* die negative Bedingung.
- Sie können eine kompliziertere Fehlerbedingung behandeln, anstatt nur einen String auszugeben.

tal:repeat: Schleifen ausführen

Die Anweisung `tal:repeat` erlaubt Schleifendurchgänge über Objekte und ist eine der komplizierteren Anweisungen. Eine Schleife enthält den Wert, der bei jeder Iteration über die Ergebnisse zugewiesen werden soll. Dieser Wert ist durch ein Leerzeichen von den Ergebnissen getrennt, über die iteriert wird.

Hier sehen Sie ein Beispiel für Schleifen:

```
<table>
  <tr tal:repeat="row context/portal_catalog">
    <td tal:content="row/Title">Title</td>
  </tr>
</table>
```

In diesem Beispiel gibt der Ausdruck `here/portal_catalog` eine Liste von Ergebnissen zurück. Da die Wiederholung beim `row`-Tag der Tabelle beginnt, wird für jede Zeile in der Ergebnisliste eine neue Zeile in der Tabelle erzeugt. Wie bei `tal:define` findet in jeder Iteration über die Ergebnisse eine Zuweisung an eine lokale Variable, in diesem Fall `row`, statt. Dieses Beispiel gibt für jeden Eintrag in der Ergebnisliste eine Zeile aus.

In der `repeat`-Anweisung können Sie auf einige nützliche Variablen wie die laufende Nummer der aktuellen Iteration zugreifen. Darauf können Sie über die Variable `repeat` zugreifen, die zum Namespace hinzugefügt wird. Um beispielsweise auf die laufende Nummer zuzugreifen, benutzen Sie Folgendes:

```
<table>
  <tr tal:repeat="row context/portal_catalog">
    <td tal:content="repeat/row/number">1</td>
    <td tal:content="row/Title">Title</td>
  </tr>
</table>
```

Die vollständige Liste aller in `repeat` verfügbaren Variablen lautet:

- **index**: Dies ist die laufende Nummer, beginnend bei null.
- **number**: Dies ist die laufende Nummer, beginnend bei eins.
- **even**: Dies ist `True` bei geradem Iterationsindex (`0`, `2`, `4`, ...).

- **odd**: Dies ist `True` bei ungeradem Iterationsindex (`1`, `3`, `5`, ...).
- **start**: Dies ist `True` bei der ersten Iteration.
- **end**: Dies ist `True` bei der letzten Iteration.
- **length**: Dies ist die Gesamtzahl an Iterationen.
- **letter**: Dies ist die Iterationszahl als Kleinbuchstabe (`a-z`, `aa-az`, `ba-bz`, ..., `za-zz`, `aaa-aaz`, usw.).
- **Letter**: Dies ist eine Großbuchstabenversion von letter.
- **roman**: Dies ist die Zahl mit römischen Ziffern in Kleinbuchstaben (`i`, `ii`, `iii`, `iv`, `v`, usw.), beginnend bei eins.

Im `repeat`-Namespace sind zwei weitere Werte verfügbar, die recht ungewöhnlich sind und selten verwendet werden, `first` und `last`. Mit diesen beiden Variablen können Sie Informationen über die Daten der Iteration speichern. Durch die Verwendung des Werts, den Sie in einem Ausdruck speichern möchten, wird ein boolescher Wert zurückgegeben. Bei der Variablen `first` gibt `True` an, dass dieser Wert zum ersten Mal in der Iteration aufgetreten ist. Entsprechend gilt bei der Variablen `last`, dass `True` anzeigt, dass der Wert zum letzten Mal in der Iteration aufgetreten ist.

Hier ist ein Beispiel dafür:

```
<ul>
  <li tal:repeat="val context/objectValues">
    First: <i tal:content="repeat/val/first/meta_type" />,
    Last: <i tal:content="repeat/val/last/meta_type" />:
    <b tal:content="val/meta_type" />,
    <b tal:content="val/title_or_id" />
  </li>
</ul>
```

tal:replace: Text hinzufügen

Die Anweisung `tal:replace` ähnelt `tal:content`, mit dem Unterschied, dass das gesamte Tag entfernt wird.

Beispiel:

```
<p tal:replace="context/title_or_id">Ein Titel</p>
```

Hiermit wird das Ergebnis des Ausdrucks `context/title_or_id` angezeigt, aber die Absatz-Tags werden aus dem Ergebnis entfernt. Das ist gleichbedeutend mit dem Folgenden:

```
<p
  tal:content="here/title_or_id"
  tal:omit-tag="">Ein Titel</p>
```

Falls das Element mit der Anweisung `tal:replace` weitere Elemente enthält, werden diese Elemente alle ersetzt. Die Anweisung `tal:replace` können Sie nicht zusammen mit `tal:attributes` oder `tal:content` verwenden, da sie sich gegenseitig ausschließen und ein Fehler auftritt, wenn Sie beide im gleichen Element benutzen.

5.3.3 Einführung in die Ausführungsreihenfolge

Die Reihenfolge, in der TAL-Attribute geschrieben werden, ist nicht die gleiche, in der sie ausgeführt werden, weil sie Teil von XML-Elementen sind (und XML kümmert sich nicht um die Reihenfolge von Attributen). Die Attribute werden in der folgenden Reihenfolge ausgeführt:

1. `define`
2. `condition`
3. `repeat`
4. `content`
5. `replace`
6. `attributes`
7. `omit-tag`

Die Anweisungen `content` und `replace` können Sie nicht im gleichen Element verwenden, da sie sich gegenseitig ausschließen. Die Anweisung `attributes` zusammen mit `replace` oder `omit-tag` ist sinnlos in einem Element, weil die Attribute entfernt werden. Das `on-error`-Tag wird nicht erwähnt, weil es verwendet wird, wenn der erste Fehler in irgendeinem der vorigen Elemente auftritt.

5.3.4 Beispiel: Benutzerinformationen anzeigen

Damit Sie sich die bisher gelernten Punkte veranschaulichen können, werden Sie nun ein Page Template erstellen, das eine einfache Aufgabe hat: Informationen über einen Benutzer im System anzuzeigen.

In diesem Beispiel verwendet eine Firma Plone intern in einem Intranet. Jeder Angestellte ist in Plone registriert und kann sich anmelden. Es gibt jedoch keine einfache Seite, die alle Angestellten anzeigt oder darüber informiert, wie man sie kontaktieren kann. Sie werden eine einfache Informationsseite für die Benutzer erstellen, die für alle Benutzer deren E-Mail-Adresse, Homepage und Bild anzeigt und die angibt, wann sie sich das letzte Mal angemeldet haben.

Ein erster Prototyp dieser Seite ist mit TAL, TALES und ein wenig Grundlagenwissen über CMF-Werkzeuge (Content-Management-Framework) schnell erstellt. Weil die APIs (Application Programming Interfaces) zu diesen Werkzeugen aber leider sehr verworren sind, ist ein Teil dieses Codes länger, als er sein sollte. Seien Sie erst einmal unbesorgt wegen der API dieser Werkzeuge, sie werden in Kapitel 9 behandelt. Wenn Sie das API erst einmal als gegeben hinnehmen, können Sie sich auf TAL konzentrieren.

Zuerst müssen Sie ein Page Template erstellen. Klicken Sie also auf PORTAL_SKINS, dann auf CUSTOM, und fügen Sie ein Page Template mit der ID **user_info** hinzu. Als Zweites werden Sie es wie folgt bearbeiten. Ein vollständiges Listing diese Page Templates finden Sie in Anhang A. Wenn Sie dieses Listing untersuchen, sehen Sie, dass es mit `HTML`- und `body`-Tags beginnt.

Aus Gründen der Bequemlichkeit setzen Sie die Hauptdefinitionen in ein `div`-Tag:

```
<div
  tal:omit-tag=""
  tal:define="
    userName request/userName|nothing;
    userObj python: here.portal_membership.getMemberById(userName);
    getPortrait nocall: here/portal_membership/getPersonalPortrait;
    getFolder nocall: here/portal_membership/getHomeFolder
    ">
```

In diesem `div`-Tag gibt es vier `define`-Anweisungen: eine, um an den Benutzernamen zu gelangen, der im `request`-Objekt übergeben wird, und eine weitere, um diesen Benutzernamen in ein Benutzerobjekt zu übersetzen. Die letzten beiden `define`-Anweisungen stellen sicher, dass Sie eine gültige Referenz auf die Methoden haben, mit denen Sie an die Bilder und Ordner der Benutzer gelangen. Auch diese sind bequem, weil sie den Code später vereinfachen. Ein solches `div`-Tag oder auch anderes Tag einzurichten, das eine Reihe von Definitionen enthält, ist ein übliches Vorgehen beim Zope Page Templates-System. Der Code wird dadurch einfach sauberer.

Als Nächstes erstellen Sie zwei einfache Bedingungen, um zu prüfen, ob Sie einen Benutzer haben:

```
<p tal:condition="not: userName">
    Keinen Benutzernamen gewählt.
</p>
<p tal:condition="not: userObj">
    Dieser Benutzername existiert nicht.
</p>
```

Wenn die Anfrage keinen Benutzernamen enthält, resultiert der Ausdruck `request/username|nothing` darin, dass `userName` gleich `nothing` ist, und der einfache Test schlägt fehl. Wenn der Benutzername ungültig ist, wird `userObj` zu `None`, und es werden Fehlermeldungen für diese beiden Bedingungen ausgegeben.

Nun sind Sie so weit, dass Sie den Benutzer bearbeiten können:

```
<table tal:condition="userObj">
  <tr>
    <td>
      <img src=""
      tal:replace="structure python: getPortrait(userName)" />
    </td>
```

Da Sie einen Benutzer nur dann anzeigen können, wenn Sie einen gefunden haben, werden Sie sicherstellen, dass es eine einfache Bedingung für diese Tabelle `tal:condition="userObj"` gibt. Um das Bild eines Benutzers anzuzeigen, verwenden Sie die zuvor definierte Methode `getPortrait`. Diese Funktion gibt das gesamte Tag zurück, d.h., das `structure`-Tag stellt sicher, dass das ganze Bild korrekt dargestellt wird. Als Nächstes möchten Sie einige Eigenschaften wie `name` und `email` anzeigen. Im Folgenden wird dies für eine dieser Optionen, den `home`-Ordner, gezeigt:

```
<li
    tal:define="home python: getFolder(userName)"
    tal:condition="home">
    <a href=""
        tal:attributes="href home/absolute_url"
        >Home-Ordner</a>
</li>
```

Zuerst verwenden Sie ein `define`, um an den Ordner zu gelangen, und weisen ihn der Variablen `home` zu. In einer Plone-Site ist das Erstellen eines `home`-Ordners für einen Benutzer optional, d.h., beim Verweis auf einen Ordner müssen Sie sicher sein, dass dieser auch existiert. Wegen der Ausführungsreihenfolge von TAL kommt die Definition vor der Bedingung. Danach zeigen Sie einen Link auf den Ordner mit Hilfe des Ordnerattributs `absolute_url` an.

Dieses Page Template enthält ein paar weitere Zeilen, um einige andere nützliche und aufregende Eigenschaften zu finden, die es dem Benutzer zeigen kann. Wie bei den meisten Dingen in Plone kommt es darauf an, die korrekten API-Aufrufe zu finden und die Ausgabe entsprechend weiterzuverarbeiten.

Die Seite endet schließlich mit dem Schließen aller relevanten Tags. Wenn alles gut geht, sollten Sie die Seite aufrufen können, indem Sie auf den URL *http://ihre-site/user_info?userName=[einbenutzer*] zugreifen, wobei `einbenutzer` ein vorhandener Benutzername in Ihrer Plone-Site ist.

Momentan ist dieses Page Template ziemlich beschränkt. Nur ein Benutzer mit Managerrechten kann diese Seite sehen, sie kann nur jeweils ein Mitglied zu einem Zeitpunkt anzeigen, und die Information über diesen Benutzer ist recht mager. In Kapitel 6 werde ich zeigen, wie Sie dieses Beispiel ausbauen und einiges an Wiederverwendbarkeit von Komponenten hinzufügen – sowie die Möglichkeit, den Text in andere Sprachen zu übersetzen.

6 Einführung in Plone-Templating und -Scripting für Fortgeschrittene

Das vorangegangene Kapitel behandelte die Funktionsweise des Zope Page Templates-Systems. Um Ihnen Page Templates nahe zu bringen, habe ich in Kapitel 5 auch die Objekthierarchie, Akquisition und TALES (Template Attribute Language Expression Syntax) behandelt. Mit dem dortigen Code konnten Sie dynamische Webseiten erzeugen. In dem Kapitel haben Sie ein Beispiel-Page-Template gesehen, das den Code zusammengesetzt hat. Außerdem wurden darin die Bestandteile des Templating-Systems in Plone behandelt, d.h., Sie haben die Schlüsselinformationen erhalten, die Sie brauchen, um Plone zu benutzen.

Nun ist es Zeit, zu einigen der weitergehenden Eigenschaften von Page Templates und allgemein zum Templating in Plone überzugehen. Zuerst werde ich dazu METAL (Macro Expansion Template Attribute Language) und Internationalisierungs-(I18N-)Namespaces einführen. Wie der TAL-Namespace bieten diese dem Site-Entwickler einiges an Funktionalität. Diejenigen, die ganz genau wissen möchten, wie eine Plone-Seite zusammengesetzt wird, finden im Abschnitt »Sich mit METAL in Plone einklinken« viele Antworten auf ihre Fragen.

Bisher habe ich gezeigt, wie Sie einfache Python-Ausdrücke in Page Templates verwenden können. Natürlich ist manchmal ein einzeiliger Python-Ausdruck nicht ausreichend. Im Abschnitt »Plone mit Python scripten« werde ich zeigen, dass Sie Python eine Ebene höher einsetzen können und Ihre Scripten damit wesentlich mächtiger machen können.

Abschließend werde ich ein häufig vorkommendes Beispiel behandeln, das zeigt, wie man ein Formular in Plone zusammenbaut. Dieses Beispiel demonstriert Konzepte, die in den vorherigen Kapiteln vermittelt wurden, und kombiniert alles miteinander, während Sie genau sehen können, wie Plone mit Formularen umgeht.

6.1 Hintergrund zu fortgeschrittenem Plone Templating

Eine der hübschen Eigenschaften von Page Templates ist die, dass verschiedene Funktionen klar in verschiedenen Namespaces voneinander getrennt sind. Im vorigen Kapitel haben Sie die TAL-Namespaces gesehen. Das ist nicht der einzige Namespace, den Page Templates zur Verfügung stellen. Zwei weitere Namespaces sind sehr wichtig für Plone.

Der erste ist METAL. Wie aus dem ziemlich langen Namen hervorgeht, ist er insofern ähnlich zu TAL, als er eine Attributsprache ist und sich selbst in Elementattribute einfügt. Sein Hauptzweck ist jedoch sicherzustellen, dass Sie Codeteile von anderen Page Templates wiederverwenden können, und zwar mit Hilfe von Slots und Makro-Funktionen.

Der zweite ist I18N, mit dem Sie den Inhalt von Page Templates übersetzen können. In Plone wird das dazu verwendet, die Schnittstelle von Plone in über 30 Sprachen zu lokalisieren, was für viele Benutzer eines der Schlüsselmerkmale von Plone ist. Wie Sie sehen werden, ist die Möglichkeit, Texte zu lokalisieren, für alle Benutzer von Interesse, auch für jene, die eine einsprachige Site bauen. Beginnen wir mit METAL.

6.1.1 Sich mit METAL in Plone einklinken

Bisher haben Sie gesehen, wie Sie Teile von Webseiten mit TAL dynamisch generieren können. Damit können Sie allerdings nicht wirklich komplexes Templating betreiben. Es gibt einfach keinen Mechanismus, um oben auf jede Seite einen Standardkopf zu setzen, außer eine TAL-Anweisung zu benutzen. METAL ist eine Methode zur Vorverarbeitung der Templates und bietet einige mächtigere Funktionen als TAL. Alle METAL-Funktionen beginnen mit dem Präfix `metal:`.

metal:define-macro

Mit dem Befehl `metal:define-macro` können Sie ein Element so definieren, dass es von einem anderen Template referenziert werden kann. Der Name des referenzierten Teils ist der Name des Makros. Es folgt ein Beispiel, das `boxA` als etwas definiert, das Sie anderswo benutzen möchten:

```
<div metal:define-macro="boxA">
    ...
</div>
```

Das `div`-Element ist nun ein Makro, das aus anderen Templates referenziert werden kann. Das Makro bezieht sich nur auf den Teil der Seite, der von dem Ele-

ment referenziert wird, was in diesem Fall der `div`-Tag ist. Es ist üblich, auf einer Seite mehrfach `macro:defines` zu verwenden, auch, damit die Seite eine gültige HTML-Seite ist, z.B. so:

```
<html xmlns:tal="http://xml.zope.org/namespaces/tal"
    xmlns:metal="http://xml.zope.org/namespaces/metal"
    i18n:domain="plone">
    <body>
        <div metal:define-macro="boxA">
            ...
        </div>
        <div metal:define-macro="boxB">
            ...
        </div>
    </body>
</html>
```

Übereinstimmend mit den früheren Zielen von Page Templates ist diese Seite eine gültige HTML-Seite, die von einem Designer bearbeitet werden kann. Wenn das Makro aufgerufen wird, wird das HTML außerhalb der `div`-Tags weggeworfen.

metal:use-macro

Der Befehl `metal:use-macro` verwendet ein Makro, das mit `define-macro` definiert wird. Wenn ein Template ein Makro mit dem Befehl `define-macro` definiert, können andere Templates mit dem `macros`-Property darauf zugreifen. Wenn Sie z.B. das Makro `portlet` aus dem Template `portlet_login` verwenden möchten, können Sie Folgendes machen:

```
<div metal:use-macro="context/portlet_login/macros/portlet">
    Der Einloggen-Slot kommt hierhin
</div>
```

Damit wird das Makro geholt und sein Ergebnis an dessen Stelle eingefügt. Wie Sie sehen, erwartet der Befehl `use-macro` einen Pfadausdruck, der auf das Template und auf das spezielle Makro darin zeigt.

Beispiel: Die Makros use-macro und define-macro verwenden

Folgendes Template namens `time_template` ist ein Beispiel hierfür. Dieses Template zeigt Datum und Zeit auf dem aktuellen Plone-Server an. Das ist eine recht nützliche Funktion, d.h., Sie können sie in ein Makro zur weiteren Wiederverwendung verpacken. Das Folgende ist ein Beispiel-Page-Template mit der Anweisung `define-macro`:

```
<html>
    <body>
        <div metal:define-macro="time">
            <div tal:content="context/ZopeTime">
                 Die Zeit
            </div>
        </div>
    </body>
</html>
```

Falls Ihr Template `time_template` heißt, können Sie dieses Makro in einem anderen Template referenzieren. Sie können es auch aus mehreren Templates referenzieren. Hier ist ein solches Beispiel-Template:

```
<html>
    <body>
    <div metal:use-macro="context/time_template/macros/time">
      Falls es eine Message gibt, zeigt das Makro sie hier an.
    </div>
    </body>
</html>
```

Wenn dieses Template dargestellt wird, sieht der von Plone generierte HTML-Code dafür wie folgt aus:

```
<html>
    <body>
    <div>
       <div>2004/04/15 17:18:18.312 GMT-7</div>
    </div>
    </body>
</html>
```

metal:define-slot

Ein *Slot* ist ein Makro-Abschnitt, von dem der Autor des Templates erwartet, dass er von einem anderen Template überschrieben wird. Sie können sich einen Slot als Loch in Ihrem Page Template vorstellen, bei dem Sie davon ausgehen, dass es von einem anderen gefüllt wird. Alle `define-slot`-Befehle müssen in einem `define-macro` vorkommen. Beispiel:

```
<div metal:define-macro="master">
  <div metal:define-slot="main">
  ...
  </div>
</div>
```

metal:fill-slot

Hiermit wird ein Slot gefüllt, der mit dem Befehl `define-slot` definiert wurde. Ein `fill-slot` muss mit dem Befehl `use-macro` definiert werden. Wenn der `define-macro`-Teil aufgerufen wird, versucht das Makro, alle definierten Slots mit dem entsprechenden `fill-slots` zu füllen. Hier sehen Sie ein Beispiel für einen `fill-slot`:

```
<div metal:use-macro="master">
  <div metal:fill-slot="main">
    Der Haupt-Slot kommt hierhin
  </div>
</div>
```

Beispiel: Makros und Slots verwenden

Kommen wir zum vorigen Beispiel zurück, das Sie nun ein wenig verbessern werden. Wenn Sie eine spezielle Meldung vor die Zeitangabe setzen möchten, würden Sie einen Slot am Anfang von `time_template` innerhalb von `define-macro` hinzufügen. Der Slot heißt `time` und sieht wie folgt aus:

```
<html>
    <body>
        <div metal:define-macro="time">
            <div metal:define-slot="msg">Time slot</div>
            <div tal:content="context/ZopeTime">
                 Die Zeit
            </div>
        </div>
    </body>
</html>
```

Im aufrufenden Page Template können Sie nun `fill-slot` aufrufen:

```
<html>
    <body>
    <div metal:use-macro="context/time_template/macros/time">
      <div metal:fill-slot="msg">Die Zeit ist:</div>
      Falls es eine Message gibt, zeigt das Makro sie hier an.
    </div>
    </body>
</html>
```

Als Endergebnis sehen Sie den wie folgt gefüllten `time-slot`:

```
<html>
    <body>
    <div>
```

```
        <div>Die Zeit ist: </div>
        <div>2004/04/15 17:18:18.312 GMT-7</div>
    </div>
    </body>
</html>
```

Wie Plone Makros und Slots benutzt

Makros und Slots sind sich insofern ähnlich, als beide Inhalte aus einem anderen Template extrahieren und Inhalte einfügen, aber sie tun das auf unterschiedliche Weise. Der Unterschied besteht darin, wie sie benutzt werden. Makros sind Elemente eines Templates, die explizit aufgerufen werden, während Slots wie Löcher in einem Template sind, die von anderen Templates für Sie gefüllt werden sollen. Bei Plone selbst sind Portlets wie Kalender, Navigation etc. Makros, die explizit aufgerufen werden.

Wenn Sie sich im ZMI (Zope Management Interface) eine Datei anschauen, indem Sie hintereinander auf PORTAL_SKINS, PLONE_TEMPLATES und MAIN_TEMPLATE klicken, werden Sie sogar sehen, dass die gesamte Seite aus Makros und Slots besteht. In diesem Stadium ist sie vermutlich ein wenig verwirrend, aber wenn sie aufgerufen wird, läuft sie durch eine Reihe von Makros und trägt alles zusammen. Damit kann ein Benutzer ganz leicht alle Teile einer Plone-Site dadurch ändern, dass er das Makro überschreibt, was Sie im nächsten Kapitel sehen werden. Beispiel:

```
...
<div metal:use-macro="here/global_siteactions/macros/site_actions">
    Site-wide actions (Contact, Sitemap, Help, Style Switcher etc)
</div>

<div metal:use-macro="here/global_searchbox/macros/quick_search">
    The quicksearch box, normally placed at the top right
</div>
...
```

Beim weiteren Herunterscrollen in `main_template` werden Sie einige define-Slots sehen. Ich wiederhole kurz, wie eine Seite in Plone dargestellt wird. Wenn ein Objekt angezeigt wird, wird ein Template für die Ansicht dieses Inhalts angezeigt. Wenn Sie ein Bild anzeigen, wird das Template `image_view` angezeigt, und dieses Template bestimmt, wie das Bild angezeigt wird. Um diese Aufgabe zu erfüllen, füllt das Bild-Template den Slot `main`. Wenn Sie ins Template `image_view` schauen, sehen Sie darin den folgenden Code:

```
<div metal:fill-slot="main">
...
</div>
```

Wenn Sie zurück zum `main_template` springen, werden Sie sehen, dass es die Definition `define-slot` für den Slot `main` enthält:

```
<metal:bodytext metal:define-slot="main" tal:content="nothing">
    Page body text
</metal:bodytext>
```

Zu jedem Inhaltstyp gibt es ein eigenes Template, und jedes Template definiert für sich, wie es den Slot `main` benutzt. Das heißt, jeder Inhaltstyp erhält sein eigenes Look-and-feel aus dem Template. Nur ein Teil fehlt noch in dieser Gleichung. Als Sie `image_view` aufgerufen haben, wusste das Template irgendwie, dass es `main_template` verwenden sollte. Im Template `image_view` benutzen Sie das Makro aus `main_template`. Dieses wird mit folgendem HTML-Code definiert:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-US"
    lang="en-US"
    metal:use-macro="here/main_template/macros/master"
    i18n:domain="plone">
```

In diesem Fall wird in `main_template` der Slot `main` von dem als `main` definierten Slot in demjenigen Template gefüllt, das dargestellt wird. Folgendes ist die zeitliche Abfolge der Ereignisse bei der Darstellung eines Bildes:

1. Der Benutzer verlangt das Template `image_view` eines Bildes.
2. Das Makro `main_template` oben im Template `image_view` wird gefunden und aufgerufen.
3. Das `main_template` wird verarbeitet.
4. Das `define-slot="main"` wird gefunden, und das `fill-slot` wird in `image_view` gefunden.
5. Das `fill-slot` wird aufgerufen, und die Ergebnisse werden in `main_template` eingefügt.
6. Das `main_template` wird weiterverarbeitet.
7. Das Ergebnis wird zum Browser zurückgeschickt.

Das ermöglicht Plone eine hohe Flexibilität bei der Definition einer Seite. So definiert z.B. `main_template` kaum mehr als diesen einen Slot. Es gibt auch einen Slot zum Einfügen von CSS-Code (Cascading Style Sheets):

```
<metal:cssslot fill-slot="css_slot">
    <metal:cssslot define-slot="css_slot" />
</metal:cssslot>
```

Falls eine Ansicht einen speziellen CSS-Teil bräuchte, könnten Sie diesen Slot in der Ansicht definieren, und er würde bei der Darstellung gefüllt werden. Einige der Makros in `main_template` definieren darin auch Slots und füllen sie im

`main_template` wieder. Das heißt, wenn Sie wirklich wollten, könnten Sie auch diese Slots füllen. Allerdings ist das eine Technik für Fortgeschrittene, d.h., Sie sollten erst die Grundlagen beherrschen, bevor Sie sich auf diesen Weg begeben.

6.1.2 Einführung in die Internationalisierung

Die Macher von Plone sind kontinuierlich bestrebt, eine große Zahl qualitativ hochwertiger Übersetzungen anzubieten. Die Tatsache, dass Plone eine zugängliche Benutzerschnittstelle in über 30 Sprachen bietet, ist ein großes Verkaufsargument für Plone. Das heißt auch, dass I18N ein Schlüsselmerkmal von Templates ist. Das wird dadurch unterstützt, dass der I18N-Namespace ein Extra-Namespace wie TAL oder METAL ist, der spezielle Anweisungen enthält.

Dieser Abschnitt enthält, was Benutzer in Bezug auf Templates wissen müssen. In einem Template können Sie einen `i18n`-Tag zu einem Element hinzufügen, der die Übersetzung eines Attributs oder seines Inhalts erlaubt. Es sind sechs Anweisungen vorhanden: `attributes`, `data`, `domain`, `source`, `target` und `translate`. Das Grundmuster besteht darin, den zu übersetzenden Text einzuwickeln und die passenden `i18n`-Attribute hinzuzufügen. Wenn Sie z.B. Folgendes übersetzen möchten:

```
<i>Irgendein Text</i>
```

würde daraus Folgendes:

```
<i i18n:translate="some_text_label">Irgendein Text</i>
```

Jede Lokalisierung enthält eine Übersetzung von `Irgendein Text`, und das Translation-Werkzeug sucht nach einer Übersetzung für den Benutzer. Bei der Übersetzung muss jeder zu übersetzende String eine eindeutige Message-ID haben, die das zu übersetzende Element identifiziert. Ein String wie `Suchen` kann z.B. die Message-ID `search_widget_label` haben. Mit dieser Message-ID kann der String eindeutig identifiziert werden, und die Übersetzung kann wiederholt werden.

i18n:translate

Folgendes übersetzt den Inhalt eines Elements, wobei eine optionale Message-ID als Anweisung übergeben werden kann. Zum Beispiel erzeugt dieser Code eine Message-ID namens `title_string`:

```
<h1 i18n:translate="title_string">Dies ist ein Titel</h1>
```

Dieses Beispiel steht für einen statischen Text, der sich nicht verändert. In manchen Situationen könnte dieser Textteil dynamisch aus einer Datenbank oder einem Objekt kommen. Dadurch, dass die Anweisung `translate` leer gelassen wird, setzt sich die Message-ID aus dem Wert im Feld zusammen. Wenn im fol-

genden Beispiel der Titel vom Pfadausdruck `here/title` z.B. `Alice im Wunderland` wäre, würde dieser Titel an das Translation-Werkzeug übergeben. Wenn keine Übersetzung vorhanden ist, wird der Originalwert eingesetzt:

```
<h1
    tal:content="here/title"
    i18n:translate="">
      Dies ist ein Titel.
</h1>
```

Der `translation`-Befehl ist wahrscheinlich einer der häufigsten `i18n`-Tags, den Sie benutzen werden, und Sie werden ihn in allen Plone-Templates sehen. Damit können Sie nicht nur statische Teile Ihrer Site wie Feldnamen in Formularen, Hilfen und Beschreibungen übersetzen, sondern auch die dynamischen Teile Ihrer Site, die sich öfter verändern können, beispielsweise Titel.

i18n:domain

Hiermit wird die Domain für die Übersetzung gesetzt. Um Konflikte zu vermeiden, kann jede Site mehrere Domains oder Gruppen von Übersetzungen haben. Es kann z.B. eine Domain für Plone und eine für Ihre eigene Anwendung geben. Plone verwendet die Domain `plone`, was in Plone normalerweise die Standard-Domain ist:

```
<body i18n:domain="plone">
```

Diesen Tag sollten Sie nicht oft benutzen müssen. Wenn Sie jedoch eine eigene Anwendung schreiben, ist es für Sie vielleicht nützlich, eine Domain zu haben, die keine Konflikte mit anderen Domains hervorruft.

li8n:source

Hiermit wird die Ausgangssprache für den zu übersetzenden Text gesetzt. In Plone wird das nicht benutzt:

```
<p i18n:source="en" i18n:translate="">Irgendein Text</p>
```

i18n:name

Dies bietet eine Möglichkeit, Elemente in einem längeren Textblock beizubehalten, damit der Textblock umgeordnet werden kann. In vielen Sprachen sind nicht nur die Wörter andere, sondern auch ihre Reihenfolge variiert. Wenn Sie einen ganzen Absatz oder einen Satz übersetzen müssen, der kleinere Teile enthält, die nicht übersetzt werden sollen, können diese durchgereicht werden:

```
<p i18n:translate="book_message">
    Das
    <span
```

```
        tal:omit-tag=""
        tal:content="book/color"
        i18n:name="color">blaue</span>
    Buch
</p>
```

Das produziert den folgenden String:

```
Das {color} Buch
```

Wenn in der Zielsprache verlangt wird, dass diese Wörter in einer anderen Reihenfolge stehen, könnten sie verschoben werden, und der dynamische Inhalt würde weiterhin am richtigen Platz eingesetzt. Auf Französisch müsste das wie folgt übersetzt werden:

```
Le Livre {color}
```

i18n:target

Hiermit wird die Zielsprache für den zu übersetzenden Text gesetzt. In Plone wird das nicht benutzt.

i18n:attributes

Dadurch ist es möglich, Attribute in einem Element statt deren Inhalt zu übersetzen. Ein `image`-Tag hat z.B. das Attribut `alt`, das eine alternative Textrepräsentation des Bildes enthält:

```
<img
    href="/einBild.jpg"
    alt="Irgendein Text"
    i18n:attributes="alt alternate_image_label" />
```

Mehrere Attribute sollten durch ein Semikolon getrennt werden, genau wie bei `tal:attributes`.

i18n:data

Damit können andere Dinge als Strings übersetzt werden. Ein Beispiel dafür ist das Objekt `DateTime`. Eine `i18n:data`-Anweisung benötigt eine passende `i18n:translate`-Anweisung, damit eine gültige Message-ID verfügbar ist. Beispiel:

```
<span i18n:data="here/currentTime"
    i18n:translate="timefmt"
    i18n:name="time">2:32 pm</span>... Piep!
```

Übersetzungsdienst

Nachdem ich die Tags nun behandelt habe, werde ich jetzt den Mechanismus für die Durchführung der Übersetzung beschreiben. Plone enthält standardmäßig einen I18N-Mechanismus. Mit diesem können Sie die Benutzerschnittstelle so lokalisieren, dass Sie Meldungen, Reiter und Formulare übersetzen können. Der von Benutzern hinzugefügte Inhalt wird im Moment davon nicht erfasst. Wenn Sie ein Dokument in englischer Sprache hinzufügen und die Seite anzeigen, die es auf Französisch abruft, erhalten Sie das englische Dokument mit französischem Text drumherum (siehe Abbildung 6.1).

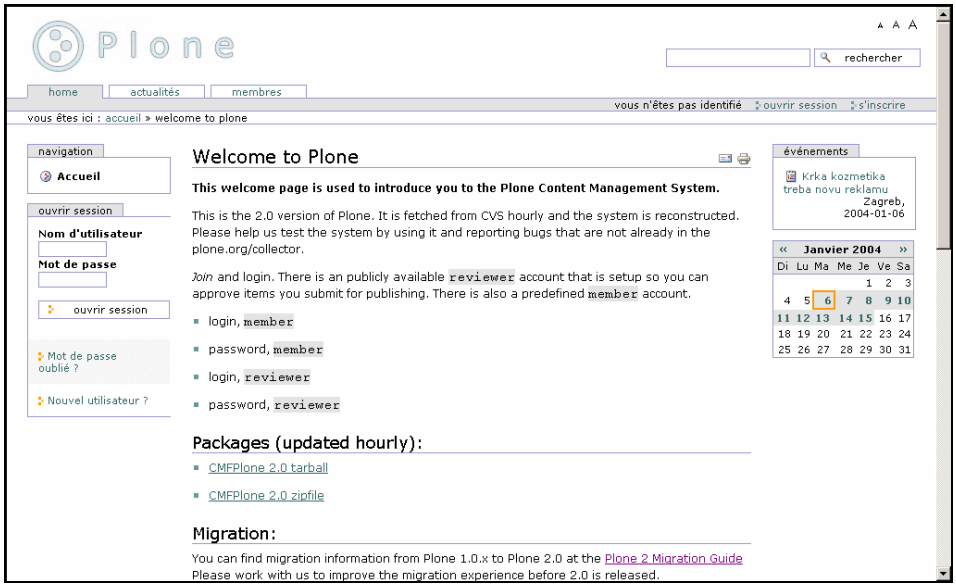


Abbildung 6.1: Plone.org auf Französisch

Plone liest die HTTP-Header, die ein Browser an den Client sendet, um eine Sprache abzufragen. Wenn in Ihrem Browser Englisch eingestellt ist, werden Sie nicht viel sehen.

Tun Sie Folgendes, um die Spracheinstellungen im Internet Explorer zu ändern:

1. Gehen Sie zu WERKZEUGE – INTERNET-OPTIONEN.
2. Klicken Sie auf den SPRACHE-Button unten im Dialogfeld.
3. Klicken Sie auf HINZUFÜGEN, um eine neue Sprache hinzuzufügen, und wählen Sie eine Sprache, für die Plone eine Übersetzung hat, z.B. Französisch, und klicken Sie auf HINZUFÜGEN.
4. Stellen Sie dann sicher, dass diese Sprache oben ist, indem Sie den NACH OBEN-Button verwenden.

5. Klicken Sie auf OK und noch einmal auf OK.

Wenn Sie damit fertig sind, wählen Sie Ihre bevorzugte Plone-Site, und besuchen Sie diese mit Ihrem Browser.

Die Übersetzungen für Plone werden mit einem Werkzeug namens Placeless Translation Service (PTS) vorgenommen. Das PTS-Werkzeug finden Sie im Zope-Control Panel. Unten auf der Seite sehen Sie eine Option für PLACELESS TRANSLATION SERVICE. Klicken Sie darauf, und es werden alle vorhandenen Übersetzungen geöffnet. Diese Übersetzungen werden aus dem Dateisystem gelesen. Klicken Sie auf eine Übersetzung, um Angaben über die Sprache wie den Übersetzer, die Codierung und den Pfad zur Datei zu sehen. All diese Dateien werden im `i18n`-Unterverzeichnis vom `CMFPlone`-Verzeichnis gespeichert.

Übersetzungen werden jeweils mit zwei Dateien mit den Endungen `.po` und `.mo` vorgenommen. So enthält z.B. `plone-de.po` die deutschen Übersetzungen (`de` ist der Code für Deutsch). Die `.mo`-Datei ist die »kompilierte« Version der `.po`-Datei und wird aus Performance-Gründen von Plone benutzt. In die `.mo`-Datei müssen Sie nie hineinschauen, und Sie können sie einfach ignorieren. Die `.po`-Datei ist jene, die Sie bearbeiten können, um eine Übersetzung zu ändern. Wenn Sie diese Datei in einem Texteditor öffnen, sehen Sie eine Reihe von Zeilen, die mit `msgid` oder `msgstr` anfangen. Über dem `msgid` befindet sich der Code, wo der `i18n`-Befehl vorkommt, d.h., Sie sehen, welchen Teil einer Seite Sie übersetzen. Beispiel:

```
#: from plone_forms/content_status_history.pt
#.   <input attributes="tabindex tabindex/next;" value="Apply"
class="context" name="workflow_action_submit" type="submit" />
#.
#: from plone_forms/personalize_form.pt
#.   <input attributes="tabindex tabindex/next;" tabindex=""
value="Apply" class="context" type="submit" />
#.
msgid "Apply"
msgstr "Anwenden"
```

In den zwei Teilen des vorigen Page Templates wird das Wort *Apply* für deutsche Benutzer in das Wort *Anwenden* übersetzt. Was übersetzt wird, bestimmen die `i18n`-Tags, die in die Page Templates eingefügt wurden, wie Sie zuvor schon gesehen haben. Wenn Sie diese Übersetzung ändern möchten oder wenn Sie eine eigene Variante davon hinzufügen möchten, müssen Sie lediglich die `.po`-Datei ändern. Falls kein `msgstr` gefunden wird, wird die englische Standardübersetzung gefunden. Nach einer solchen Änderung müssen Sie Plone neu starten. Dabei wird Plone diese Datei in die Version `.mo` neu kompilieren und dann Ihre aktualisierte Übersetzung verwenden.

Was Plone angeht, so wird per Standardeinstellung immer die englische Übersetzungsdatei verwendet, falls keine Sprache angegeben oder keine Übersetzung verfügbar ist. Tatsächlich ist die Datei `plone-en.po` leer, d.h., es ist keine solche Übersetzung verfügbar. Deswegen greift Plone zum letzten Mittel, nimmt keine Übersetzung vor, und zeigt den Text im Page Template an. Der Text in allen Page Templates liegt in englischer Sprache vor, da die meisten Entwickler englisch sprechen. Zusammengefasst heißt das, dass es keine englische Übersetzung gibt.

Deswegen können Sie eine neue Übersetzung dadurch machen, dass Sie die Datei `plone.pot` in eine neue Datei namens `plone-xx.po` kopieren. Dabei sollte der Wert von `xx` dem Landescode Ihrer Übersetzung entsprechen. Eine Liste von Sprachcodes finden Sie unter *http://www.unicode.org/onlinedat/languages.html*. Sobald Sie mit der Übersetzung angefangen haben, setzen Sie die obigen Werte, inklusive Sprachcode, und übersetzen drauflos. Wenn Sie eine neue Sprachdatei vollendet haben, wird das Plone-I18N-Team diese glücklich entgegennehmen und Ihnen bei der Fertigstellung helfen. Die Mailing-Liste des Plone-Teams finden Sie unter *http://sourceforge.net/mailarchive/forum.php?forum_id=11647*.

Den von Benutzern hinzugefügten Inhalt zu übersetzen ist eine wesentlich komplexere Angelegenheit, bei der die Tücke im Detail liegt. Bislang gab es hier den Ansatz mit `I18NLayer`, der aber aufgrund von Seiteneffekten nicht einfach anzuwenden war. Inzwischen wurde von PloneSolutions das Nachfolgeprodukt *LinguaPlone* entwickelt, das wesentlich weniger Seiteneffekte hat und dem Ersteller von Inhalten wesentlich weniger Kenntnis der Materie abverlangt. Als dieser Abschnitt geschrieben wurde, war LinguaPlone in der Version 0.7.2 auf der Webseite von PloneSoutions (*http://www.PloneSolution.com*) verfügbar. Beim Einrichten von LinguaPlone muss man allerdings noch einige manuelle Schritte vornehmen und sicherstellen, dass Archetypes 1.3 und ATContentTypes installiert sind. Erst bei der Plone-Version 2.1 wird LinguaPlone standardmäßig mit ausgeliefert und installiert werden.

6.1.3 Beispiel: Mehr Benutzerinformationen anzeigen

In Kapitel 5 haben Sie einfache TAL-Befehle benutzt, um detailliertere Angaben zu einem Benutzer anzuzeigen. Das dortige Template hat einige Nachteile. Einer davon ist, dass es nur jeweils einen Benutzer anzeigt. Wie Sie gesehen haben, können Sie mit einem einfachen `tal:repeat` Inhalt wiederholen, aber jetzt werden Sie ein Makro verwenden, um diese Seite modularer zu machen.

Sie werden das Page Template `user_info` so ändern, dass es alle Site-Mitglieder auflistet. Anstatt nach einem in der Anfrage übergebenen Benutzernamen zu suchen, werden Sie dabei die Funktion `listMembers` benutzen, die eine Liste aller Mitglieder der Site zurückgibt:

```
<div metal:fill-slot="main">
  <tal:block
   tal:define="
   getPortrait nocall: here/portal_membership/getPersonalPortrait;
   getFolder nocall: here/portal_membership/getHomeFolder
   ">
   <table>
     <tr tal:repeat="userObj here/portal_membership/listMembers">
         <metal:block
          metal:use-macro="here/user_section/macros/userSection" />
     </tr>
   </table>
 </tal:block>
</div>
```

Sie werden bemerken, dass nun der Code für `user_info` wesentlich kürzer ist. Das von `listMembers` zurückgegebene Mitglied wird an `tal:repeat` weitergegeben. Für jedes Mitglied wird es eine Tabellenzeile geben und dann ein Makro, um dem Benutzer Informationen anzuzeigen. In dieser Tabellenzeile enthält die lokal definierte Variable `userObj` nun die Benutzerangaben. Natürlich müssen Sie nun ein Makro namens `userSection` in einem Page Template erstellen, d.h., Sie erstellen ein Page Template namens `user_section`, wie es vom Makro referenziert wird. Dieses Template enthält den gesamten Code zwischen den `row`-Tags der Tabelle. Auch hierfür finden Sie das vollständige Listing für dieses Page Template in Anhang B:

```
<div metal:define-macro="userSection"
     tal:define="userName userObj/getUserName">
     ...
```

Die einzige wirkliche Änderung besteht darin, dass `use-macro` im Haupt-Template entfernt werden und ein neues Makro definiert werden muss, damit dieses Makro definiert werden kann. Weil der Benutzername nicht mehr explizit übergeben wird, müssen Sie ihn mit der Methode `getUserName` aus dem Benutzerobjekt holen. Um die Ergebnisseite zu testen, gehen Sie zu *http://ihresite/user_info*, wo Sie eine Liste der Benutzer sehen sollten.

Nun ist die Seite benutzerfreundlich, da sie mehrere Benutzer auf einmal anzeigt. Der Code ist modularer, da er die Benutzerangaben in einem separaten Makro darstellt, das unabhängig vom Rest verändert werden kann. Die Seite ist noch nicht perfekt, wird aber in späteren Kapiteln noch verbessert.

6.1.4 Beispiel: Ein neues Portlet mit Google Ads erstellen

In Kapitel 4 haben Sie gesehen, wie Sie die Portlets in einer Plone-Site einfach bearbeiten können. Ihr eigenes Portlet zu erstellen ist nicht viel schwerer. Um Ihren eigenen Slot zu schreiben, müssen Sie ein neues Page Template mit einem Makro darin erstellen. Dann wird ein TALES-Ausdruck, der auf das Makro zeigt, zur Liste der Portlets hinzugefügt, wodurch das Portlet auf der Seite angezeigt wird.

Das Grund-Template für ein Portlet sieht wie folgt aus:

```
<div metal:define-macro="portlet">
    <div class="portlet">
      <!-- Geben Sie hier Code ein -->
    </div>
</div>
```

Alles, was Sie tun müssen, ist, den passenden Code ins Portlet einzufügen. Google hat 2003 ein textbasiertes Anzeigensystem erstellt, das Text auf Ihrer Site unterbringt. Die Anzeigen basieren auf Googles Annahmen darüber, worum es bei Ihrer Site geht, und diese Annahmen basieren auf den Suchergebnissen zu Ihrer Site. Das Google-System ist unter *http://www.google.com/adsense* verfügbar. Um Anzeigen anzuzeigen (und Geld dafür zu bekommen), müssen Sie sich bei Google registrieren. Auf der Google-Website werden Sie gebeten, einige Farben und Stile auszuwählen. Da Sie das in einen Slot einsetzen werden, empfehle ich die Größe »skyscraper«, d.h. hoch und dünn. Machen Sie eine Kopie von dem JavaScript, das die Site produziert.

Als Nächstes müssen Sie ein Portlet erstellen:

1. Erstellen Sie im Ordner `portal_skins/custom` ein Page Template namens `googleAds`.
2. Nehmen Sie den vorigen Template-Grundcode, und ändern Sie den Portlet-Namen auf `googleBox`.
3. Setzen Sie den Code von Google ein, wobei Sie den Abschnitt `<!-- Geben Sie hier Code ein -->` ersetzen.

Das Endergebnis sollte Listing 6.1 ähneln. Ihre Version wird allerdings einen gültigen Wert für `google_ad_client` enthalten statt `yourUniqueValue`. Dieser Wert sagt Google, welche Site diese Anzeige angefordert hat und wer dafür bezahlt werden soll. Merkwürdigerweise wird Google, wenn Sie keinen gültigen Wert dafür haben, die Anzeigen trotzdem anzeigen, ohne Sie dafür zu bezahlen!

Listing 6.1: Anzeigen von Google anzeigen

```
<div metal:define-macro="portlet">
    <div class="portlet">
<script type="text/javascript"><!--
google_ad_client = "yourUniqueValue";
google_ad_width = 120;
google_ad_height = 600;
google_ad_format = "120x600_as";
//--></script>
<script type="text/javascript"
  src="http://pagead2.googlesyndication.com/pagead/show_ads.js">
</script>

   </div>
</div>
```

Um das auf Ihrer Site einzufügen, fügen Sie folgendes Portlet zu Ihrer Portlet-Liste hinzu, wie es in Kapitel 4 beschrieben wurde:

```
here/googleAds/macros/portlet
```

6.2 Plone mit Python scripten

In Plone existieren mindestens vier verschiedene Ebenen, auf denen eine Programmlogik erstellt werden kann. Die einfachste Ebene, auf der Python in Plone verwendet werden kann, ist der Python-TALES-Ausdruck, den ich im vorigen Kapitel besprochen habe. Ein Python-Ausdruck darf allerdings nur eine Zeile Code enthalten, und oftmals werden Sie etwas Komplexeres machen wollen.

Noch häufiger kommt es vor, dass Sie nur ungern die ganze Logik in das Template hineinzwängen wollen. Ganz allgemein ist es keine gute Idee, Anwendungslogik in Ihr Template zu setzen. Immer, wenn Sie etwas, das nicht explizit Präsentationslogik ist, aus Ihrem Template hinaus verlegen können, haben Sie eine Menge Kopfschmerzen gespart. Die Trennung von Anwendungslogik und Präsentation ermöglicht es, dass verschiedene Leute an unterschiedlichen Projektteilen arbeiten, und sie verbessert die Wiederverwendung von Code. Die anderen Ebenen, auf denen Scripten zu Plone hinzugefügt werden können, kommen ungefähr in folgender Reihenfolge vor:

- **Template-Attributausdrücke**: Diese bieten Ausdrücke und eine Möglichkeit, kleine Schnipsel mit Logik oder einfachen Pfaden an vielen Orten einzufügen.
- **Script (Python)-Objekte**: Dies sind einfache Scripten, die in einer eingeschränkten Umgebung in Plone ausgeführt werden.

- **Externe Methodenobjekte**: Dies sind kompliziertere Module, die nicht in eingeschränkten Umgebungen ausgeführt werden.
- **Python-Produkte**: Daraus bestehen die wichtigsten Quellen, in denen das CMF und Plone geschrieben sind. Damit hat man Zugriff auf alles in Plone. Python-Produkte sind ein Thema für Fortgeschrittene und werden in Kapitel 12 behandelt.

Nach einem Ausdruck ist ein *Script (Python)-Objekt* die nächstkompliziertere Stufe. Dieses Objekt ermöglicht mehrere Zeilen Python-Code, und Sie können es aus einem Ausdruck aufrufen. Wenn Sie ein Script (Python)-Objekt aufrufen, fällt ein kleiner zusätzlicher Aufwand an, weil Plone zu diesem Objekt umschaltet. Der Aufwand ist jedoch minimal, da es einen Kompromiss zwischen Klarheit, Trennung und Performance gibt. Mein Rat ist der, so viel Logik wie möglich nach Python zu verlagern und Page Templates so einfach und sauber wie möglich zu halten. Wenn es ein Performance-Problem gibt, kann man sie später leicht wieder zurückverlegen, aber zumindest werden Sie später verstehen, was passiert.

6.2.1 Script(Python)-Objekte verwenden

Ein Script (Python)-Objekt ist das, woran Sie in Plone normalerweise bei einem Script denken. Es ist ein Python-Schnipsel, den Sie schreiben und dann aus anderen Templates oder direkt über das Web aufrufen können. Tatsächlich verfügt Plone über eine große Anzahl dieser Scripten, um verschiedene wichtige Funktionen auszuführen. Was seine Mächtigkeit angeht, liegt ein Python-Script etwa auf halbem Weg zwischen einem Ausdruck und einer externen Methode.

Um ein Script (Python)-Objekt hinzuzufügen, gehen Sie ins ZMI, wählen SCRIPT (PYTHON) im Dropdown-Menü und klicken auf ADD, wie in Abbildung 6.2 zu sehen ist.

Geben Sie dem Script z.B. die ID **test_py**, und klicken Sie dann auf ADD AND EDIT. Danach wird die Bearbeiten-Seite für das Script (Python)-Objekt geöffnet, die aussieht wie in Abbildung 6.3.

Sie können das Script direkt über das Web bearbeiten. Falls Sie einen Syntaxfehler machen, erfahren Sie das, gleich nachdem Sie auf SAVE CHANGES geklickt haben (siehe Abbildung 6.4).

Falls es keinen Fehler in Ihrem Python-Script gibt, können Sie den TEST-Reiter anklicken, um zu sehen, wie die Ausgabe aussieht. In diesem Fall ist das Beispiel recht langweilig und gibt folgenden Text aus:

```
This is the Script (Python) "test_py" in
http://gloin:8080/Plone/portal_skins/custom
```

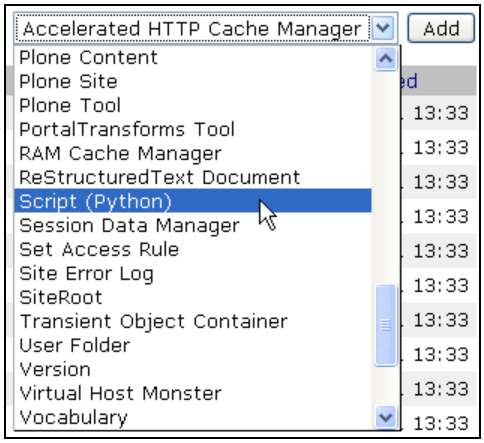


Abbildung 6.2: Ein Script (Python)-Objekt hinzufügen

Abbildung 6.3: Ein Script (Python)-Objekt bearbeiten

Ein Script verfügt auch über die folgenden Optionen:

- **Title**: Das Bearbeiten-Formular hat eine TITLE-Option, mit der Sie dem Script einen Titel geben können. Dieser wird im ZMI angezeigt, d.h., man erinnert sich dann leichter daran, was es macht.
- **Parameter List**: Dies ist eine Liste von Parametern, die das Script erwartet, z.B. `variableA` oder `variableB=None`. Dies ist sogar die Standardparameterliste, die Sie in einer normalen Python-Funktion erwarten würden. Manche Parameter sind in diesem Objekt allerdings schon für Sie definiert. Sie können Sie sehen, wenn Sie auf den Reiter BINDINGS klicken. Dort sehen Sie eine Liste der Variablen, die bereits an das Objeckt gebunden sind. Mit ihren Namen sollten Sie schon vertraut sein.

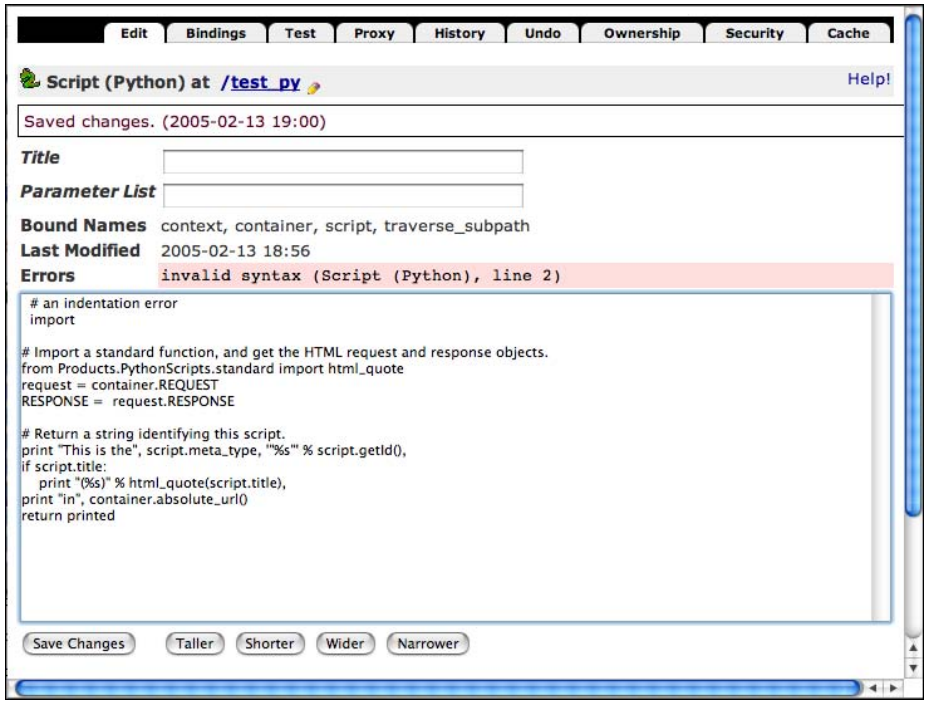


Abbildung 6.4: Ein absichtlicher Einrückungsfehler im Script (Python)-Objekt

Es gibt folgende an das Script gebundene Variablen, auf die von einem Script (Python)-Objekt zugegriffen werden kann:

- **context**: Dies ist das Objekt, auf dem das Script aufgerufen wird.
- **container**: Dies ist das Objekt, das dieses Script enthält.

- **script**: Dies ist das Script (Python)-Objekt selbst. Das Äquivalent in Zope Page Templates ist `template`.
- **namespace**: Dies ist für den Fall da, dass das Script aus DTML (Document Template Markup Language) heraus aufgerufen wird – etwas, was in Plone nicht passiert.
- **traverse_subpath**: Dies ist der URL-Pfad nach dem Scriptnamen, was eine Eigenschaft für Fortgeschrittene ist.

Ich werde nun ein einfaches Beispiel zeigen, das diese Themen mit dem Zope Page Templates-System verbindet, wobei der Python-Ausdruck zur Addition zweier Zahlen verwendet wird, den ich im vorigen Kapitel angegeben habe. Wie Sie gesehen haben, konnten Sie ein Page Template dafür anlegen, das wie folgt aussieht:

```
<p>1 + 2 = <em tal:content="python: 1 + 2" /></p>
```

Das Äquivalent mit einem Script (Python)-Objekt sieht wie folgt aus. Ändern Sie das Script `test_py` auf die folgende Zeile:

```
return 1+2
```

Wie Sie am Anfang des vorigen Kapitels gesehen haben, rufen Sie ein Objekt auf, indem Sie seinen Pfad als Ausdruck angeben. In einem Page Template können Sie also Folgendes machen:

```
<p>1 + 2 = <em tal:content="here/test_py" /></p>
```

Durch den Pfadausdruck erhalten Sie das Objekt `test_py` und rufen es auf. Dann gibt es den Python-Code ans Template zurück und gibt es aus. Soeben haben Sie ein Script aus einem Template aufgerufen! Dies ist ganz offensichtlich ein einfaches Beispiel, aber der Punkt ist der, dass Sie in einem Script (Python)-Objekt eine ganze Menge dessen tun können, was Sie in einem Page Template nicht können.

In einem Script (Python)-Objekt können Sie den Titel, Parameter und Bindungseinstellungen angeben, indem Sie die *##*-Notation am Anfang eines Scripts verwenden. Wenn Sie ein Script mit diesem Stück Text oben speichern, wird Plone diese Zeile entfernen und stattdessen den entsprechenden Wert im Objekt ändern. Diese Syntax wird sehr oft im Script (Python)-Objekt in diesem Buch verwendet, um sicherzugehen, dass Sie über den richtigen Titel und die richtigen Parameter verfügen. Das heißt, Sie könnten das vorige Script wie folgt umschreiben:

```
##title=Ergibt 1+2
##parameters=
return 1+2
```

Plone scripten

Plone mit Scripten zu steuern ist ein recht kompliziertes Unterfangen, weil Sie, sobald Sie Plone scripten können, die API (Application Programming Interface) aller Objekte und Werkzeuge beachten müssen, die Sie verwenden möchten. Eine Erklärung der APIs würde den Rahmen des Buches sprengen, daher werde ich demonstrieren, wie man einige einfache Aufgaben mit Script(Python)-Objekten erledigt. Wenn Sie einmal etwas vertrauter damit sind, werde ich weitere API-spezifische Funktionen beschreiben.

Page Templates können recht schön über Python-Dictionaries und -Listen iterieren. Aber oftmals liegen Ihre Daten nicht in einem dieser bequemen Formate vor, d.h., Sie müssen zu einem Script (Python)-Objekt greifen, um die Daten hübsch zu formatieren und sie an das Page Template zurück zu übergeben.

Am bequemsten ist das Datenformat einer Liste von Dictionaries, bei der Sie die Mächtigkeit eines `tal:repeat` mit einem Pfadausdruck in einer Funktion kombinieren können. Als Beispiel werden Sie eine Funktion sehen, die eine Liste von Objekten erwartet. Jedes dieser Objekte ist ein Objekt in einem Ordner. Und jedes Objekt wird dann vorhanden sein, wenn es in den letzten fünf Tagen aktualisiert worden ist. Listing 6.2 zeigt ein nützliches kleines Portlet, das ich für eine Site erstellt habe, die genau diese Art von Informationen suchen und die entsprechenden Einträge hervorheben sollte.

Listing 6.2: Bis zu fünf Tage alte Objekte zurückgeben

```
##title=recentlyChanged
##parameters=objects
from DateTime import DateTime

now = DateTime()
difference = 5 # in Tagen
result = []

for object in objects:
  diff = now - object.bobobase_modification_time()
  if diff < difference:
    dct = {"object":object,"diff":int(diff)}
    result.append(dct)

return result
```

In diesem Script (Python)-Objekt habe ich ein paar neue Konzepte benutzt. Zuerst einmal importieren Sie Zopes `DateTime`-Modul mit der Anweisung `import`. Das Modul `DateTime`, das in Anhang C behandelt wird, bietet Zugriff auf Datumsangaben. Es ist ziemlich einfach, aber wenn Sie ein neues `DateTime`-Objekt ohne

Parameter anlegen, erhalten Sie das aktuelle Datum und die aktuelle Zeit, und zwar in der Variablen `now`. Wenn Sie zwei `DateTime`-Objekte subtrahieren, erhalten Sie die Anzahl der Tage dazwischen. Diese können Sie mit der Anzahl vergleichen, die ein Benutzer überwachen möchte, und wenn sie für ein Objekt größer ist, können Sie dieses zur Ergebnisliste hinzufügen. Das Ergebnis ist eine Liste von Dictionary-Objekten, die wie in Listing 6.3 aussieht.

Listing 6.3: Das Ergebnis von Listing 6.2

```
[
  {
      'diff': 1,
      'object': <PloneFolder instance at 02C0C110>
  },
  {
]      'diff': 4,
      'object': <PloneFolder instance at 02FE3321>
  },
  ...
```

Nun, da Sie die Ergebnisse in der richtigen Reihenfolge haben, benötigen Sie ein Page Template, das die Liste der Objekte übergibt und die Ergebnisse verarbeitet. Hier ist ein Beispiel dafür:

```
<ul>
  <li tal:repeat="updated python:
context.updateScript(context.contentValues())">
```

Dieses Template enthält zu Beginn einen Aufruf `tal:repeat`, der das Script aufruft (in diesem Fall wird `updateScript` aufgerufen). An diese Funktion übergibt es einen Wert, und zwar eine Liste von `contentValues` aus dem aktuellen Kontext. Vorher haben Sie das Script (Python)-Objekt mit einem Pfadausdruck aufgerufen, was Sie hier mit `context/updateScript` auch tun könnten. Allerdings können Sie mit dieser Syntax keine Parameter an das aufgerufene Script übergeben, also nehmen Sie stattdessen einen Python-Ausdruck, `python: context.updateScript()`. Die Funktion `contentValues` gibt eine Liste aller Inhaltsobjekte in einem Ordner zurück. Betrachten Sie nun den Code für jede Iteration:

```
<a href="#"
     tal:attributes="href updated/object/absolute_url"
     tal:content="updated/object/title_or_id">
     Der Titel des Artikels</a>
  Vor <em tal:content="updated/diff" /> Tagen
  </li>
</ul>
```

Wie Sie sehen, können Sie über diese Liste von Werten iterieren, und Sie können dann Pfadausdrücke benutzen, um zuerst auf den wiederholten Wert (`updated`), dann auf das Objekt (`object`) und dann auf eine Methode dieses Objekts (`title_or_id`) zuzugreifen. Dies ist ein Beispiel für die Auslagerung einer komplizierten Logik in ein Script (Python)-Objekt.

Eingeschränktes Python

Ich habe mehrfach erwähnt, dass Script (Python)-Objekte und Python-TAL-Ausdrücke alle in einem *eingeschränkten* Python-Modus ausgeführt werden. Eingeschränktes (Restricted) Python ist eine Umgebung, aus der einige Funktionen entfernt worden sind. Diese Funktionen sind in einer Web-Umgebung wie Plone potenziell gefährlich. Der ursprüngliche Grund ist der, dass Sie es zwar mit authentifizierten Benutzern zu tun haben, denen Sie aber nicht trauen können und die Python-Code auf Ihrer Site ausführen. Wenn Sie bei einem der vielen Zope-Webhosts einen Account eröffnen, werden Sie feststellen, dass Sie das können. Wenn Sie den Leuten jedoch das Recht geben, das zu tun, möchten Sie nicht, dass sie Zugriff auf bestimmte Dinge wie das Dateisystem haben.

In eingeschränktem Python wurden aus Sicherheitsgründen einige bekannte Python-Funktionen entfernt, insbesondere sind `dir` und `open` nicht verfügbar. Das heißt, dass wie bei Script (Python)-Objekten keine Introspektion darauf möglich ist und der Zugriff auf das Dateisystem beschränkt ist. Einige Python-Module stehen dem Benutzer aber zur Verfügung. Die meisten davon sind für erfahrene Entwickler gedacht. Weitere Informationen finden Sie in der Dokumentation oder im entsprechenden Modul-Code:

- **string**: Dies ist das Python-Modul string (*http://python.org/doc/current/lib/module-string.html*).
- **random**: Dies ist das Python-Modul random (*http://python.org/doc/current/lib/module-random.html*).
- **whrandom**: Dies ist das Python-Modul whrandom. Heute benutzen Sie meist besser das Modul random (*http://python.org/doc/current/lib/module-whrandom.html*).
- **math**: Dies ist das Python-Modul math (*http://python.org/doc/current/lib/module-math.html*).
- **DateTime**: Dies ist Zopes eigenes `DateTime`-Modul.
- **sequence**: Dies ist ein Zope-Modul zum einfachen Sortieren von Sequenzen.
- **ZTUtils**: Dies ist ein Zope-Modul mit verschiedenen Hilfsfunktionen.
- **AccessControl**: Hiermit haben Sie Zugriff auf Zopes `Access`-Modul.

- **Products.PythonScripts.standard**: Das ermöglicht den Zugriff auf die normalen String-verarbeitenden Funktionen von DTML, z.B. `html_quote`, `thousands_commas` usw.

Wenn Sie ein Modul importieren möchten, das in der obigen Liste nicht auftaucht, können Sie im Modul `PythonScript` eine sehr gute Anleitung dafür finden. Diese finden Sie unter `Zope/lib/python/Products/PythonScripts/module_access_examples.py`. Es steht Ihnen allerdings auch eine einfachere Methode zur Verfügung, nämlich die Verwendung einer externen Methode.

6.2.2 Externe Methodenobjekte verwenden

Eine *externe Methode* ist ein Python-Modul im Dateisystem, das von Plone aufgerufen wird. Weil sie im Dateisystem gespeichert ist, wird sie nicht im eingeschränkten Python-Modus ausgeführt und gehorcht daher den normalen Plone-Sicherheitseinstellungen.

Sie können also ein Script schreiben, das tut, was immer Sie wollen, und es dann aus einem Page Template heraus aufrufen. Häufige Aufgaben hierfür sind das Öffnen und Schließen von Dateien, das Zugreifen auf andere Prozesse oder ausführbare Programme und das Ausführen von Aufgaben in Plone oder Zope, die Sie mit anderen Mitteln einfach nicht erledigen können. Aus offensichtlichen Gründen müssen Sie, wenn Sie ein solches Script schreiben, sicher sein, dass Sie nichts Gefährliches tun, z.B. die Passwortdatei auf Ihrem Server lesen oder ungewollt eine Datei löschen.

Um eine externe Methode hinzuzufügen, gehen Sie im Dateisystem zur Wurzel Ihrer Plone-Instanz und finden darin das Verzeichnis `Extensions`. Darin fügen Sie eine neue Python-Scriptdatei hinzu. Abbildung 6.5 zeigt, dass ich z.B. `test.py` in das Verzeichnis auf meinem Windows-Rechner eingefügt habe.

Nun können Sie `test.py` öffnen, nach Belieben bearbeiten und beliebigen Python-Code darin schreiben. Der einzige Haken ist der, dass Sie eine Eingangsfunktion haben müssen, die mindestens ein Argument erhält, nämlich `self`. Dieses Argument ist das externe Methodenobjekt in Plone, das Sie gleich hinzufügen werden. Folgendes ist ein Beispiel für eine Eingangsfunktion, die die Datei `README.txt` aus dem gleichen Verzeichnis `Extensions` liest und an den Benutzer zurückgibt (Sie müssen den Pfad so ändern, dass er auf Ihre Datei zeigt):

```
def readFile(self):
   fh = open(r'c:\Program Files\Plone\Data\Extensions\README.txt', 'rb')
   data = fh.read()
   return data
```

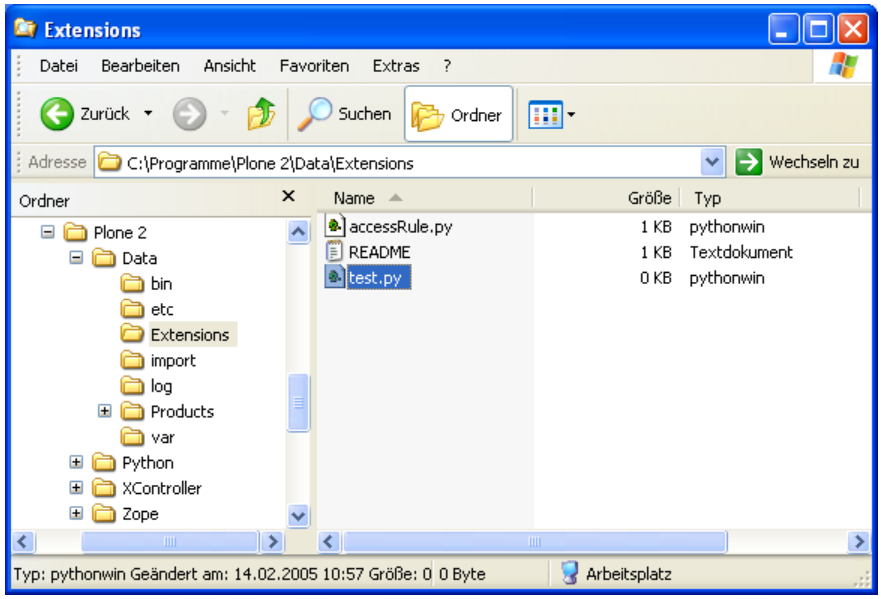


Abbildung 6.5: Eine neue externe Methode, test.py

Nachdem Sie das getan haben, müssen Sie eine externe Methode auf dieses Script abbilden. Da das ein Zope-Objekt ist, gehen Sie wieder zum ZMI, klicken auf PORTAL_SKINS und dann auf CUSTOM. Wählen Sie schließlich in der Dropdown-Liste ADD NEW ITEMS den Eintrag EXTERNAL METHOD. Wenn Sie eine externe Methode hinzufügen, müssen Sie den Namen des Moduls (ohne `.py`) und die Eingangsfunktion angeben. In diesem Fall sieht das ADD-Formular wie in Abbildung 6.6 aus.

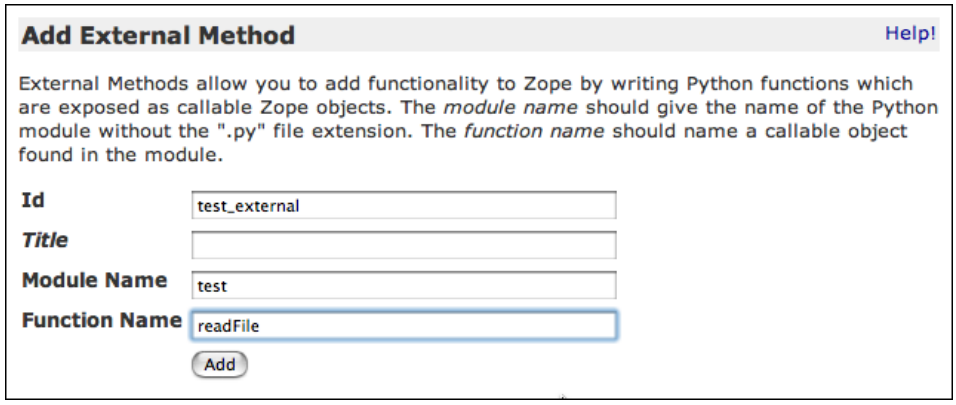


Abbildung 6.6: Die neu hinzugefügte externe Methode

Nach einem Klick auf SAVE CHANGES können Sie den TEST-Reiter anklicken, um zu sehen, was passiert, wenn die Methode ausgeführt wird. In diesem Fall sollten Sie ein oder zwei Zeilen Text erhalten. Da Ihr externes Methodenmodul in Plone ist, können Sie von einem Page Template genauso darauf zugreifen wie auf jedes andere Objekt. Ein Pfadausdruck auf `here/test_external` ist in dem Fall ausreichend. Beispiel:

```
<h1>README.txt</h1>
<p tal:content="here/test_external" />
```

Die wirkliche Mächtigkeit besteht darin, dass Sie Code in den uneingeschränkten Python-Modus und von dort an eine beliebige Funktion auslagern können, ohne sich um die Sicherheit sorgen zu müssen. Auch wenn das nach einer coolen Funktion aussieht, werden externe Methoden in Plone nicht übermäßig benutzt, weil komplexe Logik normalerweise in ein Produkt verlagert wird, während einfache Logik in einem Script(Python)-Objekt belassen wird. Sollten Sie feststellen, dass Sie sehr ausgiebigen Gebrauch von externen Methodenobjekten machen, möchten Sie vielleicht einmal einen Blick auf die in Kapitel 12 beschriebenen Werkzeuge werfen.

6.3 Nützliche Hinweise

Da Page Templates gültiger XML-Code sind und unabhängig von Zope oder Plone benutzt werden können, gibt es mehrere nützliche Scripten zum Aufräumen von Template-Code und zur Durchführung von Syntax-Prüfungen. Dies sind zusätzliche Werkzeuge und Prüfungen. Wenn Sie ein Page Template hochladen, führt Zope tatsächlich alle notwendigen Prüfungen durch. Bei einem Projekt wie Plone kann es hilfreich sein, automatische Prüfungen über Ihren Code laufen zu lassen oder ihn lokal zu prüfen, bevor Änderungen vorgenommen werden.

Um diese Prüfungen durchzuführen müssen Sie in der Lage sein, diese Werkzeuge lokal zu bearbeiten, und Sie müssen Python auf Ihrem Rechner installiert haben. Weitere Informationen zum External Editor, einer Methode für die lokale Bearbeitung von entferntem Code, finden Sie in Kapitel 10.

6.3.1 Einführung in XML-Namespaces

Page Templates verwenden XML-Namespaces, um Code zu generieren. Programmierer können sich mit den Regeln von XML-Namespaces das Leben leichter machen. Am Anfang eines Page Templates sehen Sie im ersten Tag eine Deklaration des Namespaces:

```
<html xmlns="http://www.w3.org/1999/xhtml"...
```

Damit wird der Standard-Namespace auf XHTML (Extensible HTML) gesetzt. Für alle darin enthaltenen Elemente gilt, dass sie diesen Namespace verwenden, wenn kein anderer angegeben ist. Somit wissen Sie z.B., dass das nächste Element XHTML ist, weil es kein Namespace-Präfix hat:

```
<body>
```

Bei TAL- und METAL-Elementen und -Attributen haben Sie normalerweise das Präfix `tal:` und `metal:` hinzugefügt, um den Namespace anzugeben. Der folgende Code sollte Ihnen nun vertraut vorkommen:

```
<span tal:omit-tag="" tal:content="python: 1+2" />
```

Das gibt 3 aus. Folgendes ist jedoch eine Alternative dazu:

```
<tal:number content="python: 1+2" />
```

Dadurch, dass Sie das Präfix `tal:` im Element verwenden, haben Sie den Standard-Namespace für dieses ganze Element als `tal` angegeben. Ohne weiteres Präfix wird der Namespace `tal` benutzt. Im Beispiel mit den `span`-Tags ist der Standard-Namespace XHTML, d.h. Sie müssen das `tal:`-Präfix explizit angeben, wenn Sie den Content-Reiter benutzen.

Beachten Sie, dass der Elementname sprechend ist und alles sein darf, was noch nicht im tal-Namespace definiert worden ist (z.B. `content` oder `replace`). Weil `tal:number` kein gültiges XHTML-Element ist, wird dieser Tag nicht angezeigt, wohl aber sein Inhalt, was das `omit-tag` überflüssig macht. Diese Technik wird in Plone sehr oft eingesetzt, um kürzeren, prägnanteren Code zu erhalten, in dem man Fehler einfacher suchen kann.

6.3.2 Einführung in Code-Säuberung

HTML Tidy ist ein hervorragendes Werkzeug zum Testen und Säubern von HTML-Code, das einige nützliche Aufgaben erledigen kann. Es existieren Versionen von HTML Tidy für alle Betriebssysteme. Sie können es unter *http://tidy.sourceforge.net* herunterladen. Für Windows-Benutzer: Finden Sie das passende Paket für Ihre Windows-Version, packen Sie die Datei `tidy.zip` aus, und fügen Sie `tidy.exe` zu Ihrem `PATH` hinzu (normalerweise Ihr Windows-Verzeichnis, z.B. `C:\WINNT`).

HTML Tidy sagt Ihnen, wenn es irgendwelche XHTML-Fehler in Ihrem Page Template gibt. Zu diesem Zweck kann ein Flag einen großen Unterschied ausmachen: `-xml`. Damit weiß HTML Tidy, dass es die Datei als XML behandeln und alle XML-Fehler berichten soll. Am Beispiel eines »defekten« Templates in Listing 6.4 können Sie einige Fehler sehen. Der Code ist nicht nur nicht eingerückt, sondern

es fehlen auch schließende Elemente, und auch die Verschachtelung stimmt nicht.

Listing 6.4: Ein Beispiel für ein defektes Page Template: bad_template.pt

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title></title>
</head>
<body>
<p>
<div>
Das ist schlechtes HTML,
XHTML oder XML...<a tal:contents="string: someUrl"></a>
</p>
<img>
Außerdem ist er nicht eingerückt!
</body>
</html>
```

Wenn Sie HTML Tidy über das Listing 6.4 laufen lassen, werden Sie die Fehler im Template sehen und schön eingerückten Code erhalten, wie in Listing 6.5 zu sehen ist.

Listing 6.5: Ausgabe von HTML Tidy

```
$ tidy -q -i bad_template.pt
line 11 column 1 - Warning: <img> element not empty or not closed
line 10 column 1 - Warning: missing </div>
line 10 column 39 - Warning: <a> proprietary attribute "tal\::contents"
line 11 column 1 - Warning: <img> lacks "alt" attribute
line 11 column 1 - Warning: <img> lacks "src" attribute
line 9 column 1 - Warning: trimming empty <p>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta name="generator" content=
  "HTML Tidy for Linux/x86 (vers 1st August 2003), see www.w3.org" />

  <title></title>
</head>
```

```
<body>
  <div>
    Das ist schlechtes HTML, XHTML oder XML...<a tal:contents=
    "string: someUrl"></a> <img />Außerdem ist er nicht eingerückt!
  </div>
</body>
</html>
```

Die Beschwerden über proprietäre Attribute können etwas nervig sein. Um zu prüfen, ob Ihr Page Template gültiges XML ist, übergeben Sie das Flag `-xml`. Die Ausgabe ist weniger ausführlich und weist lediglich auf die fehlenden Tags hin:

```
$ tidy -q -xml bad_template.pt
line 15 column 1 - Error: unexpected </body> in <img>
line 16 column 1 - Error: unexpected </html> in <img>
```

6.3.3 Syntax-Prüfungen durchführen

Wenn Sie ein Page Template im ZMI bearbeiten, führt Zope auf dem Dokument eine Syntax-Prüfung auf Dinge wie ungültige Tags durch. Falls ein Tag ungültig ist, wird ein Fehler bei dem Template angezeigt, während Sie es über das Web bearbeiten. Wenn Sie wie ich die meisten Ihrer Page Templates im Dateisystem schreiben (was ich in Kapitel 7 demonstrieren werde), so ist eine einfache Syntax-Prüfung des Page Templates wirklich sehr hilfreich. Listing 6.6 ist ein Python-Script in Ihrem Dateisystem, das unabhängig von Zope läuft.

Um es auszuführen, müssen Sie einen Python-Interpreter haben, und das Python-Modul `PageTemplate` muss importiert werden können. Damit `PageTemplate` in Ihren Python-Interpreter importiert werden kann, müssen Sie das Verzeichnis `Products` Ihrer Zope-Installation zu Ihrem Python-Pfad hinzufügen. Das können Sie auf mehrere Arten tun (siehe Anhang B).

Listing 6.6: Fehlersuche bei Page Templates

```
#!/usr/bin/python
from Products.PageTemplates.PageTemplate import PageTemplate
import sys

def test(file):
    raw_data = open(file, 'r').read()
    pt = PageTemplate()
    pt.write(raw_data)
    if pt._v_errors:
        print "*** Error in:", file
        for error in pt._v_errors[1:]:
```

```
            print error

if __name__=='__main__':
    if len(sys.argv) < 2:
        print "python check.py file [files...]"
        sys.exit(1)
    else:
        for arg in sys.argv[1:]:
            test(arg)
```

Für jede an das Script übergebene Datei kompiliert das ZMI das Page Template und sucht nach irgendwelchen TAL-Fehlern. Für die Datei `bad_template.pt` aus Listing 6.4 erhalten Sie einen Fehler:

```
$ python zpt.py /tmp/bad_template.pt
*** Error in: /tmp/bad_template.pt
TAL.TALDefs.TALError: bad TAL attribute: 'contents', at line 10, column 39
```

In diesem Fall beschwert es sich über die unrichtige Buchstabierung von `tal:content` in Form von `tal:contents`. Dies ist ein Fehler, den HTML Tidy nicht findet. Dummerweise wird die Verarbeitung beim ersten Syntax-Fehler abgebrochen. Wenn es mehrere Fehler gibt, wird nur der erste genannt, d.h., manchmal müssen Sie die Syntax mehrfach überprüfen.

6.4 Formulare verwenden

Formulare sind ein wichtiger Bestandteil einer Site, und fast jeder muss eine Methode zum Erstellen und Ändern von Formularen in seiner Plone-Site benutzen. Mit dem Formular-Framework von Plone können Sie die Validierung von Formularen ändern. Dieses Framework ist nicht ausschließlich für isolierte Formulare gedacht, die eine einfache Aufgabe erfüllen, z.B. ein Passwort abfragen, eine Anmeldung vornehmen etc. Das Framework funktioniert auch für alle Inhaltstypen bei Aufgaben wie der Bearbeitung eines Inhaltstyps, was ich später in den Kapiteln 11 bis 13 behandeln werde.

Alle einfachen Formulare verfügen über mindestens zwei Komponenten, die Sie bereits gesehen haben: ein Page Template-Objekt, um das Formular dem Benutzer anzuzeigen, und ein Script (Python)-Objekt, um die Ergebnisse zu parsen und irgendeine Aktion darauf durchzuführen.

Das Form Controller-Framework von Plone führt ein paar neue Objekttypen ein, die äquivalent zu den Typen sind, die Sie in diesem Kapitel gesehen haben. Dies sind das Controller Page Template-Objekt, das Controller Script(Python)-Objekt und das Controller Validator-Objekt. Zu diesen neuen Objekten gibt es äquivalente

Zope-Objekte, wie Sie in Tabelle 6.1 sehen. Diese neuen Objekte haben mehr Eigenschaften und agieren auf leicht andere Weise als die dazu äquivalenten Objekte.

Objekttyp	Äquivalentes Zope-Objekt
Controller Filesystem Page Template	Page Template
Controller Python Script	Python Script
Controller Validator	Python Script

Tabelle 6.1: Neue Objekttypen, die der Controller bietet

Um eines dieser Objekte mit dem ZMI zu hinzuzufügen, gehen Sie ins Dropdown-Menü und wählen seinen Namen aus.

Das Form Controller-Framework erzeugt eine Reihe von Ereignissen für ein Formular, die ein Benutzer dann definieren kann. Folgendes ist die Reihe der Ereignisse, wenn ein Formular ausgeführt wird:

1. Dem Benutzer wird ein Controller Page Template-Objekt angezeigt.
2. Der Benutzer füllt das Formular aus und schickt es ab.
3. Eine eventuelle Validierung des abgeschickten Inhalts wird ausgeführt.
4. Eine den Daten entsprechende Aktion (oft Erfolg oder Fehlschlag) wird ausgeführt.

Wenn diese Reihe von Ereignissen eintritt, wird ein Zustandsobjekt herumgereicht, das Informationen über den Objektstatus, den Erfolg irgendwelcher Validierungen und irgendwelche Meldungen enthält, die weitergegeben werden sollen.

Um zu zeigen, wie ein Formular validiert werden kann, gehen die folgenden Abschnitte diese Schritte durch. Anschließend werde ich ein vollständiges Beispiel im Abschnitt »E-Mail-Beispiel: E-Mail an den Webmaster schicken« zeigen.

6.4.1 Erstellen eines Beispielformulars und zugehörige Scripten

Am Anfang dieses Prozesses steht ein Formular. Obwohl es tatsächlich ein Controller Page Template-Objekt ist, ist es mit normalem TAL-Code geschrieben. Um ein Formular hinzuzufügen, wählen Sie im nun vertrauten Dropdown-Menü CONTROLLER PAGE TEMPLATE und geben ihm die ID **test_cpt**.

Ein Formular in Plone besteht eigentlich aus ziemlich viel Code, falls Sie alle verfügbaren Optionen benutzen möchten. Dieser Codeteil ist in Anhang B vollständig enthalten und wird im späteren Beispiel benutzt:

```
<form method="post"
      tal:define="errors options/state/getErrors"
      tal:attributes="action template/id;">
    ...
    <input type="hidden" name="form.submitted" value="1" />
</form>
```

Wenn Sie diesen Code betrachten, sollte Ihnen auffallen, dass einige kleine Unterschiede zu einer Art Standardformular existieren, damit der Code im Framework funktioniert. Zunächst ist das Formular so eingerichtet, dass es Daten an sich selbst schickt. Das ist *nicht* optional. Weiterhin existiert eine spezielle versteckte Variable namens `form.submitted`.

Das Controller Page Template-Objekt prüft die Anfragevariable auf den Wert `form.submitted`, um zu sehen, ob das Formular abgeschickt wurde. Wenn es stattdessen gerade erst aufgerufen worden ist, z.B. über einen Link, ist das *nicht* optional. Am Anfang des Formulars setzen Sie die Variable `error`. Dieses Dictionary kommt aus dem `state`-Objekt, das an die Templates übergeben wird. Das `state`-Objekt ist in diesem System das gleiche Objekt für alle Templates und Scripten.

Validierer erstellen

Sobald der Benutzer den SENDEN-Button auf Ihrem Formular anklickt, gehen die Daten durch die Validierer und werden von ihnen geprüft. Validierer sind optional. Daten müssen nicht validiert werden, aber eine Anwendung sollte das natürlich bei Bedarf tun. Der VALIDATOR-Reiter für ein Controller Page Template-Objekt enthält einen Link zu den möglichen Validierern.

Ein Validierungsscript ist identisch mit einem normalen Script(Python)-Objekt, das eine Extravariable, `state`, hat. Mit dieser Variablen können Sie Validierungsergebnisse übergeben. Listing 6.7 zeigt ein einfaches Validierungsscript, das überprüft, ob eine Zahl eingegeben wurde.

Listing 6.7: Validierung einer Zahleneingabe

```
##title=Ein Script zum Überprüfen einer Zahleneingabe
##parameters=
num = context.REQUEST.get('num', None)
try:
    int(num)
except ValueError:
    state.setError("num", "Keine Zahl", new_status="failure")
except TypeError:
    state.setError("num", "Keine Zahl eingegeben.", new_status="failure")
if state.getErrors():
```

```
    state.set(portal_status_message="Bitte Fehler korrigieren.")
return state
```

Dieses `state`-Objekt enthält einfache Angaben darüber, was in der Validierungskette passiert ist. Es speichert für jedes Feld die Fehler, den Status und alle anderen Werte. Wenn z.B. die eingegebene Zahl nicht in eine Ganzzahl umgewandelt werden kann, setzen Sie den Status auf Fehlschlag und geben eine Fehlermeldung für das Feld mit der Methode `setError` aus. Später wird die Fehlermeldung für dieses Feld angezeigt. Am Ende des Scripts werden alle bisher zurückgegebenen Fehler mit der Methode `getErrors` übergeben.

Um das obige Script hinzuzufügen, klicken Sie auf PORTAL_SKINS, dann auf CUSTOM und wählen im Dropdown-Menü CONTROLLER VALIDATOR. Geben Sie ihm die ID **test_validator**. Nun können Sie zum VALIDATION-Reiter Ihres Controller Page Template-Objekts zurückgehen und einen Zeiger auf dieses Validierungsscript hinzufügen, wie in Abbildung 6.7 zu sehen ist.

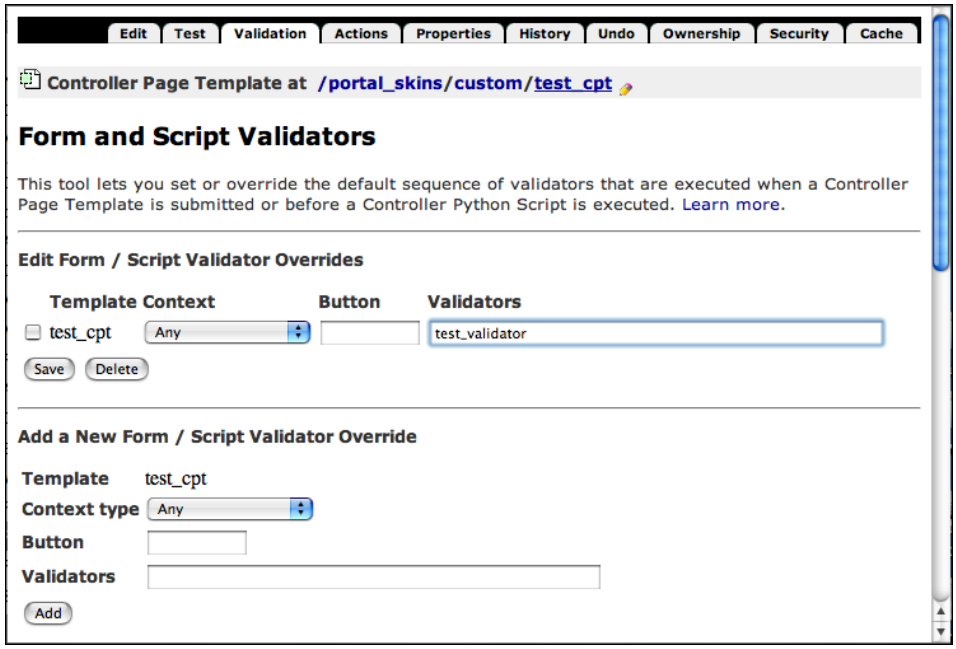


Abbildung 6.7: Hinzufügen von test_validator zum Controller Page Template-Objekt

Bei einer Validierung haben Sie einige Optionen zur Auswahl. Im Beispiel habe ich diese ignoriert, weil sie nicht relevant sind, aber hier ist die Liste dieser Optionen:

- **contextType**: Dies ist der Typ des context-Objekts, falls vorhanden, in dem das Template ausgeführt wird. Dies ist eine Abkürzung zum Inhaltstyp des context-Objekts. Wenn Sie wollten, dass nur diese Validierung bei einem Link ausgeführt werden soll, könnten Sie diesen Wert auf LINK setzen.
- **button**: Dies ist der Button, falls vorhanden, der angeklickt wird, um das Formular abzuschicken. Sie könnten verschiedene Buttons in einem Formular haben (z.B. einen SENDEN- und einen ABBRECHEN-Button). Jeder dieser Buttons könnte dann auf eine andere Aktion abgebildet werden. Ein Klick auf ABBRECHEN würde Sie an einen Ort bringen und ein Klick auf SENDEN an einen anderen.
- **validators**: Dies ist eine mit Kommata getrennte Liste von Validierern, d.h. Controller Validator-Objekten, die das Template verwenden wird. Im vorigen Beispiel haben Sie die Validierer-ID `test_validator` benutzt.

Hinweis

Verwenden Sie Controller Validator-Objekte statt Script(Python)-Objekte, wenn Sie Validierungsscripts schreiben. Controller Validator-Objekte entsprechen gewöhnlichen Script(Python)-Objekten, haben aber einen zusätzlichen ACTIONS-Reiter im ZMI.

Aktionen angeben

Aktionen werden ausgeführt, nachdem die Validierer beendet worden sind, und hängen vom Status ab, den die Validierer zurückgeben. Der ACTIONS-Reiter für ein Controller Page Template-Objekt zeigt alle Aktionen des entsprechenden Page Templates an. Aktionen können Sie mit der gleichen Art von Spezialisierungsoptionen angeben, die zuvor schon mit einem Webformular beschrieben wurden (siehe Abbildung 6.8).

Für die eigentliche Ergebnisaktion haben Sie folgende vier Möglichkeiten:

- **redirect_to**: Damit wird eine Umleitung zu der im Argument (einem TALES-Ausdruck) angegebenen URL erzielt. Die URL kann entweder absolut oder relativ sein.
- **redirect_to_action**: Hier erfolgt die Umleitung auf die im Argument (einem TALES-Ausdruck) angegebene Aktion zu dem aktuellen Inhaltsobjekt (z.B. `string:view`). Zu diesem Zeitpunkt habe ich Aktionen noch nicht behandelt, aber jedes Inhaltsobjekt verfügt über Aktionen wie Anzeigen und Bearbeiten. Aktionen zu einem Objekt werden in Kapitel 11 behandelt.

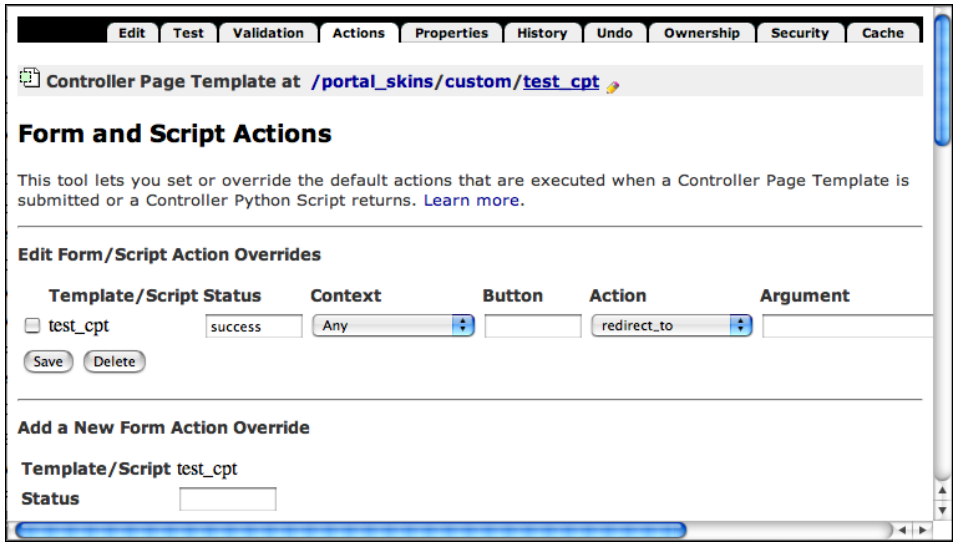


Abbildung 6.8: Eine Aktion hinzufügen

- **traverse_to**: Führt eine Traversierung zu der im Argument (einem TALES-Ausdruck) angegebenen URL durch. Die URL kann entweder absolut oder relativ sein.
- **traverse_to_action**: Führt eine Traversierung zu der Aktion durch, die im Argument (einem TALES-Ausdruck) zu dem aktuellen Inhaltsobjekt angegeben ist (z.B. `string:view`).

Ein Beispiel hierfür ist, wenn Sie bei erfolgreichem Ausfüllen eines Formulars zu einem Controller Python Script-Objekt traversieren, das Sie geschrieben haben und das das Ergebnis des Formulars verarbeitet. Wenn die Seite fehlschlägt, traversieren Sie zum Template zurück und zeigen den Fehler an.

Der Unterschied zwischen einer Weiterleitung (Redirect) und einer Traversierung ist der, dass die Weiterleitung eine HTTP-Weiterleitung ist, die an den Browser des Benutzers gesendet wird. Der Browser verarbeitet sie und leitet den Benutzer zur nächsten Seite weiter. Daher verlieren die Weiterleitungsaktionen alle in der ursprünglichen Anfrage übergebenen Werte. Falls Sie den Inhalt des ursprünglichen Formulars untersuchen müssen, ist das nicht der beste Ansatz. Ich empfehle stattdessen die Traversierung zu den Optionen. Das Ergebnis ist das gleiche, aber die Traversierung erledigt das alles auf dem Server. Dabei werden die Anfragevariablen erhalten, die Sie dann in Scripten untersuchen können.

6.4.2 E-Mail-Beispiel: E-Mail an den Webmaster schicken

Nun werden Sie ein echtes Beispiel sehen, das Sie im Rest dieses Kapitels erstellen werden. Eine häufige Anforderung ist ein spezielles Formular, das E-Mails an den Webmaster schickt. Ein solches Formular werden Sie in den folgenden Abschnitten erstellen. Die vollständigen Scripten, das Page Template und der zugehörige Code sind in Anhang B verfügbar. Wenn Sie all das nicht eintippen möchten, können Sie das Beispiel online auf der Buch-Website sehen. Sie können es auch als komprimierte Datei von der Plone-Buch-Website (*http://plone-book.agmweb.ca*) herunterladen. Das heißt, Sie können es einfach installieren und ausprobieren. Bei diesem Beispiel gibt es nur zwei Felder im Formular: die E-Mail-Adresse der Person, die das Formular abschickt, und ihren Kommentar. Bei diesem Formular ist die E-Mail-Adresse der Person notwendig, damit Sie auf ihre Kommentare antworten können.

Das Formular erstellen

Das Formular ist der größte und komplizierteste Teil dieser Prozedur, vor allem, weil man viel Arbeit in die Fehlerbehandlung investieren muss. Dieses Formular ist ein Controller Page Template-Objekt namens `feedbackForm`. Um sicher zu sein, dass es vom Haupt-Template umfasst wird, beginne ich das Formular mit der üblichen Methode:

```
<html
    xmlns="http://www.w3.org/1999/xhtml"
    xml:lang="en-US"
    lang="en-US"
    i18n:domain="plone"
    metal:use-macro="here/main_template/macros/master">
  <body>
    <div metal:fill-slot="main"
         tal:define="errors options/state/getErrors;">
```

Ein Zusatz hierbei ist `errors options/state/getErrors`, womit alle eventuellen Fehler für eine spätere Verwendung in die lokale Variable `errors` abgelegt werden.

Wegen der Bedingung, dass das Formular sich selbst aufrufen soll, setzen Sie diese Aktion in TAL mit dem Ausdruck `template/id`. Dieser Pfad extrahiert die ID des Templates und fügt sie in die Aktion ein. Das heißt, dieser Pfad funktioniert immer, sogar dann, wenn Sie das Template umbenennen. Beachten Sie, dass Sie auch die zuvor gesehenen `i18n`-Tags hinzufügen, um zu gewährleisten, dass dieses Formular später lokalisiert werden kann:

```
<form method="post"
      tal:attributes="action template/id;">
```

```
<legend i18n:translate="legend_feedback_form">
    Website Feedback
</legend>
```

Der folgende Code ist der Anfang der Zeile für die E-Mail-Adresse. Hierbei definieren Sie eine Variable namens `error_email_address`, die auf einen Fehler-String gesetzt ist, falls es einen passenden String im Dictionary `errors` gibt. Dieser Fehlerwert wird vom Validierer erzeugt, falls ein Fehler vorkommen sollte:

```
<div class="field"
     tal:attributes="class python:test(error_email_address,
                                       'field error', 'field')">
     tal:define="error_email_address errors/email_address|nothing;">
```

Folgendes ist die Bezeichnung für das E-Mail-Adressenfeld. Darin fügen Sie ein `div` für den Hilfetext ein. Das `span`-Element wird zu dem nun vertrauten roten Punkt neben der Bezeichnung, damit der Benutzer weiß, dass das Feld ausgefüllt werden muss:

```
<label i18n:translate="label_email_address">Your email address</label>
<span class="fieldRequired" title="Required">(Required)</span>
<div class="formHelp"
     i18n:translate="label_email_address_help">
     Enter your email address.
</div>
```

Als Nächstes fügen Sie das eigentliche Element hinzu:

```
    <div tal:condition="error_email_address">
        <tal:block i18n:translate=""
                   content="error_email_address">Error
        </tal:block>
    </div>
    <input type="text" name="email_address"
           tal:attributes="tabindex tabindex/next;
                           value request/email_address|nothing" />
</div>
```

Zu Beginn dieses Blocks testen Sie, ob es einen Fehler gibt. Wenn ja, hat die Klasse des Elements zu `field error` gewechselt. Diese Klasse zeigt einen hübschen orangefarbenen Kasten um das Feld herum. Wenn für dieses Feld ein Fehler aufgetreten ist (was Sie bereits getestet haben), wird die entsprechende Meldung angezeigt. Schließlich zeigen Sie das Formularelement an, und wenn es schon einen Wert für `email_address` in der Anfrage gibt, füllen Sie das Formularelement mit diesem Wert.

Das Werkzeug `tabindex` ist recht nützlich in Plone. Es enthält eine fortlaufende Nummer, die für jedes Element inkrementiert wird und einen neuen Wert für den HTML-`tabindex` setzt. Das ist eine nette Eigenschaft der Benutzerschnittstelle, denn dadurch kann jedes Formularelement gefahrlos herumgereicht werden, ohne dass man sich diese `tabindex`-Zahlen merken muss, weil das schon automatisch geschieht.

Das ist eine Menge Arbeit für ein Element, aber es ist überwiegend Code nach dem immer gleichen Schema, den Sie einfach kopieren oder ändern können. Den Rest des Formulars finden Sie in Anhang B.

Einen Validierer erstellen

In dem Beispiel gibt es nur ein notwendiges Element (die E-Mail-Adresse), d.h., es gibt ein einfaches Stück Python namens `validEmail.vpy`, das die Arbeit macht. Der Inhalt dieses Scripts sieht wie folgt aus:

```
email = context.REQUEST.get('email_address', None)
if not email:
    state.setError('email_address', 'Email is required',
                   new_status='failure')
if state.getErrors():
    state.set(portal_status_message='Please correct the errors.')
return state
```

Wenn keine E-Mail-Adresse gefunden wird, fügt dieses Script einen Fehler mit dem Schlüssel `email_address` und einer Meldung zum Fehler-Dictionary hinzu. Dieser Schlüssel wird im Page Template benutzt, um zu prüfen, ob es in diesem speziellen Feld einen Fehler gab.

Das Script verarbeiten

Dieses Beispiel verwendet ein einfaches E-Mail-Script, das die (bereits validierten) Werte bekommt und daraus eine E-Mail macht. Das ist ein Controller Python Script-Objekt. Es ähnelt dem normalen Script (Python)-Objekt, mit dem Unterschied, dass es eine zusätzliche Variable namens `state` hat und Sie ihm, wie beim Controller Page Template, Aktionen angeben können, wenn es Erfolg hat:

```
mhost = context.MailHost
emailAddress = context.REQUEST.get('email_address')
administratorEmailAddress = context.email_from_address
comments = context.REQUEST.get('comments')

# the message format, %s will be filled in from data
message = """
From: %s
To: %s
```

```
Subject: Website Feedback

%s
URL: %s """

# format the message
message = message % (
    emailAddress,
    administratorEmailAddress,
    comments,
    context.absolute_url())

mhost.send(message)
```

Nun haben Sie ein einfaches Script zum Verschicken von E-Mail gesehen. Das ist ein gängiges Script, das Sie immer wieder sehen werden. Im Prinzip bekommt dabei das `MailHost`-Objekt in Plone eine E-Mail-Adresse als String, sofern es die RFC-Spezifikation (Request for Comment) für E-Mails erfüllt, die `From`- und `To`-Adressen enthält.

In dieser E-Mail nehmen Sie die Administrator-Adresse, die Sie in den Portal-Einstellungen angegeben haben, und schicken die E-Mail an diese Person. Der einzige zusätzliche Teil in diesem Script besteht darin, dass außerdem auch der Zustand gesetzt wird. Damit wird eine Meldung gesetzt, die dem Benutzer ein Feedback gibt:

```
screenMsg = "Comments sent, thank you."
state.setKwargs( {'portal_status_message':screenMsg} )
return state
```

Verknüpfen aller drei Teile

Im Moment existieren jedoch drei separate Einheiten: ein Formular, ein Validierer und ein Aktionsscript. Diese müssen miteinander verbunden werden, um die ganze Kette zu bilden, damit Sie wieder zum Controller Template-Objekt zurückkommen. Klicken Sie auf den VALIDATOR-Reiter, und geben Sie einen neuen Validierer ein, der auf das Script `validEmail` zeigt. Sie werden auch eine Erfolgsaktion für den Fall hinzufügen, dass die Verarbeitung korrekt ist, um zu dem Script `sendEmail` zu traversieren. Bei dem `sendEmail`-Script können Sie nun eine weitere Traversierung zurück zu `feedbackForm` hinzufügen, damit der Benutzer nach einem erfolgreichen `sendEmail` wieder zur ursprünglichen Seite zurückgelangt.

Hinweis

In Plone gibt es ein wesentlich vollständigeres E-Mail-Validierungsscript namens `validate_emailaddr`, das prüft, ob die E-Mail das richtige Format hat. Wenn Sie lieber dieses Script benutzen möchten, können Sie den Validierer auf dieses Script zeigen lassen.

Das war's, Sie sind fertig! Nun sollten Sie in der Lage sein, das Formular auf der Buch-Website zu testen. Um es noch einfacher zu machen, habe ich einen Feedback-Reiter erstellt, der auf das Template `feedbackForm` zeigt, und von dort können Sie mir nun Feedback zu diesem Buch geben!

7 Das Look-and-Feel von Plone anpassen

In den beiden vorangegangenen Kapiteln habe ich einige der Kernkomponenten von Plones Benutzerschnittstelle behandelt, inklusive Script(Python)-Objekten und Page Templates. Nun ist es Zeit, die Details bei der Gestaltung des Look-and-Feel einer Plone-Site zu behandeln. Dieses Kapitel enthält Objekte aus vorangegangenen Kapiteln und führt auch einige neue ein.

Zu Beginn werde ich die Schlüsseldefinitionen und jene Plone-Elemente behandeln, die eine Site ausmachen. Ich werde Begriffe definieren, die Sie vielleicht schon gehört haben, z.B. *Skins* und *Ebenen*. Dann werde ich die Anpassung der Benutzerschnittstelle von Plone behandeln – mit besonderem Augenmerk auf den Möglichkeiten, über die ein Site-Entwickler mit Cascading Style Sheets (CSS) verfügt. Ich werde die wichtigsten Variablen durchgehen und Ihnen zeigen, wie Sie sie ändern können. Dann betrachte ich nochmals die Anpassung von Schleifen und Skin-Elementen, wobei alle Themen wieder auftreten werden, die ich in den drei vorangegangenen Kapiteln kurz behandelt habe.

Anschließend zeige ich Ihnen, wie man eine neue Skin erstellt, und erkläre die Techniken, mit denen man all das im Dateisystem entwickeln kann.

Und schließlich beende ich dieses Kapitel mit einer Beispiel-Site, nämlich der Maestro-Site, auf der die NASA Daten über die Mars-Rover bereitgestellt hat. Dies ist eine Plone-Website mit hohen Datentransferraten, und die Skin bietet eine hervorragende Fallstudie für die Anpassungen einer Site. Dieses sehr reale Beispiel dafür, wie man eine Plone-Site anpassen und ändern kann, wird Ihnen eine große Hilfe dabei sein, das Gleiche bei Ihrer eigenen Site durchzuführen.

7.1 Einführung in Plone-Skins

Wenn Plone ein Dokument anzeigt, wird der Inhalt dieses Dokuments mit der nunmehr vertrauten grün-blauen Plone-Schnittstelle angezeigt. Eine *Skin* (in der deutschen Plone-Lokalisierung auch *Aussehen* genannt) bestimmt ganz genau,

wie dieses Dokument dem Benutzer angezeigt wird, inklusive Bilder und Styles um den Inhalt herum. Eine Skin gruppiert Elemente, hüllt dieses Stück Inhalt ein und stellt es auf eine bestimmte Weise dar.

Um die Darstellung zu generieren, die ein Benutzer sieht, hat eine Skin viele Elemente, inklusive eines statischen Teils, z.B. ein Bild, und dynamischer Teile wie Scripts. Das Feedback-Formular im vorigen Kapitel war ein Beispiel für das Hinzufügen einiger Elemente zu einer Skin, um neue Teile einer Skin zu erstellen. Dieses Beispiel enthielt eine Logik in Form eines Python(Script)-Objekts und neue Seiten in Form von Page Templates. Diese Teile haben Sie zu einer Skin hinzugefügt, damit ein Benutzer mit dem Formular interagieren konnte.

Von einer Skin können Sie so viel oder so wenig wiederverwenden, wie Sie möchten, um eine neue Skin mit Plone zusammenzustellen. Die Skin können Sie mit kleinen Korrekturen oder mit umfangreichen Änderungen erstellen, wie Sie z.B. auf Community-Sites wie *http://www.zopezen.org* und *http://www.zopera.org* sehen können. Jede Plone-Site muss über mindestens eine Skin verfügen, die als Standard verwendet wird, aber sie kann so viele haben, wie der Site-Entwickler das möchte. Ein Benutzer kann gelegentlich zwischen verschiedenen Skins wechseln, falls der Site-Entwickler das erlauben möchte, auch wenn mir scheint, dass das selten der Fall ist.

Die Standard-Skin in Plone ist diejenige, die Sie auf einer Plone-Site wie *http://www.plone.org* sehen, mit der bekannten blau-grünen Schnittstelle. Aber Plone muss in keinster Weise so aussehen oder auch nur im Entferntesten als Plone-Site zu erkennen sein. Sein Aussehen liegt ganz in Ihrer Hand. Nehmen Sie z.B. die Site-Liste unter *http://www.plone.org/about/sites*. Diese Sites haben alle ein eigenes Aussehen und eine eigene Benutzerführung. In den meisten Fällen können diese Sites leicht zwischen Skins umschalten, um den Benutzern ein anderes Aussehen anzubieten. Andere Sites machen internen Gebrauch von der mächtigen und flexiblen Schnittstelle von Plone, um Inhalte zu erstellen und zu bearbeiten, während externe Benutzer ein völlig anderes Aussehen der Site zu Gesicht bekommen.

Auf Mailinglisten habe ich viele Fragen dazu gelesen, ob Plone wie eine Plone-Site aussehen muss? Kann es für einen Benutzer so und für einen anderen anders aussehen? Kann es wie meine Firmen-Site aussehen? Die Antwort auf diese Fragen lautet »ja«: Nur Ihre Vorstellungsgabe setzt Ihnen Grenzen (und die Menge an Zeit, die Sie in die Anpassung Ihrer Site investieren können).

7.1.1 Ebenen in einer Skin benutzen

Ein Skin wird logisch in eine Ansammlung von Templates und Scripts namens *Ebenen* (*layers*) unterteilt. Das Ändern dieser individuellen Sammlungen erlaubt

es dem Benutzer, Komponenten auf einfache Weise zu einer Skin hinzuzufügen oder daraus zu entfernen. Die Ebenen werden in einer Skin durch eine hierarchische Liste von Ordnern dargestellt. Jede Ebene entspricht einem Ordnernamen, und jeder Ordner enthält die Skin-Elemente.

Eine Skin kann z.B. folgende Ebenen haben:

```
custom, gruf, plone_ecmascript, plone_wysiwyg ...
```

Die Reihenfolge der Ebenen in dieser Liste ist der bestimmende Faktor dafür, wie Plone die Elemente findet. Wenn ein Element wie `logo.jpg` von der Skin verlangt wird, geht die Skin die Ebenen durch, um das Element zu finden. Die Skin beginnt bei der ersten ihr zugeordneten Ebene (in diesem Beispiel `custom`). Wenn die Skin das Element dort nicht finden kann, wird die Suche in der zweiten Ebene fortgesetzt (hier `gruf`). Das setzt sich so lange fort, bis sie das gesuchte Element findet. Sonst wird ein Fehler 404 ausgelöst und an den Browser zurückgegeben.

Ein ähnliches Konzept dazu ist die Verwendung der Umgebungsvariablen PATH auf den meisten Betriebssystemen. Bei der Eingabe eines Befehls oder bei der Suche nach einem Programm durchsucht das Betriebssystem die Verzeichnisse, wie sie in der Umgebungsvariablen PATH angegeben sind. Etwas Ähnliches passiert bei Ebenen, wo diese auch nacheinander durchsucht werden, um das Element zu finden.

Dadurch, dass höhere Ebenen Vorrang vor niedrigeren haben, verfügen Entwickler und Administratoren nun über die Möglichkeit, ihre Site mit Hilfe von Ebenen anzupassen. Wenn Sie ein bestimmtes Element einer Plone-Skin nicht mögen, können Sie das Element anpassen, indem Sie es eine Ebene nach oben bewegen und das Ergebnis anpassen. Sie können Ihre Skins und Ebenen in Plone mit dem Werkzeug `portal_skins` sortieren, das ich als Nächstes behandle.

7.1.2 Skins mit dem Werkzeug portal_skins verwalten

Das `portal_skins`-Werkzeug in Plone können Sie dazu benutzen, das Verhalten von Skins und Ebenen zu definieren. Es verfügt auch über eine API (Application Programming Interface) zum Erstellen und Benutzen von Skins.

Um auf das `portal_skins`-Werkzeug zuzugreifen, gehen Sie ins ZMI (Zope Management Interface) und klicken auf PORTAL_SKINS. Im ZMI sehen Sie dann zwei wichtige Bildschirme. Der erste, der CONTENTS-Reiter , zeigt alle Ordner und Dateisystem-Verzeichnisansichten (*file system directory views*, FSDVs) in diesem Werkzeug an (siehe Abbildung 7.1).

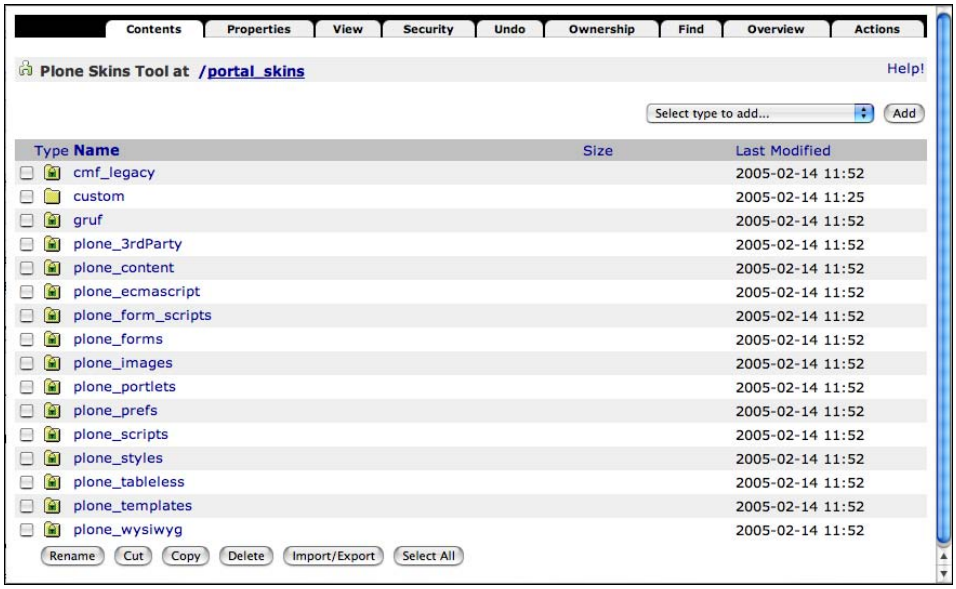


Abbildung 7.1: Inhalt des portal_skins-Werkzeugs in einer Plone-Standardinstallation

Die Ordner und Dateisystem-Verzeichnisansichten unter dem CONTENTS-Reiter sind nicht standardmäßig Ebenen, aber Sie können sie nun in Ebenen verwandeln. Der zweite wichtige Bildschirm, der PROPERTIES-Reiter, zeigt alle Skins und Ebenen an, die Sie in Ihrer Plone-Site definiert haben (siehe Abbildung 7.2).

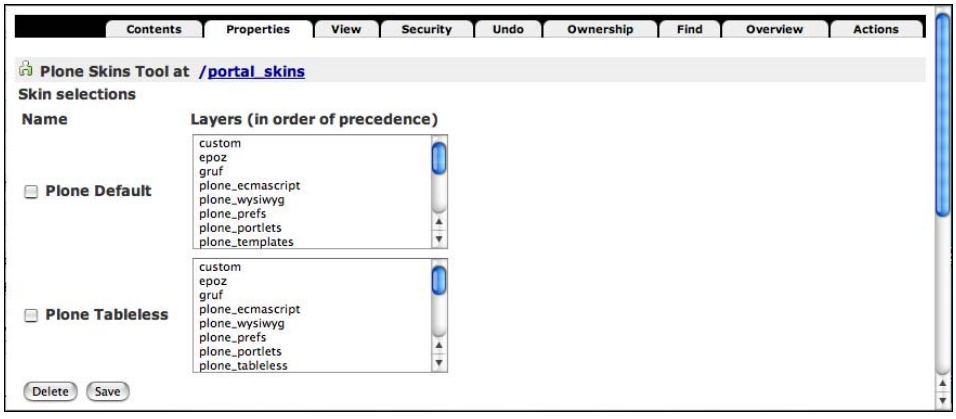


Abbildung 7.2: Skins und Ebenen in einer Plone-Standardinstallation

Wie Abbildung 7.2 zeigt, ist die Liste dieser Ebenen ziemlich lang. Auch wenn sie etwas einschüchternd wirken mag, gibt diese große Anzahl von Ebenen dem Entwickler ein großes Maß an Flexibilität und Wiederverwendbarkeit. Jede Skin wird links angezeigt. Dazu gibt es rechts einen Textbereich, in dem alle zugehörigen Ebenen angezeigt werden. Wie ich schon sagte, durchsucht Plone bei der Suche nach Elementen die Ebenen von oben nach unten. Jede Ebene entspricht dem Namen eines Ordners oder FSDV unter dem CONTENTS-Reiter. In Abbildung 7.2, können Sie ein Verzeichnis namens `plone_ecmascript` sehen, und in Abbildung 7.1 sehen Sie das passende FSDV-Objekt.

Ein FSDV ist ein neues Objekt, das eine nützliche Fähigkeit in Plone bietet: es bietet direkten Zugriff auf Skin-Elemente, die im Dateisystem, anstatt wie üblich in Zopes Objektdatenbank, definiert sind. Dadurch machen sie die Entwicklung und Anpassung leichter. Durch das direkte Lesen der Objekte aus dem Dateisystem ist es für Entwickler viel leichter, den Code zu schreiben und zu bearbeiten, aus dem die Site generiert wird. Wenn Sie Plone installieren, wird die Skin ins Dateisystem geschrieben. Wenn Sie ein Objekt anpassen, legen Sie eine lokale Kopie davon in Ihrer Plone-Datenbank an. Ein FSDV erlaubt eine klare Trennung zwischen Code, den Sie aus dem Web heruntergeladen haben, und Code, der in Ihrer lokalen Instanz angepasst wurde.

Plone 2 wird mit zwei Skins ausgeliefert: *Plone Default* und *Plone Tableless*. Plone Default verwendet Tabellen bei der Darstellung des Haupttextes, flankiert von zwei Tabellenzellen auf beiden Seiten, die die linken bzw. rechten Slots enthalten. Aus Gründen der Browser-Kompatibilität ist das die Standardeinstellung. Wenn Sie jedoch zu Plone Tableless umschalten, bekommen Sie eine Skin, die gleich aussieht, aber bei der Darstellung der Seiten völlig ohne Tabellen auskommt, was Ihnen als Site-Entwickler mehr Flexibilität gibt. Beim Schreiben dieses Buches war die Plone Tableless-Skin auf manchen Browsern wie dem Internet Explorer vielleicht etwas problematisch. In Zukunft, so hoffe ich, wird die Plone Tableless-Skin der Standard werden.

Um die Skin zu wechseln, scrollen Sie ans untere Formularende, wo Sie den Wert für Default Skin sehen, und wählen die Skin PLONE DEFAULT aus der Liste aus. Falls Sie die Option SKIN FLEXIBILITY wählen, können Benutzer ihre eigenen Skins im Abschnitt MEINE EINSTELLUNGEN wählen.

Zurück im CONTENTS-Reiter des `portal_skins`-Werkzeugs können Sie sehen, dass einige der Ordner, z.B. `custom`, Standardordner aus Zope sind. Diese haben das normale Ordner-Icon. Andere, z.B. `plone_images`, sind FSDVs, die in Bereiche des Dateisystems zeigen. Sie haben ein Ordner-Icon mit einem grünen Schloss darin. Dieses Schloss zeigt an, dass Sie keine Elemente über das Web, sondern nur über das Dateisystem in einem FSDV hinzufügen oder bearbeiten können.

Um zu sehen, wo die Dateien zu einem FSDV sich auf Ihrer Festplatte aufhalten, klicken Sie auf den PROPERTIES-Reiter des FSDV. Klicken Sie z.B. vom CONTENTS-Reiter des `portal_skins`-Werkzeugs auf PROPERTIES, und Sie sehen den im System verwendeten Dateipfad `CMFPlone/skins/plone_images`. Dieser Pfad ist der Ort dieses Verzeichnisses im Dateisystem, relativ zum Instanzursprung, den Sie bei der Installation angegeben haben. Da Sie Dateien über das Web im FSDV oder im Dateisystem sehen können, können Sie auf beide Arten lesend darauf zugreifen. Im Allgemeinen ist es angenehmer und leichter, Dateien im Dateisystem anzuzeigen, weswegen ich als Datei einen Pfad im Dateisystem bezeichnen werde, auf den man mit den gewohnten Werkzeugen zugreifen kann.

7.2 Skins anpassen

Sie haben gesehen, wie Skins und Ebenen miteinander interagieren. Nun werden Sie sehen, wie Sie eine Plone-Site anpassen können. Ich beginne mit dem Beispiel aus Kapitel 4, in dem Sie gelernt haben, wie man das Logo anpassen kann. Mit Ihrem neuen Wissen über die Funktionsweise von Skins können Sie hier anknüpfen, um auch die Skin anzupassen. Dann mache ich damit weiter, dass ich Ihnen die Mächtigkeit vom Plone-CSS zeige und demonstriere, wie Sie es anpassen können. Zum Abschluss behandle ich das Haupt-Template, das Sie in früheren Kapiteln gesehen haben, und werde dessen Elemente detailliert beschreiben.

7.2.1 Logo anpassen, zweiter Teil

In Kapitel 4 haben Sie gelernt, wie Sie das Logo in der oberen linken Ecke einer Plone-Site anpassen können, aber ich habe nicht erklärt, was dabei wirklich passiert. Dieser Abschnitt holt das nach.

Das Bild `logo.jpg` ist jenes Bild, das in der oberen linken Ecke jeder Seite erscheint. Sie werden nun sehen, was passiert, wenn ein Browser versucht, diese Seite darzustellen. Nachdem Plone die Anfrage zu diesem Bild erhält, durchsucht es die Ebenen nach `logo.jpg`. In einer Plone Default-Site ist dies das Element in `plone_images` namens `logo.jpg`. Weil dies, wie ich zuvor erklärt habe, ein FSDV ist, können Sie das Bild nicht über das Web ändern. Um Ihre Site vor zukünftigen Änderungen zu schützen, wollen Sie es aber auch nicht im Dateisystem ändern. Betrachten Sie also einmal genauer, was der CUSTOMIZE-Button tut. Wenn Sie diesen Button noch einmal anschauen, sehen Sie links daneben ein Dropdown-Menü mit Ordnern unter dem CONTENTS-Reiter des `portal_skins`-Werkzeugs.

Hinweis

Die aufgelisteten Ordner existieren in der Zope-Datenbank. FSDVs werden im Dropdown-Menü nicht aufgeführt. Standardmäßig zeigt es nur Ordner an.

Mit einem Klick auf den CUSTOMIZE-Button wird eine lokale Kopie des Elements in dem Ordner angelegt, der im Dropdown-Menü gewählt wurde. Per Voreinstellung ist dies der Ordner `custom`. Das heißt, nun haben Sie eine Kopie im `custom`-Ordner. Wenn Plone das Element `logo.jpg` sucht, greift es auf die Version im `custom`-Ordner zu. Wenn Sie wieder die Ebenen der Plone Default-Skin betrachten, sehen Sie, dass der `custom`-Ordner die oberste Ebene der Skin ist. Da beim Aufruf von `logo.jpg` das Bild in der `custom`-Ebene gefunden wird, wird dieses neue `logo.jpg` dargestellt.

Die Ablage angepasster Elemente in den `custom`-Ordner ist der schnellste Weg, Ihre Plone-Site abzuwandeln. Dieser Ordner ist ein Standardordner in Plone, d.h., Sie können darin so viele Elemente ablegen und frühere Versionen überschreiben, wie Sie möchten.

7.2.2 Einführung in die Cascading Style Sheets von Plone

Die visuelle Darstellung einer Plone-Site in einem Browser wird fast vollständig mit Hilfe von CSS zusammengestellt. Der vielleicht einfachste Weg, um zu sehen, was das CSS für eine Plone-Site leistet, ist der, die Abbildungen 7.3 und 7.4 zu vergleichen. Die erste zeigt Plone mit Stylesheets, und die zweite zeigt es ohne Stylesheets.

Tipp

Falls Sie diesen Effekt reproduzieren möchten, schalten Sie Stylesheets in Ihrem Browser ab. Der Internet Explorer lässt Sie das nicht ohne weiteres machen, aber in Firefox (*http://www.mozilla.org/products/firefox/*), dem Mozilla-basierten Open-Source-Browser, können Sie das ganz leicht tun. Wählen Sie in Firefox TOOLS – WEB DEVELOPER – DISABLE – DISABLE STYLES. Wegen seiner großen Anzahl von CSS- und anderen Entwicklerwerkzeugen ist Firefox für viele Plone-Entwickler der Browser der Wahl.

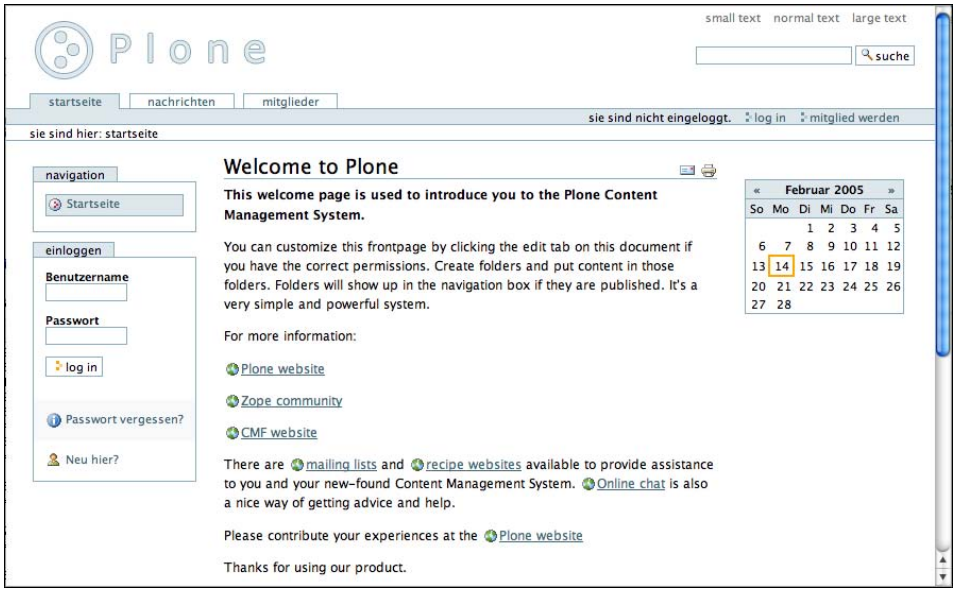


Abbildung 7.3: Plone mit Stylesheets

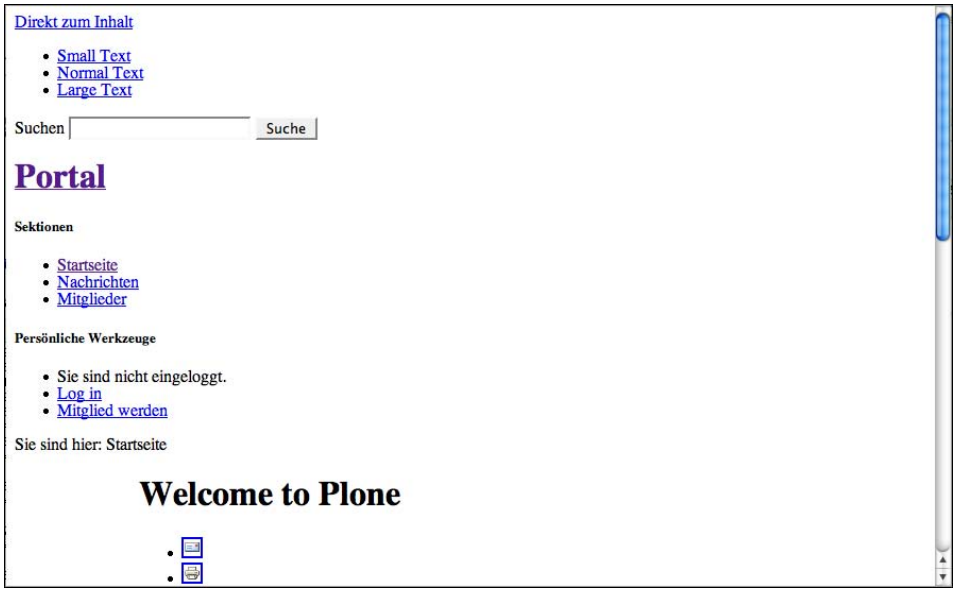


Abbildung 7.4: Plone ohne Stylesheets

Der Unterschied ist, gelinde gesagt, bestechend. CSS übernimmt nicht nur die visuelle Darstellung von Seiten, sondern auch deren Layout. Durch eine Änderung im CSS können Sie diese visuelle Darstellung und das Layout einer Plone-Site ändern (innerhalb des Rahmens von CSS).

Dass die Darstellung von Plone von CSS durchgeführt wird, ist eine beeindruckende Leistung von vielen talentierten Entwicklern von Benutzerschnittstellen. zu den Vorteilen eines CSS-Layouts zählen unter anderem:

- CSS bietet eine Trennung zwischen der Darstellung und den Templates, die die Darstellung generieren.
- Sie können eine große Anzahl von Änderungen vornehmen, ohne die darunter liegenden Templates anfassen zu müssen. Man braucht nur einen erfahrenen CSS-Entwickler.
- Mit CSS wird die Site schneller, weil kleinere Dateien übertragen werden. Jede HTML-Datei ist kleiner, da das Layout der Site nicht als HTML-Auszeichnungen, sondern als CSS vorliegt, das dann im Cache gespeichert werden kann.
- CSS ermöglicht die Anpassung des Look-and-Feel, ohne die zugrunde liegende Arbeit an der Zugänglichkeit der Site zu ruinieren.

Exkurs: Code-Ebenen

Wenn eine Plone-Seite dargestellt wird, sind mindestens drei Ebenen von Code an der Erzeugung der Seite beteiligt. Zum Beispiel werden die Reiter, die oben auf einer Plone-Site erscheinen, wie folgt zusammengesetzt:

1. Zuerst erzeugen die Aktionsdefinitionen (engl. *action providers*), die ich in Kapitel 4 besprochen habe) eine Liste der anzuzeigenden Reiter. Sicherheits- und andere Überprüfungen werden dort vorgenommen.
2. Das Zope Page Templates-System setzt die Reiter zusammen und iteriert über die Werte, die von den Aktionsdefinitionen zurückgegeben werden, wobei HTML entsteht, das an den Browser geliefert wird.
3. Und schließlich erzeugt CSS die visuelle Darstellung des HTML im Browser auf dem Rechner des Benutzers.

Anstatt sich also zu fragen: »Wie kann ich die Reiter anpassen?«, müssen Sie sich genau überlegen, welche Anpassung Sie vornehmen möchten. Das könnte eine Änderung im CSS, im HTML, in den Daten oder den darunter liegenden Reitern bedeuten. Die allgemeinen Regeln sind folgende:

- Plone stellt Daten auf irgendeine Weise dar, entweder aus einem Datensatz oder aus Inhalten.
- CSS produziert die visuelle Darstellung.
- Das HTML und die Templates sind der Klebstoff zwischen den beiden.

Tatsächlich ist Plone auf so vielen Ebenen derart anpassbar, dass man leicht Kopfschmerzen bei dem Gedanken bekommt, welcher Teil angepasst werden muss. Um sicherzustellen, dass zukünftige Änderungen in Plone-Templates Ihr Anwendungsdesign nicht beeinflussen, sollten Sie nicht versuchen, die Templates anzupassen. Stattdessen empfehle ich, dass Sie es zuerst mit CSS oder den Aktionen versuchen. Auf diese Art reduziert sich die Wahrscheinlichkeit für Probleme, wenn sich die Templates in zukünftigen Versionen von Plone ändern.

7.2.3 Schriftart, Farben und Abstände anpassen

Das eigentliche Stylesheet, das die meiste Arbeit erledigt, `plone.css`, enthält eine Reihe von Variablen, die mit DTML (Document Template Markup Language) gesetzt werden. In diesem Buch behandle ich kein DTML, und wahrscheinlich ist das die einzige Stelle in Plone, wo es verwendet wird. Falls Sie DTML nicht schon kennen, rate ich Ihnen davon ab, es zu lernen, wenn es nicht sein muss! Das Zope Page Templates-System enthält alles, was Sie zum Erzeugen von Markup-Code (z.B. XML und [X]HTML) brauchen. DTML empfiehlt sich heutzutage lediglich, wenn Sie Text erzeugen wollen, der kein Markup-Code ist, wie zum Beispiel CSS. Es gibt sehr gute Online-Referenzen zu DTML für Zope, z.B. unter: *http://zope.org/Documentation/Books/ZopeBook/2_6Edition/DTML.stx*.

Die DTML-Syntax für dieses Stylesheet ist wirklich sehr einfach: Jede Variable bezieht sich auf ein entsprechendes Attribut in einer Eigenschaftsliste. Auf diese Eigenschaftsliste greifen Sie zu, indem Sie nacheinander auf PORTAL_SKINS, PLONE_STYLES und BASE_PROPERTIES klicken. In Abbildung 7.5 können Sie sehen, wie diese Datei im ZMI aussieht.

So findet z.B. `&dtml-fontColor;` die Variable `fontColor` und setzt sie ins Stylesheet, d.h., `fontColor` wird schwarz sein. Nun sehen Sie, wo diese Variable in der Datei `plone.css` referenziert wird. Um auf die CSS-Datei zuzugreifen, klicken Sie auf `portal_skins`, `portal_skins` und dann auf `plone.css`. In dieser Datei sehen Sie, dass `mainFontColor` an mehreren Stellen referenziert wird, z.B. wie folgt im Haupttext einer Seite:

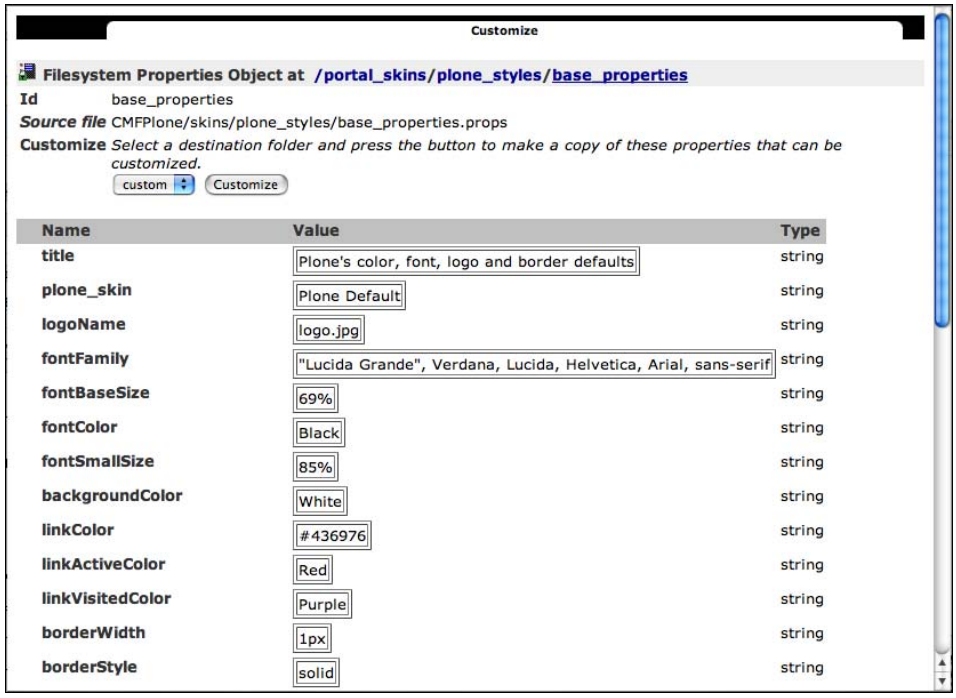


Abbildung 7.5: Die Grundeigenschaften des Stylesheets

```
body {
    font: &dtml-fontBaseSize; <dtml-var fontFamily>;
    background-color: &dtml-backgroundColor;;

    color: &dtml-fontColor;;

    margin: 0;
    padding: 0;
}
```

Sie können einfach im Stylesheet weiterlesen, wenn Sie wirklich wollen, aber die Variable zu ändern ist immer die schnellste Art herauszufinden, was davon betroffen ist.

Klicken Sie im ZMI auf PORTAL_SKINS, PLONE_STYLES, BASE_PROPERTIES und dann auf den CUSTOMIZE-Button. Wie Sie gesehen haben, wird dadurch ein Objekt im ZMI erzeugt, das Sie anpassen können. Dieses Mal ist das angepasste Objekt ein Ordner, der über die Eigenschaften verfügt, die in diesem Ordner vorhanden sind. Um auf die gerade angepassten Eigenschaften zuzugreifen, klicken Sie auf PORTAL_SKINS, CUSTOM und dann auf BASE_PROPERTIES. Als Nächstes wählen Sie den Properties-Reiter (siehe Abbildung 7.6).

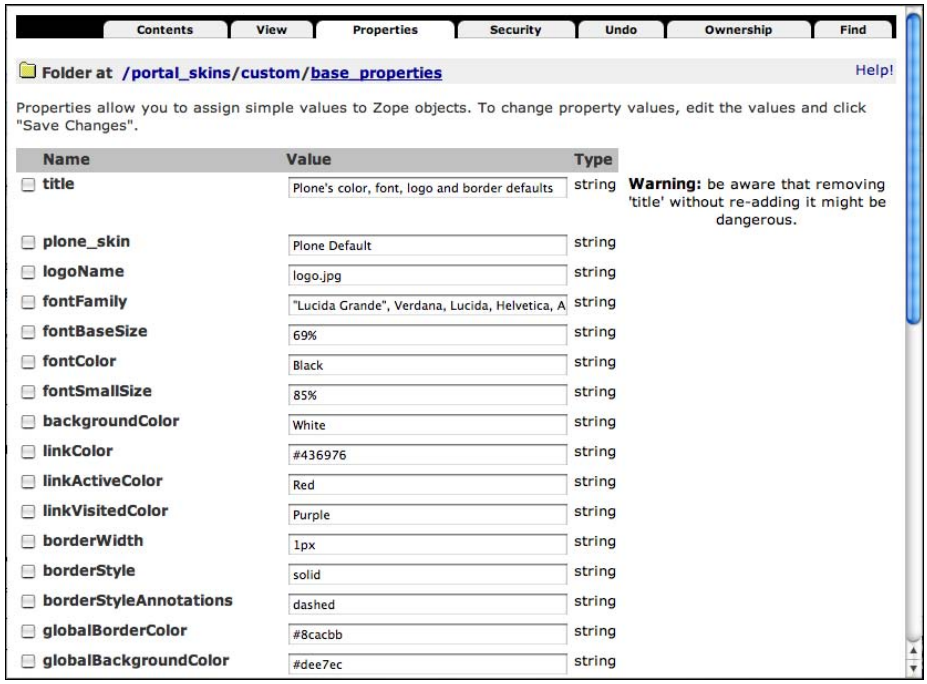


Abbildung 7.6: Eigenschaften des Ordners

In dieser Eigenschaftsliste können Sie die Eigenschaft `mainColor` ändern, z.B. auf `red` oder `#cc9900`. Ändern Sie den Wert dieser Eigenschaft, und klicken Sie dann auf SAVE CHANGES. Zurück in der Plone-Site sollten Sie nun die neue Farbe sehen.

In Kapitel 4 haben Sie ein Beispiel gesehen, in dem Benutzer Aktionen verändert haben, um einen Reiter oben auf der Seite zu ändern. Obwohl eine Aktion zwar mit einem Großbuchstaben anfangen kann, z.B. *Mitglieder*, wird sie dann auf der Webseite in Kleinbuchstaben angezeigt. Das liegt daran, dass CSS den Text wegen der Eigenschaft `textTransform` in der Eigenschaftsliste in Kleinbuchstaben umwandelt. Um diese Umwandlung auszusetzen, ändern Sie die Eigenschaft `textTransform` auf `none`.

Im Stylesheet werden Eigenschaften für alle Farben, Abstände und Schriftarten definiert, die in einer Plone-Site benutzt werden. Tabelle 7.1 beschreibt all diese Parameter.

Variablenname	Beschreibung
`LogoName`	Der Dateiname des Portallogos
`FontFamily`	Die Schriftfamilie für alle Texte, die keine Überschrift sind

Tabelle 7.1: CSS-Eigenschaften

Variablenname	Beschreibung
FontBaseSize	Die Schriftgrundgröße, aus der alles berechnet wird
FontColor	Die Schrifthauptfarbe
BackgroundColor	Die Hintergrundfarbe
LinkColor	Die Farbe normaler Links
LinkActiveColor	Die Farbe aktiver Links
LinkVisitedColor	Die Farbe besuchter Links
BorderWidth	Die Breite der meisten Rahmen in Plone
BorderStyle	Der Stil der Rahmenlinien (normalerweise solid)
borderStyleAnnotations	Der Stil der Rahmenlinien bei Kommentaren etc.
GlobalBorderColor	Die Rahmenfarbe der Hauptreiter, Portlets etc.
globalBackgroundColor	Die Hintergrundfarbe ausgewählter Reiter, Portlet-Überschriften etc.
GlobalFontColor	Die Schriftfarbe in Reitern und Portlet-Überschriften
HeadingFontFamily	Die Schriftfamilie der Überschriften h1, h2, ..., h6
headingFontBaseSize	Die Grundgröße bei der Berechnung der verschiedenen Überschriftengrößen
contentViewBorderColor	Die Rahmenfarbe der Reiter im INHALT-Reiter
contentViewBackgroundColor	Die Hintergrundfarbe im ANZEIGEN-Reiter des INHALT-Reiters
contentViewFontColor	Die Schriftfarbe in den Reitern des INHALT-Reiters
TextTransform	Gibt an, ob Text in Portlets, Reitern etc. in Kleinbuchstaben sein soll
evenRowBackgroundColor	Die Hintergrundfarbe gerader Zeilen in Listings
oddRowBackgroundColor	Die Hintergrundfarbe ungerader Zeilen in Listings
NotifyBorderColor	Die Rahmenfarbe von Meldungen wie Statusmeldungen und vom Kalender
notifyBackgroundColor	Die Hintergrundfarbe von Meldungen
helpBackgroundColor	Die Hintergrundfarbe des Kalender-Pop-up-Widgets
DiscreetColor	Die Farbe von Credits, Dokumentverfasserangaben, Formularhilfen
PortalMinWidth	Die minimale Breite des Portals
ColumnOneWidth	Die Breite der linken Spalte
ColumnTwoWidth	Die Breite der rechten Spalte

Tabelle 7.1: CSS-Eigenschaften (Forts.)

7.2.4 CSS anpassen

Wenn Sie kleine Anpassungen vornehmen, können Sie das in `ploneCustom.css` tun. Dies ist ein zweites Stylesheet, das nach `plone.css` geladen wird. Dank der Kaskadierungsfunktion von Stylesheets können Sie in `ploneCustom.css` beliebige Änderungen am gesamten Stylesheet vornehmen.

Um z.B. die Verfasserangaben zu ändern, die unten auf jeder Seite erscheinen, ändern Sie einfach `ploneCustom.css`. Greifen Sie noch einmal mit dem ZMI auf die Datei zu, und klicken Sie dann auf CUSTOMIZE. Dadurch wird eine Kopie dieses Stylesheets im Ordner `custom` angelegt. Ändern Sie die Verfasserangaben, indem Sie sie auf der Seite nach links verschieben und sie fett machen, wie in Abbildung 7.7 zu sehen ist.

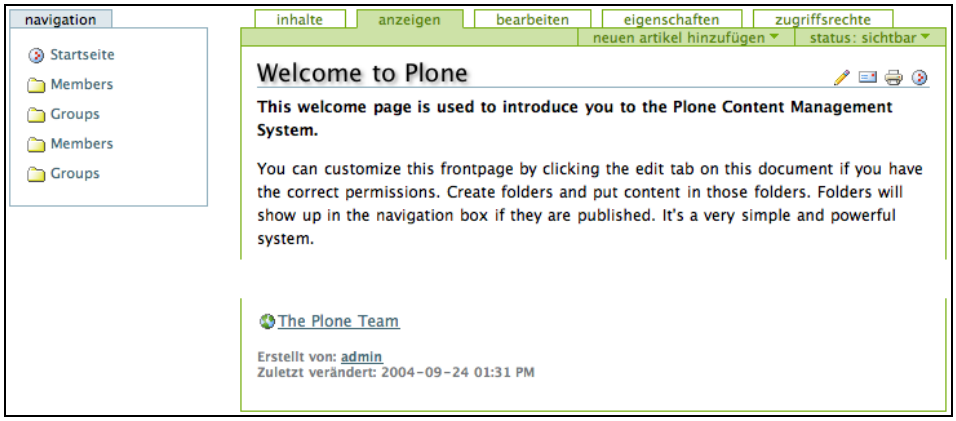


Abbildung 7.7: Neue fette Verfasserangaben links

Das tun Sie, indem Sie Folgendes hinzufügen:

```
div.documentByLine {
    text-align: left;
    font-weight: bold;
}
```

Hierbei haben Sie zwei Attribute für das `byline`-Element gesetzt: `text-align` und `font-weight`. Beachten Sie, dass Sie keine anderen Attribute im `byline`-Element ändern. Die anderen Attribute werden vom Original-Stylesheet geerbt. Mit ein paar einfachen Zeilen CSS haben Sie die Site geändert und haben sichergestellt, dass andere Änderungen an Plone keinen Einfluss auf Ihre Site haben. Für kleine Änderungen ist `ploneCustom.css` der am besten geeignete Ort.

Durch die Verwendung verschiedener Stylesheets können Sie Plone so benutzen, dass es für verschiedene Clients unterschiedlich aussieht. Websites verfügen oftmals über einen Button für eine Druckansicht, bei der die Seite in vereinfachter Form mit weniger Formatierungen angezeigt wird. Plone verringert dieses Problem dadurch, dass es ein separates Stylesheet anbietet. Wenn ein Browser die Seite druckt, wird sie von diesem Stylesheet formatiert. Alle weiteren Stylesheets sind oben auf der Seite enthalten. Sie finden sie, wenn Sie nacheinander auf PORTAL_SKINS, PLONE_TEMPLATES und HEADER.PT klicken.

Hinweis

Ein etwas ungewöhnlicheres Stylesheet hat den Namen *Präsentation*. Es wird nur von Opera unterstützt. Wenn dort der Browser in vollem Bildschirmmodus benutzt wird, werden aus Überschriften separate Seiten, die in einer präsentationsähnlichen Schnittstelle angezeigt werden.

7.2.5 Das Haupt-Template anpassen

Wie Sie im vorigen Kapitel gesehen haben, müssen Sie das `master`-Makro aus `main_template` benutzen, um das Look-and-Feel von Plone auf einer Seite zu erhalten. Jede Plone-Seite benutzt dieses Makro und füllt darin die passenden Slots. Wenn Sie sich dieses Haupt-Template im Detail anschauen, können Sie genau sehen, wie eine Plone-Seite in einem Page-Template gebaut wird und wie Sie diese individuellen Seitenelemente anpassen können.

Wenn Sie sich die Plone-Hauptseite anschauen, sehen Sie darauf eine Reihe von Elementen. Abbildung 7.8 zeigt eine Plone-Seite, auf der alle wesentlichen Elemente der Benutzerschnittstelle markiert sind. Tabelle 7.2 beschreibt den Zweck jedes dieser Elemente. Zu jedem Element in Abbildung 7.8 finden Sie einen entsprechenden Eintrag in der Tabelle.

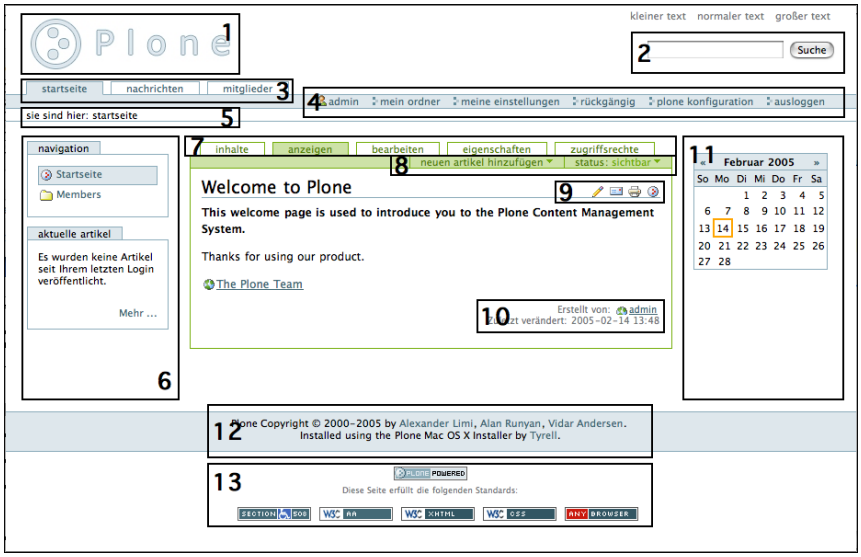


Abbildung 7.8: Alle Hauptelemente in der Plone-Benutzerschnittstelle

Nr.	Name	Beschreibung
1	Site-Logo	Enthält das obere Logo.
2	Suchformular	Enthält das Suchformular.
3	Portal-Reiter	Enthält die Reiter oben auf der Site.
4	Persönliche Leiste	Enthält persönliche Informationen zu dem Benutzer, z.B. *Login* und *Mein Ordner*.
5	Pfadnavigation	Zeigt den Ort des aktuellen Inhalts an.
6	Linker Slot	Hier werden Portlets angezeigt, die in der Eigenschaft `left_slot` vorkommen.
7	Inhalts-Reiter	Zeigt die Aktionen der Kategorie `content_tabs` für dieses Stück Inhalt.
8	Inhalts Drop-Down--Listen	Zeigt einige Dropdown-Menüs für diesen Inhalt, Workflow oder neue Inhaltstypen an.
9	Dokument-Aktionen	Zeigt die Aktionen für dieses spezielle Stück Inhalt an: Drucken oder als E-Mail versenden.
10	Herkunftsangaben	Zeigt eine Beschreibung des Inhalts und seines Autors.
11	Rechter Slot	Hier werden Portlets angezeigt, die in der Eigenschaft `right_slot` vorkommen.
12	Fußzeile	Zeigt Informationen am Seitenende an.
13	Kolophon	Zeigt weitere Angaben unter der Fußzeile an.

Tabelle 7.2: Elemente der Benutzerschnittstelle

Einen Abschnitt dieses Templates habe ich nicht behandelt: den Inhalt. Der ganze Text von *Welcome to Plone* bis hinunter zu *The Plone Team* ist Inhalt, der von Benutzern erstellt und bearbeitet wird. Dies ist der `main`-Slot im Page Template, der mit einem bestimmten Inhaltstyp oder Page Template gefüllt wird, wie Sie gesehen haben. In Kapitel 6 habe ich Slots behandelt und Ihnen gezeigt, wie Sie durch die Verwendung des `main`-Slots sicherstellen können, dass der Inhalt einer Plone-Seite erscheint.

Wie passen Sie also bei all diesen Komponenten einer Plone-Seite einen bestimmten Teil an? Die Antwort lautet, dass Sie den passenden Teil im `main_template` erst finden und dann sehen müssen, welchen Teil er aufruft, um diesen anschließend anzupassen. Aus diesem Grund werde ich das Haupt-Template im Detail behandeln.

Auf den ersten Blick scheint das Haupt-Template ziemlich lang und kompliziert zu sein, aber es besteht fast nur aus Makros, und seine Hauptaufgabe ist lediglich die, Inhalte aus anderen Bereichen zu holen. Das Haupt-Template finden Sie, indem Sie nacheinander auf PORTAL_SKINS, PLONE_TEMPLATES und MAIN_TEMPLATE klicken.

Die Philosophie hinter dem Haupt-Template ist, dass ein Benutzer die aktuelle Konfiguration des Templates nicht ändern muss, es sei denn, es sind größere Veränderungen geplant. Weil das Haupt-Template alle Inhalte aus anderen Stellen in Plone zusammenträgt, können Sie die zusammengestellte Seite ändern, indem Sie diese individuellen Elemente anpassen. Das heißt, Sie können genau die gewünschten Abschnitte modifizieren und müssen nicht das gesamte Template ändern.

Das Haupt-Template macht ausgiebigen Gebrauch von XML-Namespaces, um den `metal`-Code so einfach mit möglich zu halten. Beispiel:

```
<metal:headslot define-slot="head_slot" />
   <!-- A slot where you can insert elements in the header from a template -->
```

Hierbei entspricht der Tag-Name keinem XHTML-Standardelement, sondern er verwendet das `metal:`-Präfix, um den Namespace `metal:headslot` zu definieren. Das hat folgende Vorteile:

- Das Element `headslot` hat eine Semantik, da es das Element beschreibt. Man kann leicht feststellen, dass es der Slot für alles Mögliche ist, was Sie zum Kopf Ihrer Seite hinzufügen möchten.
- Attribute in diesem Element benutzen den Namespace im Element, falls nicht anders angegeben, d.h., statt `metal:fill-slot` können Sie einfach `fill-slot` benutzen.

- Das eigentliche Tag ist kein gültiges XHTML-Tag, d.h., es wird nicht dargestellt. Wenn die Darstellung des Tags jedoch gültiges XHTML generiert, wird dieser XHTML-Code dargestellt.

Wenn ein Makro verwendet wird, wird der Inhalt des aufrufenden Templates entfernt, d.h., man kann Kommentare in das aufrufende Template als Text ins Makro setzen. Beispiel:

```
<div metal:use-macro="here/global_searchbox/macros/quick_search">
    The quicksearch box, normally placed at the top right
</div>
```

Wegen des Kommentars kann man leicht feststellen, dass sich dieses Makro auf den Kasten mit der Suchabfrage in der oberen rechten Ecke der Seite (Element 2 in Abbildung 7.8) bezieht. Um das Makro zu sehen, müssen Sie das Script namens `global_searchbox` mit dem Makro `quick_search` darin finden. Das Haupt-Template geht durch die `main`-Makros und zieht dabei Informationen aus verschiedenen Templates und Scripts heraus.

Nach diesem Abschnitt erreicht das Haupt-Template den Hauptinhalt der Seite, d.h. das darzustellende Objekt. In Kapitel 6 habe ich den Unterschied zwischen einem Slot und einem Makro erklärt. Erinnern Sie sich daran, dass ein Template Slots definiert, die dann mit Inhalt gefüllt werden. Für diesen Inhalt gibt es wirklich nur einen wichtigen Slot, den ich schon oft genannt habe: den `main`-Slot.

Ein häufiges, vielleicht verwirrendes Muster in Plone kommt bei der Definition eines Slots innerhalb eines `fill`-Slots vor. Folgendes ist z.B. die Definition des Slots `css_slot`:

```
<metal:cssslot fill-slot="css_slot">
    <!-- A slot where you can insert CSS from a template -->
    <metal:cssslot define-slot="css_slot" />
</metal:cssslot>
```

Dieses Muster sieht etwas merkwürdig aus, aber es definiert den Slot und erzeugt dann den fill-Slot erneut. Wenn Sie sich das Haupt-Template genau anschauen, befinden sich diese Slots tatsächlich innerhalb von `use-macro` im Kopf, d.h., das Kopf-Makro darf diesen Slot füllen. Aber Sie möchten auch, dass das End-Template den Slot füllen kann, deswegen wird der Slot neu definiert. Somit kann ein Slot nun an zwei Orten gefüllt werden, was ein nützliches Vorgehen beim Ändern des Templates ist.

Bei der Durchsicht des restlichen Haupt-Templates werden Sie die linke und rechte Spalte, die Fußzeilen und das Kolophon erreichen. Beachten Sie, dass die linke Spalte möglicherweise vor dem Hauptinhalt einer Seite erscheint (jedenfalls, wenn Ihre Sprache von links nach rechts gelesen wird), aber das Stylesheet

bewegt sie dorthin. Das garantiert, dass beim Besuchen der Site mit einem rein textbasierten Browser der Hauptinhalt zuerst erscheint und nicht erst nach den ganzen Navigationsoptionen.

Tabelle 7.3 beschreibt die Makros und Slots im Haupt-Template.

Name	Beschreibung	Slot oder Makro?
Cache headers	Setzt den Kopf des HTTP-Caches (Hypertext Transfer Protocol) für den Inhalt.	Makro: `cacheheaders` in `global_cache_settings`
Head slot	Hiermit können Inhalte etwas zum `head`-Element einer Seite hinzufügen.	Slot: `head_slot`
CSS slot	Hiermit können Inhalte einen eigenen CSS-Code zur Seite hinzufügen.	Slot: `css_slot`
JavaScript head slot	Hiermit können Inhalte einen eigenen JavaScript-Code zur Seite hinzufügen.	Slot: `javascript_head_slot`
Site actions	Die Site-Aktionen erlauben Ihnen eine Reihe von Aktionen oberhalb des SUCHEN- Kastens. Standardmäßig können Sie damit die Schriftgröße verändern.	Makro: `site_actions` in `global_siteactions`
Quick search	Der Kasten für die schnelle Suche in der oberen rechten Ecke.	Makro: `quick_search` in `global_searchbox`
Portal tabs	Die (normalerweise blauen) Portal-Tabs, die normalerweise oben links erscheinen. Welche wirklich angezeigt werden, wird von den Aktionen bestimmt. Sie bestimmen, wie die Tabs in HTML dargestellt werden.	Makro: `portal_tabs` in `global_sections`
Personal bar	Die persönliche Leiste oben rechts: einloggen, ausloggen usw.	Makro: `personal_bar` in `global_personalbar`
Path bar	Die Pfad-Navigationsleiste, die mit »Sie sind hier:« beginnt.	Makro: `path_bar` in `global_pathbar`
Content views	Die (normalerweise grünen) Tabs oberhalb des Inhalts. Sie werden nur angezeigt, wenn der aktuelle Benutzer den Inhalt bearbeiten darf. Welche Tabs wirklich angezeigt werden, wird von den Aktionen bestimmt. Sie bestimmen, wie die Tabs in in HTML dargestellt werden.	Makro: `content_views` in `global_contentviews`
Content actions	Die kleinen Dropdown-Aktionen in der oberen rechten Ecke der Kontextleiste.	Makro: `content_actions` in `global_contentviews`
Portal status message	Ein Hinweis, der immer dann erscheint, wenn sich etwas verändert.	Makro: `portal_message` in `global_statusmessage`

Tabelle 7.3: Makros und Slots im Haupt-Template

Name	Beschreibung	Slot oder Makro?
Header	Der Kopfteil eines Inhalts.	Slot: `header`
Main	Der Hauptteil eines Inhalts.	Slot: `main`
Sub	Der untere Teil eines Inhalts, in dem die Kommentare zum Objekt erscheinen.	Slot: `sub`
Linke Portlets	Die Slots bzw. Portlets, die im linken Teil der Seite erscheinen. Hier gibt es einige Definitionen: `column-one-slot` ist die ganze linke Spalte, und `portlets-one-slot` ist dann der Slot. Wenn keiner dieser Slots definiert ist, wird das Makro aufgerufen.	Makro: `left_column` in `portlets_fetcher`
Rechte Portlets	Die Slots bzw. Portlets, die im rechten Teil der Seite erscheinen (siehe linke Portlets).	Makro: `right_column` in `portlets_fetcher`
Fußzeile	Copyright- und andere Hinweise.	Makro: `portal_footer` in `footer`
Kolophon	Verschiedene Hinweise am unteren Ende.	Makro: `colophon` in `colophon`

Tabelle 7.3: Makros und Slots im Haupt-Template (Forts.)

Mit dieser Information ausgestattet, ist es nun eine Frage der Anpassung eines Makros oder Slots, wenn das Look-and-Feel der Seite geändert werden soll. Um es noch einmal zu sagen: Sie sollten nicht das Haupt-Template selbst anpassen, sondern die Teile, die es aufruft. Der nächste Abschnitt zeigt ein paar Beispiele für Anpassungen, die Sie in Plone vornehmen können.

7.2.6 Beispiele für Anpassungs-Code-Schnipsel untersuchen

Die folgenden Abschnitte enthalten einige Beispiele, die einfache Anpassungen an einer Plone-Site demonstrieren. Bei manchen Lösungen werden ein oder zwei Alternativen zur Lösung der Aufgabe gezeigt.

Einen Block entfernen

Ein ziemlich netter Trick ist es, wenn man einen Block wie die Pfadleiste oder den Suchkasten aus der Benutzerschnittstelle leicht entfernen kann. Das können Sie auf zwei Arten erreichen. Der offensichtlichste Ansatz besteht in der Anpassung des Makros, mit dem das Element angezeigt wird. Um z.B. die Pfadleiste zu entfernen, könnten Sie nacheinander auf PORTAL_SKINS, PLONE_TEMPLATES und GLOBAL_PATHBAR klicken und dann das Element auf der Ebene des Page Templates entfernen. Sie können z.B. Folgendes ändern:

```
<div metal:define-macro="path_bar"
    id="pathBar"
    tal:define="breadcrumbs python:here.breadcrumbs(here);
        portal_url portal_url|here/portal_url">
```

indem Sie wie folgt eine Code-Zeile hinzufügen:

```
<div metal:define-macro="path_bar"
    id="portal-breadcrumbs"
    tal:condition="nothing"
    tal:define="breadcrumbs python:here.breadcrumbs(here);
        portal_url portal_url|here/portal_url">
```

Das bedeutet, es wird ein Page Template angepasst, was überhaupt kein Problem ist, weil Sie mittlerweile schon damit vertraut sein sollten. Ein etwas anderer Ansatz besteht darin, Elemente auf CSS-Ebene zu verstecken. Das bedeutet, ein Element wird weiterhin dargestellt, und es wird HTML dafür generiert, wird dann aber für den Client ausgeschaltet, d.h., es wird unsichtbar. Da das HTML weiterhin generiert wird, ist diese Lösung suboptimal, aber es ist ein netter Trick.

Die meisten Elemente in Plone haben eine eindeutige DOM-Element-ID (Document Object Model). Im Fall der Pfadleiste lautet sie `portal-breadcrumbs`, wie Sie im Code oben sehen können. Um `portal-breadcrumbs` nicht mehr anzuzeigen, fügen Sie einfach Folgendes zu `ploneCustom.css` hinzu:

```
#portal-breadcrumbs {
    display: none;
}
```

Portal-Reiter ändern

Ich habe Ihnen bereits gezeigt, wie Sie den Text der Portal-Reiter ändern können, indem Sie die Aktionen ändern. Sie werden mit Hilfe des Stylesheets angezeigt, nicht etwa mit Tabellen, auch wenn man als Benutzer das vielleicht erst einmal denkt. In Tabelle 7.3 sehen Sie, dass der Code für die Portal-Reiter in `portalTabs` steht. Um den Rahmen der nicht ausgewählten Reiter gepunktet darzustellen, können Sie einfach das Stylesheet `ploneCustom` wie folgt ändern:

```
#portal-globalnav li a {
    border: 1px dotted;
}
```

Die Reiter bestehen aus einer Reihe von HTML-Listen- (`li`) und Anker-Elementen (`a`), d.h., durch eine Änderung des CSS für diese Elemente ändern Sie das Erscheinungsbild dieser Reiter. Im später folgenden Abschnitt »Fallstudie: Die NASA-Skin« werde ich zeigen, wie man diese Reiter zu Bildern abändert.

Dank CSS können Sie mit dem Attribut `position` alle Elemente an einen anderen Platz verschieben. Verschieben Sie Ihre Reiter als Nächstes nach ganz oben auf den Bildschirm, oberhalb des Logos und des SUCHEN-Kastens. Zu diesem Zweck verwenden Sie den Wert `absolute` bei der Position, wo Sie auch die Werte `left`, `right`, `top` und `bottom` verwenden dürfen. Fügen Sie Folgendes zum Stylesheet `ploneCustom` hinzu, um die Portal-Reiter oben auf Ihre Plone-Site zu platzieren:

```
#portal-globalnav {
  position: absolute;
  top: 0em;
}
```

Das ist ein mächtiges Verfahren zum Verschieben von Elementen. Beim Positionieren der Elemente haben Sie mehrere Möglichkeiten, inklusive einer relativen Positionierung, aber das erfordert etwas Arbeit mit CSS, um die Positionierung genau hinzubekommen.

7.2.7 Linke und rechte Slots verschieben

Die linken und rechten Slots habe ich in Kapitel 4 beschrieben. Dort habe ich auch gezeigt, wie Sie einen neuen Slot zur Slot-Liste hinzufügen. Vielleicht haben Sie bemerkt, dass die Bezeichnungen *linke* und *rechte* Slots ein wenig missverständlich sein können. Standardmäßig werden die Slots in diesen Positionen angezeigt, aber sie lassen sich leicht verschieben.

Hinweis

Das funktioniert nur dann, wenn Sie die Plone-Tableless-Skin verwenden. Da das aber nicht die Standardeinstellung ist, müssen Sie die Skin mit dem Werkzeug `portal_skins` so ändern, wie es oben im Abschnitt »Skins mit dem Werkzeug portal_skins verwalten« beschrieben wurde.

Wenn Sie z.B. die linken Portlets rechts auf der Seite anzeigen möchten, könnten Sie das mit der folgenden Änderung in `ploneCustom.css` bewirken:

```
#portal-column-one  {
    float: right;
}
#portal-column-content {
    float: left;
}
```

Dadurch wird die Spalte ganz links nach rechts verschoben, und der Hauptteil wandert nach links.

Hilfe in Formularen verstecken

Wenn Sie in allen Formularen die Hilfe ausblenden wollten, wäre es unrealistisch, dafür alle Templates zu ändern. Aber eine ähnliche Taktik könnten Sie einsetzen, um die Pfadleiste zu verstecken, indem Sie bei den Forularelementen einfach nur `display: none` setzen. Folgendes hat den gewünschten Effekt, nämlich das Eingabeelement nicht in eine neue Zeile zu setzen:

```
div.formHelp {
    display: none;
}
```

Abbildung 7.9 zeigt die Feedback-Seite ohne Pfadnavigation, mit versteckter Hilfe, gepunkteten Reitern und mit einem nach rechts verschobenen linken Slot, was alles mit nur ein paar Zeilen CSS geändert wurde.

Abbildung 7.9: Kombinationseffekt einiger Beispiele

7.2.8 Wie finden Sie Element X?

Wie ich gezeigt habe, sind die Templates, Scripts und Bilder, die eine Plone-Skin ausmachen, im Verzeichnis `skins` einer Plone-Installation enthalten. In diesem Verzeichnis befinden sich viele Dateien, d.h., es wäre langwierig und unproduktiv, bei einer Dateiänderung durch die ganze Liste zu gehen. Stattdessen ist es sehr hilfreich, einige Grundtechniken zum Finden jener Elemente zu beherrschen, die Sie ändern möchten.

Überlegen Sie sich, auf welcher Ebene Sie das Element anpassen möchten. Wie schon erwähnt, stehen Ihnen bei der Darstellung eines Objekts drei Ebenen zur

Verfügung. Wenn Sie die visuelle Darstellung oder Platzierung des Elements ändern möchten, können Sie die Änderungen wahrscheinlich allein mit CSS vornehmen und müssen sonst nichts tun.

Falls CSS nicht ausreicht, finden Sie den nächstbesseren Ort bei der Suche in den Templates. Nehmen Sie beispielsweise an, sie möchten den Text ändern, der auf der Seite erscheint, wenn sich ein Benutzer anmeldet, oder Sie möchten gleich die ganze Seite ändern. In diesem Beispiel werden Sie die Seite in Abbildung 7.10 ändern, damit daraus ein Script wird, das etwas Ungewöhnliches macht.

Abbildung 7.10: Die Seite »Sie sind nun eingeloggt«

Es gibt einige Hinweise darauf, wie man dieses Template findet, das Sie ändern können. Im Folgenden gehe ich sie alle durch.

Suchen mit Hilfe der URL

Der URL (Uniform Resource Locator) auf eine Seite wird in eine Folge von Objekten in Plone übersetzt, die traversiert werden. In Abbildung 7.11 habe ich eine Traversierung zur Seite `login_success` durchgeführt. In diesem Fall ist der letzte Teil des URL `login_success`, wie Sie in der Adresszeile in Abbildung 7.11 sehen können. Wenn ein Objekt in ein FSDV geladen wird, wird die Dateierweiterung weggelassen, d.h., Sie suchen nach einem Template oder Script, das mit `login_success` anfängt.

In Zope können Sie diese Suche durchführen, indem Sie zum `portal_skins`-Werkzeug gehen und auf den FIND-Reiter klicken. Dort geben Sie **login_success** im Feld `with ids` ein. Lassen Sie alle anderen Einstellungen unverändert, und klicken Sie auf den FIND-Button. Dann finden Sie ganz bestimmt das Template `login_success`.

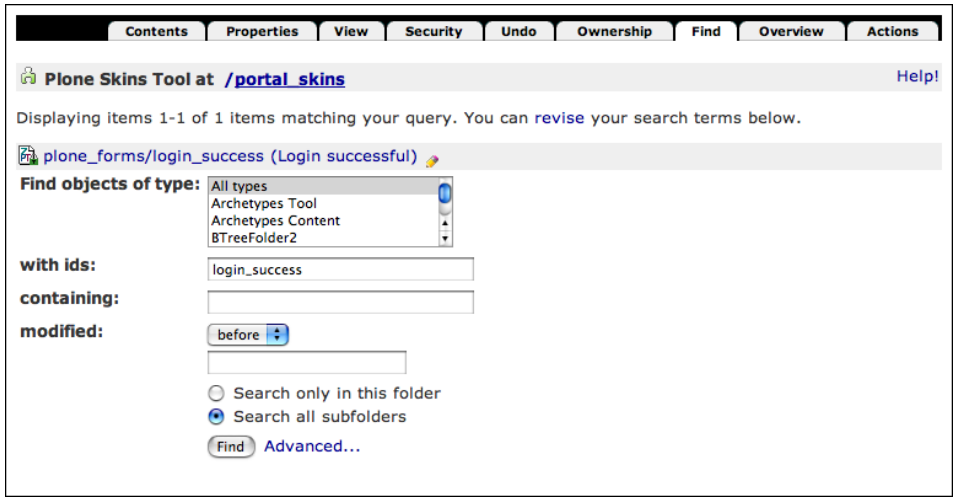


Abbildung 7.11: Suche nach einer ID

Diese Suche können Sie auch im Dateisystem durchführen, je nach Betriebssystem und verfügbaren Werkzeugen. Die schnellste Methode, um diese Datei unter Linux zu finden, ist die, in Ihr `CMFPlone`-Verzeichnis zu gehen und Folgendes zu machen:

```
$ cd skins
$ find -name 'login_success*' -print
./plone_forms/login_success.pt
```

Unter Windows öffnen Sie den Ordner `CMFPlone` im Windows Explorer und klicken auf den SUCHEN-Reiter. Geben Sie dann den Namen der Datei als **login_success** ein, und klicken Sie auf SUCHEN. Danach sollten Sie eine Liste der passenden Dateien erhalten.

Als Ergebnis dieser Suche sollte `CMFPlone/plone_forms/login_success.pt` herauskommen. Wenn Sie die gleiche Suche im ZMI durchführen, klicken Sie nacheinander auf PORTAL_SKINS, PLONE_FORMS und LOGIN_SUCCESS.

Suche nach einem Textteil

Ein etwas grober Ansatz, der meist auch zum Erfolg führt, besteht darin, eine Volltextsuche im Code durchzuführen, um das Element zu finden, das die Seite darstellt. Wenn Sie z.B. die Seite in Abbildung 7.10 betrachten, sehen Sie, dass darauf der Text *Sie werden feststellen, dass* vorkommt. Die einfachste Methode, den Teil zu finden, der diesen Text anzeigt, ist, danach zu suchen.

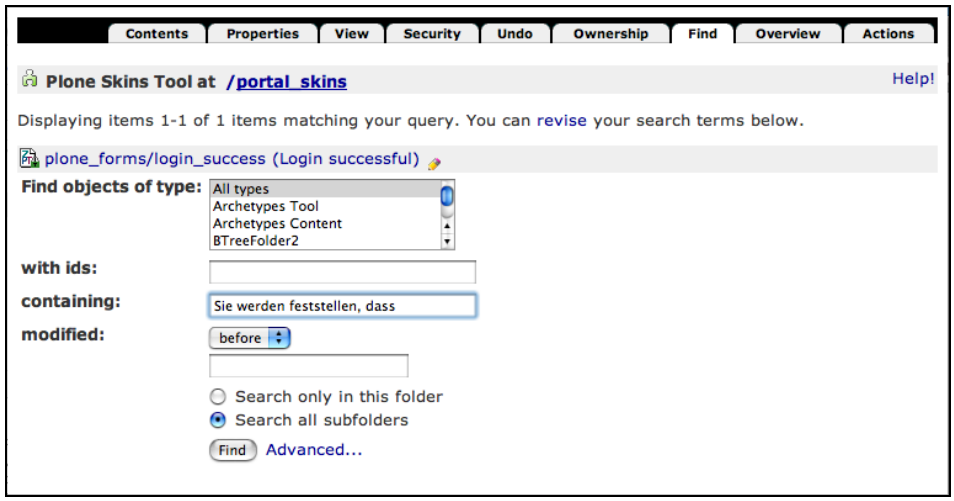


Abbildung 7.12: Suche nach Text

In Zope können Sie diese Suche auch durchführen, indem Sie zum `portal_skins`-Werkzeug gehen und auf den FIND-Reiter klicken. Dort geben Sie **Sie werden feststellen, dass** im Feld `containing` ein. Alle anderen Einstellungen lassen Sie unverändert und klicken dann auf den FIND-Button. Auch dann sollten Sie ganz bestimmt das Template `login_success` finden.

Diese Suche können Sie auch im Dateisystem durchführen, je nach Betriebssystem und verfügbaren Werkzeugen. Die schnellste Methode, diese Datei unter Linux zu finden, ist die, in Ihr `CMFPlone`-Verzeichnis zu gehen und Folgendes einzugeben:

```
$ grep -ri "Sie werden feststellen, dass" *
plone_forms/login_success.pt: Notice that the top
```

Unter Windows öffnen Sie den Ordner `CMFPlone` im Windows Explorer und klicken auf den SUCHEN-Reiter. Geben Sie dann als Inhalt der Datei **Sie werden feststellen, dass** ein, und klicken Sie auf SUCHEN. Danach sollten Sie eine Liste der passenden Dateien erhalten. Mit dieser etwas groben Technik haben Sie das Template `login_success` gefunden, das einem Benutzer diese Meldung anzeigt.

Diese Methode hat folgende Tücken:

- Passen Sie auf die Schreibweise von Inhalten in CSS auf! Suchen Sie immer unabhängig von der Schreibweise (ist in Windows standardmäßig eingestellt). Es ist ärgerlich, nach `home` zu suchen, wenn es im Template tatsächlich `Home` heißt und nur von CSS mit Kleinbuchstaben angezeigt wird.

- Wenn Sie das in einer anderen Sprache als Englisch versuchen, wurde der Inhalt eventuell lokalisiert, wodurch die Suche fehlschlägt.
- Gelegentlich gibt es keinen suchbaren Text, den man finden kann. In diesem Fall ist eine Suche über die URL ratsam.

7.3 Neue Skins und Ebenen erstellen

Bisher habe ich die Anpassung vorhandener Skins behandelt. Die Erstellung einer völlig neuen Skin oder einer neuen Ebene unterscheidet sich davon nur wenig. Ich werde einen springenden Punkt behandeln: die Platzierung Ihrer Templates und Scripts im Dateisystem.

Templates und Scripts im Dateisystem zu erstellen und neue Skins und Ebenen zu erstellen ist definitiv die beste Art, um eine langfristige Wartbarkeit und Flexibilität zu erreichen. Die Erstellung von Skin-Elementen ist nicht nur wesentlich leichter mit den gewohnten Werkzeugen im Dateisystem, sondern erlaubt Ihnen auch, Ihren Code sehr leicht auszuliefern. Das Schreiben im Dateisystem ist für fast alle Plone-Entwickler die Arbeitsweise der Wahl, bei Bedarf mit kleinen Änderungen im Verzeichnis `custom`.

7.3.1 Neue Skins erstellen

Wie Sie gesehen haben, ist eine Skin nicht mehr als eine Ansammlung von Ebenen. Für meine neue Skin wollte ich all meinen eigenen Code an einer Stelle konzentrieren. Dazu habe ich das `portal_skins`-Werkzeug gewählt und habe einen neuen Ordner mit der ID `custom_chrome` hinzugefügt.

Um dann eine neue Skin zu erstellen, müssen Sie PORTAL_SKINS anklicken, den PROPERTIES-Reiter wählen und unter dem Text ADD A NEW SKIN eine neue Skin hinzufügen. Sie müssen eine Reihe von Ebenen eingeben, die Sie für diese Skin einrichten möchten. In diesem Beispiel habe ich eine neue Skin namens *Custom Chrome* sowie eine Reihe von Ebenen hinzugefügt, wie in Abbildung 7.13 zu sehen ist.

Dann habe ich die Ebenen für die Skin hinzugefügt. In diesem Fall hatte die Skin keine Ebene namens `custom` darin. Stattdessen hatte sie einen Ordner namens `custom_chrome`. Nun haben Sie zwei Skins, die zwei Ebenen und zwei Ordner verwenden. Alle zum `custom_chrome`-Ordner hinzugefügten Objekte beeinflussen diese Skin, aber nicht die Plone-Default-Skin.

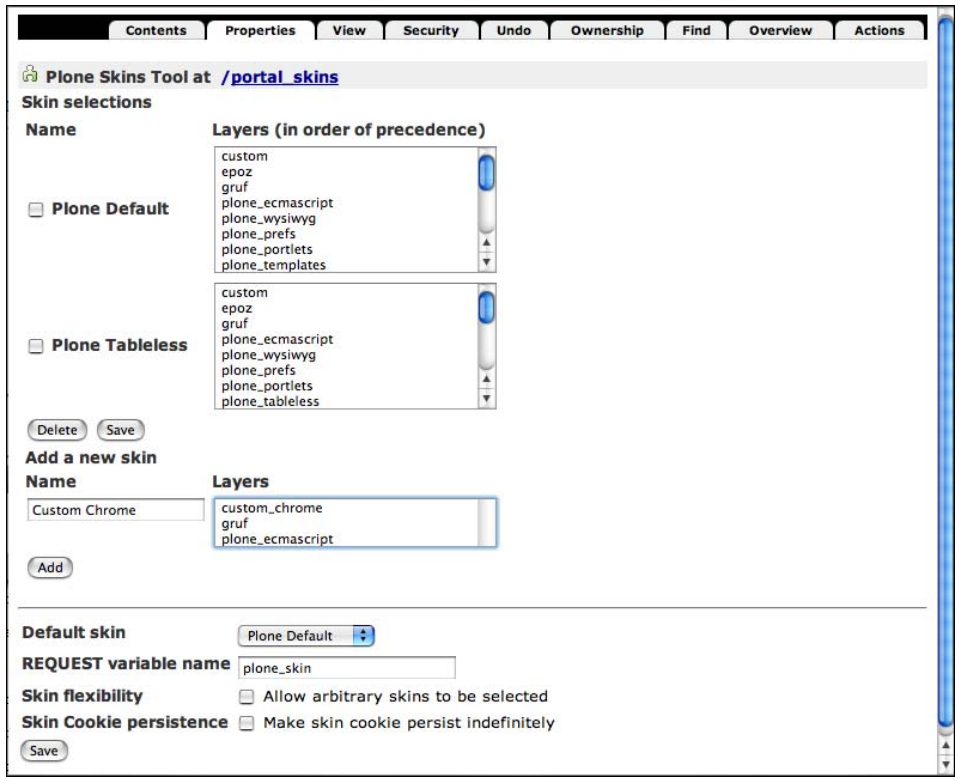


Abbildung 7.13: Hinzufügen der Skin Custom Chrome

7.3.2 Mehrere Skins benutzen

Wie schon erwähnt wurde, enthält eine Standard-Plone-Site zwei Skins: *Plone Default* und *Plone Tableless*. Im vorigen Abschnitt habe ich eine neue Skin hinzugefügt, *Custom Chrome*. Wie ich in Kapitel 4 beschrieben habe, können Sie die Standard-Skin mit der Plone-Schnittstelle setzen. Klicken Sie auf PLONE KONFIGURATION und dann auf den AUSSEHEN-Button. Dieser bildet die Möglichkeiten ab, die im ZMI verfügbar sind, wenn Sie dort PORTAL_SKINS anklicken, den PROPERTIES-Reiter wählen und auf der Seite nach unten scrollen.

Aber Sie haben noch eine weitere Möglichkeit: eine Variable namens `REQUEST`. Dies ist die Anfragevariable, die die Informationen über die Skin des Benutzers enthält. Standardmäßig ist das `plone_skin`, was der Name des Cookies ist. Aber sie kann auch über andere Anfragevariablen wie den Abfrage-String übergeben werden. Diese Variable ist nur über das ZMI verfügbar.

Sie können Skins auch mit Hilfe eines Programms setzen. Damit können Entwickler unterschiedliche Benutzer abhängig von irgendeiner Geschäfts- oder Site-Logik mit jeweils einer eigenen Skin bedienen. Wenn ein Benutzer z.B.

Inhalte für eine Site schreibt, sieht er die Plone-Standard-Skin. Anonyme Benutzer hingegen sehen eine völlig andere Skin. Nicht der Benutzer trifft dann die Entscheidung, sondern die Site. Wenn Sie wirklich wollen, können Sie die Skin abhängig vom Ordner machen, auf den zugegriffen wird, allerdings kann dieser Ansatz verwirrend sein, daher möchte ich ihn nicht empfehlen.

Um die Skin zu ändern, fügen Sie ein Script(Python)-Objekt namens `setSkin` zur Wurzel Ihrer Plone-Site hinzu. Dann fügen Sie den folgenden Code hinzu:

```
##title=Skin changing script
##parameters=
req = context.REQUEST
if req['SERVER_URL'].find('internal.somesite.org') > -1:
    context.changeSkin("Plone Default")
context.changeSkin("Custom Chrome")
```

Die eigentliche Logik für die Auswahl der Skin wird von der Geschäftslogik der Site abhängig sein. In diesem Fall erhalten alle, die auf *http://internal.somesite.org* zugreifen, die Plone-Default-Skin, während alle, die auf *http://external.somesite.org* zugreifen, die Custom Chrome-Skin erhalten. Leider ist ein Haken dabei, und zwar der, dass Sie die Skin nicht in Abhängigkeit von der Sicherheitsstufe des Benutzers wählen können (z.B. wenn authentifizierte Benutzer die eine Skin und Manager eine andere Skin sehen sollen). Dieser offensichtliche Bedarf kann momentan nicht erfüllt werden, ohne die Plone-Site ganz heftig zu hacken.

Hinweis

Man kann häufig beobachten, dass eine Skin von ungeprüften Client-Informationen abhängig gemacht wird, was aber nicht wirklich sicher ist, weil Sie der Information vom Client vertrauen. Wie Sie diese wirklich sicher machen, hängt von Ihren speziellen Netzwerkeinstellungen ab. In den meisten Fällen können Sie das einfach mit einer Firewall oder einem Proxy-Server wie Apache erledigen, der so konfiguriert werden könnte, dass alle externen Anfragen auf *http://intern.einesite.org* blockiert werden. Die Integration in Apache beschreibe ich in Kapitel 10.

Um diesen Code zu aktivieren, weisen Sie diesem Objekt eine Zugriffsregel zu. Das bedeutet, dass bei jedem Zugriff auf die Plone-Site das Script(Python)-Objekt ausgeführt wird. Jedes Mal, wenn das Script läuft, wird die Skin abhängig vom Script gesetzt. Um einem Script eine Regel zuzuweisen, wählen Sie *Set Access Rule* im Dropdown-Menü und geben dann den Namen Ihres Script (Python)-Objekts ein. Machen Sie nun einen Test, indem Sie Ihre Site besuchen und nachsehen, welche Skin Sie erhalten.

Mit Zugriffsregeln müssen Sie vorsichtig sein, weil sie bei jedem Zugriff auf diesen Ordner (oder die Plone-Site) aufgerufen werden. Sie müssen sicherstellen,

dass sie korrekt sind und dass nichts Falsches darin passiert. Wenn Sie versehentlich ein fehlerhaftes oder unkorrektes Script(Python)-Objekt geschrieben haben und nicht einmal mehr im ZMI einen Zugriff darauf bekommen, um es zu reparieren, dann können Sie die Zugriffsregeln ausschalten, indem Sie Plone mit der folgenden Umgebungsvariablen neu starten:

```
SUPPRESS_ACCESSRULE = 1
```

Anhang A erklärt, wie Umgebungsvariablen gesetzt werden, falls Sie mit ihnen nicht vertraut sein sollten.

7.3.3 Eine neue Skin im Dateisystem erstellen

Während dieser Kapitel habe ich durchgehend das ZMI benutzt. Aber die meisten Plone-Entwickler arbeiten tatsächlich mit dem Dateisystem. Eine Skin kann man im Dateisystem in der Tat sehr leicht erstellen.

Gehen Sie ins Home-Verzeichnis der Instanz Ihrer Plone-Installation. Erstellen Sie ein neues Verzeichnis im `Products`-Verzeichnis. Dessen Name ist der Name des Produkts, und laut Konvention ist er relativ kurz, ohne Leerzeichen oder Unterstriche und mit Groß- und Kleinschreibung. Beispiele hierfür sind `PloneBookExample`, `CMFPlone` und `PloneSilverCity`. Erstellen Sie in diesem Ordner eine neue Datei namens `__init__.py` und ein Verzeichnis namens `skins`. In der Datei `__init__.py` müssen Sie die beiden folgenden Zeilen hinzufügen:

```
from Products.CMFCore import DirectoryView
DirectoryView.registerDirectory('skins', globals())
```

Als Nächstes starten Sie Plone erneut und klicken auf PORTAL_SKINS, um ein FSDV hinzuzufügen. Dann wird eine Liste der registrierten Verzeichnisse geöffnet. Scrollen Sie nach unten, bis Sie dasjenige finden, das dem gerade registrierten Verzeichinis entspricht. Das wird der Name des Verzeichnisses sein, mit `/skins` am Ende. Geben Sie eine ID ein, die Sinn macht, und klicken Sie auf ADD. Nun haben Sie ein leeres Verzeichnis, in dem Sie die Ebenen Ihrer Skin erstellen können.

Fehlersuche in Skins

Ein weiterer Grund, aus dem ich mit Ihnen immer wieder das ZMI statt des Dateisystems benutzt habe, ist der, dass es Feedback bei Fehlern gibt und Sie damit vertraut macht, Objekte in andere Objekte zu platzieren. Noch eine positive Eigenschaft des ZMI ist, dass Änderungen sofort wirksam sind. Wenn Sie ein Objekt ändern und die Ansicht aktualisieren, sehen Sie sofort die Änderungen (vorausgesetzt, Sie haben keinen Cache).

Beim Dateisystem ist das nicht so. Wenn Sie etwas im Dateisystem ändern, wird es in Plone nicht aktualisiert. Der Grund dafür liegt in der Performance. Plone

kann nicht wissen, dass Sie diese Änderung vorgenommen haben, d.h., es muss die Kopie dieses Objekts im Zope-Cache aktualisieren. Ohne sich auf Tricksereien mit Benachrichtigungen in Dateisystemen einzulassen ist eine Plone-Site in einem von zwei Zuständen: im Produktions- oder im Debug-Modus. Im Debug-Modus prüft Plone alle Verzeichnisse, findet veränderte Dateien und aktualisiert sich selbst. Das heißt, Sie können etwas ändern, und es wird sofort erscheinen. Im Produktionsmodus werden Ihre Änderungen jedoch erst dann wirksam, wenn Sie die Skin aktualisieren (siehe Kapitel 11) oder Zope neu starten.

Aus offensichtlichen Gründen ist bei der Entwicklung von Skins in Plone der Betrieb im Debug-Modus die richtige Wahl. Kapitel 2 zeigte, wie man die Konfiguration von Plone so ändern kann, dass es im Debug-Modus läuft. Hier eine kurze Wiederholung: Öffnen Sie die Datei `zope.conf` im Verzeichnis `etc` Ihrer Installation, und stellen Sie sicher, dass die Direktive `debug-mode` auf `on` gesetzt ist.

Dateisystemobjekte verwenden

FSDVs können nur jene Zope-Objekte abbilden, die speziell zu diesem Zweck konfiguriert wurden. Das Zope-Objekt wird auf Grund der Erweiterung des Dateinamens bestimmt. Der Inhalt dieser Datei ist der Inhalt eines Attributs des Objekts, normalerweise der Hauptinhalt, z.B. der binäre Inhalt eines Bildes oder der Textinhalt des Templates.

Um ein Objekt in Ihrem leeren FSDV zu erstellen, gehen Sie einfach ins Verzeichnis `skins` und beginnen damit, Dateien hinzuzufügen, die den Objekten entsprechen, die Sie erstellen möchten. Wenn die Datei einmal in Zope als Zope-Objekt geladen ist, wird diese Erweiterung abgeschnitten. Beispielsweise wird aus `ein_template.pt` ein Dateisystem-Page Template mit der ID `ein_template`. Tabelle 7.4 beschreibt diese Erweiterungen.

Ereiterungen	Objekt-Typ	Äquivalentes Zope-Objekt
`.pt, .zpt, .html, .htm`	Filesystem Page Template	Page Template
`.cpt`	Controller Filesystem Page Template	Controller Page Template
`.py`	Filesystem Script (Python)	Script (Python)
`.cpy`	Controller Python Script	Controller Python Script
`.vpy`	Controller Validator	Controller Validator
`.doc, .pdf, .swf, .jar, .cab, .ico, .js, .css`	Filesystem File	File
`.gif, .jpg, .jpeg, .png`	Filesystem Image	Image
`.props`	Filesystem Properties Object	Folder with Properties
`.zsql`	Filesystem Z SQL Method	ZSQL Method
`.dtml`	Filesystem DTML Method	DTML Method

Tabelle 7.4: Erweiterungen

Um also ein Bild in Ihrer Verzeichnisansicht zu erhalten, stellen Sie eine `.gif`- oder `.jpeg`-Datei hinein. Wenn Sie ein Script(Python)-Objekt wollen, fügen Sie eine Datei mit der Endung `.py` hinzu.

Metadaten von Dateisystemobjekten setzen

Zusätzliche Informationen zu einem Objekt wie Titel, Sicherheit oder Cache werden in einer separaten Datei gespeichert. Diese Datei hat den gleichen Namen wie die Originaldatei, hat aber am Ende die Erweiterung `.metadata`. Falls die Originaldatei z.B. `logo.jpg` lautet, so liegen ihre Metadaten in der Datei `logo.jpg.metadata`.

Die Metadaten-Datei verwendet das Windows-Format der `.ini`-Dateien mit `key = value`-Paaren darin. Dieses Format wurde um Angaben zu Formularen für das `Form Controller`-Objekt erweitert, was Sie im nächsten Abschnitt sehen werden. Alle Angaben und sogar das Vorhandensein dieser Datei selbst sind optional. Es folgt eine Beispieldatei:

```
[default]
title = Testobjekt
cache = RAMCache
proxy = Manager

[security]
Access contents information = 1:Manager,Anonymous
```

In dieser Datei können Sie folgende Werte setzen:

- **title**: Dies ist der Titel, der dem Objekt im ZMI und in Plone zugewiesen wird. Er erscheint auch in den Plone-Templates.
- **cache**: Dies ist die ID des Cache-Objekts, in dem Sie das Objekt cachen möchten. Plone enthält standardmäßig zwei Cache-Objekte: einen RAM-Cache-Manager und einen HTTP-Cache-Manager. Kapitel 14 beschreibt die Funktion dieser zwei Objekte.
- **proxy**: Dies ist die Proxy-Rolle, die Sie auf dieses Objekt anwenden möchten. In Kapitel 9 finden Sie weitere Informationen dazu.
- **security**: Dies ist der Sicherheitsbereich, in dem mehrere Zeilen mit Sicherheitseinstellungen stehen können. Der Schlüssel enthält den Namen der Berechtigung, und die rechte Seite enthält die Akquisitionseinstellungen, gefolgt von den durch Kommata getrennten Rollen. Beispielsweise bedeutet `View = 0:Manager`, dass nur Benutzer mit den Rollen Mitglied und Manager ein Objekt sehen können und dass Sicherheitseinstellungen für diese Berechtigung nicht akquiriert werden.

Validierer im Dateisystem verwenden

Um einen Validierer im Dateisystem anzugeben, fügen Sie den Validierer in der `.metadata`-Datei hinzu. Der Validierer-Abschnitt der `.metadata`-Datei sähe wie folgt aus:

```
[validators]
validators = validate_script1, validate_script2
```

Damit werden die zwei Validierungsscripts `validate_script1` und `validate_script2` ausgeführt, und zwar in dieser Reihenfolge. Ein Validierungsscript wird die Daten untersuchen und Fehler zum Formular-Controller-Status hinzufügen, falls es Probleme gibt.

Die Optionen `contextType` und `button` benötigen eine leicht andere Syntax. Validierungen werden im gerade ausgeführten Kontext, z.B. einem Dokument oder Bild, ausgeführt. Sie könnten jeweils verschiedene Validierer für ein Dokument und für ein Bild ausführen. Um z.B. ein anderes Validierungsscript auszuführen, wenn es als Dokument ausgeführt wird, fügen Sie folgende Zeile hinzu:

```
validators.Document = validate_script2
```

Den Validierer können Sie abhängig von dem angeklickten Formular-Button variieren, indem Sie den Namen des Buttons im Formular links vom Validierer hinzufügen. Der Button-Name muss mit `form.button` beginnen. Beispiel:

```
<input type="submit" name="form.button.button1" value="First" />
```

Die Metadaten-Datei sähe dann wie folgt aus:

```
validators..button1 = validate_script1
```

Das `..` steht für ein Leerzeichen beim Kontexttyp. Wenn Sie also wie zuvor möchten, dass dieses Leerzeichen bei `button1` in einem Dokument erscheint, dann sähe die Metadaten-Datei wie folgt aus:

```
validators.Document.button1 = validate_script5
```

Aktionen im Dateisystem verwenden

Aktionen können Sie, wie Validierer auch, in der `.metadata`-Datei angeben. Die Syntax für den Abschnitt `actions` Ihrer Datei sähe wie folgt aus:

```
[actions]
action.success = traverse_to:string:script1
```

Wenn im vorherigen Beispiel das Formular abgeschickt wird und die Validierungsscripts einen Erfolg zurückgeben, wird die Aktion `traverse to` mit dem Argument `string:script1` aufgerufen. Dieses Argument ist eigentlich ein Aus-

druck. Die Standardaktion bei einem Fehlschlag besteht darin, das aktuelle Formular neu zu laden. Das Formular hat Zugriff auf alle Fehlermeldungen über das `state`-Objekt in seinen Optionen.

Noch einmal: Sie können eine bestimmte Aktion in einem bestimmten Kontext angeben. Um z.B. eine Aktion bei Erfolg auf einem Dokument anzugeben, können Sie Folgendes tun:

```
action.success.Documnent = traverse_to:string:document_script
```

Wieder können Sie die Aktion für den folgenden Button angeben:

```
<input type="submit" name="form.button.button1" value="Button" />
```

Dazu fügen Sie Folgendes zur `.metadata`-Datei hinzu:

```
action.success..button1 = traverse_to:string:script1
```

Dieses Beispiel enthält keinen expliziten Kontext, d.h., es ist für alle Kontexttypen gültig.

Exkurs: Beispiele für Dateisystem-Skins und die Buchbeispiele

Alle Beispiele in diesem Buch wurden in einer Skin gesammelt, die Sie installieren können. Sie finden sie auf der Website zum Plone-Buch unter *http://plone-book.agmweb.ca/Software/PloneBook-Examples*. Die Skin ist verfügbar, nachdem Sie die `.zip`-Datei heruntergeladen und ausgepackt haben. Sie werden eine Dateistruktur ähnlich zu der zuvor erwähnten finden.

Sie werden eine Datei namens `__init__.py` und ein Verzeichnis namens `skins` vorfinden. In diesem Verzeichnis befinden sich eine Reihe von Page Templates, `Controller Validator`-Objekten und alle entsprechenden Metadaten-Dateien. Wenn Sie das installieren möchten, kopieren Sie den Ordner `PloneBookExamples` in das Verzeichnis `Products` der Wurzel Ihrer Instanz. Starten Sie Plone neu, und klicken Sie dann auf PLONE KONFIGURATION. Wählen Sie PRODUKTE HINZUFÜGEN/LÖSCHEN, und Sie werden den Eintrag PLONEBOOKEXAMPLES sehen. Wählen Sie ihn, und klicken Sie dann auf INSTALLIEREN. Nun haben Sie die Templates installiert und können zur `feedbackForm` gehen. Sie erhalten das Page Template, das Sie im vorigen Kapitel gesehen haben.

Die Installationsprozedur hat den Vorgang automatisiert, wobei erst ein FSDV und dann für jede Skin eine Ebene hinzufügt wird. Wenn Sie PORTAL_SKINS anklicken und dann den Properties-Reiter wählen, werden Sie sehen, dass die neue Ebene `plone_book_examples` hinzugefügt wurde.

7.4 Fallstudie: Die NASA-Skin

Im Januar 2004 landeten zwei NASA-Sonden auf dem Mars: *Spirit* und *Opportunity*. Diese ferngesteuerten Roboter klapperten die Mars-Oberfläche ab und sendeten Bilder und Oberflächenanalysen zurück. Die Sonden waren ein großer Erfolg und brachten erstaunliche Bilder von der Mars-Oberfläche, die die ganze Welt begeisterten.

Ein kleiner Teil dieses Ereignisses war eine Website mit der Adresse *http://mars.telascience.org*. Sie machte ein Programm namens *Maestro* publik, dessen Zweck folgendes Zitat klarmacht:

> *Sie können die reduzierte Version des Programms herunter laden, das die NASA-Wissenschaftler zur Steuerung von Spirit und Opportunity einsetzen. Es sind auch Updates für Maersto erhältlich, die authentische Daten vom Mars enthalten, die Sie Ihrer Maestro-Installation hinzufügen können.*

Das hat insofern mit Plone zu tun, weil die für die Site verantwortliche Gruppe sehr schnell und leicht eine Site entwickelt hat, die toll aussieht. In diesem Fall hat eine große Zahl von Community-Mitgliedern und Freiwilligen den Maestro-Teammitgliedern bei der Entwicklung der Site geholfen. Abbildung 7.14 zeigt die Plone-Site in Betrieb.

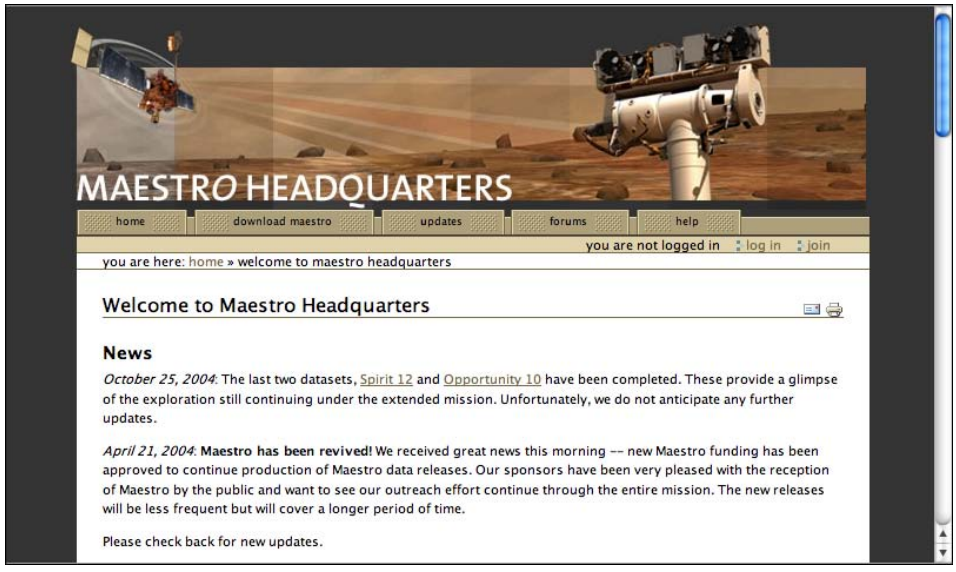


Abbildung 7.14: Die Maestro-Site

Vermutlich werden Sie einige Hinweise auf eine Plone-Site erkennen: die Reiter im oberen Bereich, die persönliche Leiste in der oberen rechten Ecke und die übliche Pfadnavigation. Abgesehen davon, sieht die Site recht anders als eine Stan-

dard-Plone-Site aus. In den folgenden Abschnitten werde ich die Schritte durchgehen, mit denen das erreicht wurde. Nun, eigentlich ist es sehr einfach, weil das meiste vom Look-and-Feel mit CSS realisiert wurde. Es gab wenige bis gar keine Änderungen, außer am `custom`-Stylesheet und einige neue Bilder.

Zuerst betrachten wir die Änderungen außerhalb von CSS an der Site, nämlich an einigen Templates und Eigenschaften.

7.4.1 Portlets und einige Hauptelemente entfernen

Die Site verwendet keine Portlets. Sie wurden entfernt, weil es für diese Site keine relevanten Portlets gibt. Nachrichten erscheinen stattdessen auf der Homepage. Um die Portlets von Ihrer Site zu entfernen, gehen Sie zur Wurzel der Plone-Site und klicken auf den PROPERTIES-Reiter. Löschen Sie alle Werte in den Formularfeldern neben `left_slots` und `right_slots`.

Auf der Maestro-Site wurden ein paar Elemente entfernt. Manchmal habe ich festgestellt, dass es das Beste ist, was man mit Merkmalen tun kann, die auf einer Site einfach nicht benötigt werden. Es kann schwierig sein, jedes Element einer Benutzerschnittstelle in eine Plone-Site zu zwängen, aber das müssen Sie gar nicht immer tun. Entfernen Sie stattdessen einfach die unnötigen Elemente. In diesem Fall wurden auch einige Elemente entfernt: die Site-Aktionen, der SUCHEN-Kasten, die Fußzeile sowie das Kolophon.

Um das zu bewerkstelligen, wurden die Templates, die den Code produzieren, angepasst und so verändert, dass sie nichts darstellen. Um z.B. den SUCHEN-Kasten zu entfernen, klicken Sie im ZMI nacheinander auf PORTAL_SKINS, PLONE_TEMPLATES und GLOBAL_SEARCHBOX. Klicken Sie dann auf den CUSTOMIZE-Button, und ändern Sie das Template wie folgt:

```
<html
    xmlns="http://www.w3.org/1999/xhtml"
    xml:lang="en" lang="en"
    i18n:domain="plone">
    <body>

        <div id="portal-searchbox"
               metal:define-macro="quick_search"
               tal:condition="nothing">
            Nothing to see here.
        </div>
    </body>
</html>
```

Diese Technik beim Entfernen von Elementen habe ich weiter oben gezeigt. Setzen Sie einfach `tal:condition` im Makro-Element, um sicherzustellen, dass die Bedingung `false` ist.

7.4.2 Farben anpassen

Im Objekt `base_properties` haben Sie die Grundfarben der Site gesetzt. Dieses Objekt wurde angepasst, und die Farben wurden wie folgt geändert (alle weiteren Elemente sind unverändert, sofern nichts anderes vermerkt ist):

```
linkColor: #776a44
globalBorderColor: #776a44
globalBackgroundColor: #e0d3ad
globalFontColor: #776a44
```

Die Farbänderung, die ich am stärksten bemerkt habe, ist die von `globalBackgroundColor`, was die Farbe der persönlichen Leiste von Blau auf Bräunlich ändert. Diese geringfügigen Farbänderungen ändern das Grund-Stylesheet, damit es besser zu den Bildern und dem allgemeinen Look-and-Feel passt.

7.4.3 Stylesheet erstellen

Der größte Teil dieser Site ist das Stylesheet, das in Anhang B vollständig wiedergegeben ist. Hier möchte ich nur einige wesentliche Teile davon hervorheben. Dieses Stylesheet basiert auf `ploneCustom`.css, das im `custom`-Ordner angepasst wurde. Dann wurden einige Elemente der Webseite in der neuen Datei `ploneCustom.css` überschrieben.

Zuerst wird die Hintergrundfarbe für den ganzen Hauptteil auf `#343434` gesetzt.

```
body {
    background: #343434;
}
```

Zweitens gilt, dass sich der eigentliche Inhalt einer Plone-Seite, der Teil, den Sie bearbeiten können, in einer Klasse namens `documentContent` befindet. Da die Hintergrundfarbe des Elements `documentContent` in der Hauptdatei `plone.css` auf `white` gesetzt ist, hat der Text einen weißen Hintergrund und bildet in der Mitte des Bildschirms einen weißen Bereich.

Anschließend wird das Bild vom Satelliten und Roboter als großes Bild mittels CSS oben auf der Website platziert. Der Code dafür ist folgender:

```
#portal-top {
   background: url("http://mars.telascience.org/header.jpg") transparent no-
repeat;
   padding: 162px 0 0 0;
   position: relative;
}
```

Dieser CSS-Code setzt die Parameter für das Element mit der ID `portal-top`. Wenn Sie sich den HTML-Code einer Plone-Site anschauen, werden Sie oben auf der Seite direkt unter dem `body`-Element das Element `portal-top` sehen. Dadurch, dass der Hintergrund für dieses Bild auf den URL des entsprechenden Bildes gesetzt wird, erscheint dieses Bild. Es ist 162 Pixel hoch, weswegen im `padding`-Wert der obere Wert des `#portal-top`-Elements auf `162px` gesetzt ist. Ohne diese Einstellung wurden alle Bilder darunter nach oben geschoben und das Bild überschreiben.

Das Titelbild ist 677 Pixel breit, und Sie werden bemerken, dass der Text auf der Seite sauber unter das Bild passt und nicht links oder rechts darüber hinaussteht. Das erreichen Sie dadurch, dass Sie den Wert des Elements auf `680px` setzen. Das HTML-Element `visual-portal-wrapper` ist direkt unter dem Hauptteil und setzt die Breite für die ganze Seite. Der Code dafür lautet wie folgt:

```
#visual-portal-wrapper {
   width: 680px;
   margin: 1em auto 0 auto;
}
```

Dies setzt die Breite aller Seiten auf einen festen Wert, was in Ordnung ist, solange Sie sicherstellen, dass die Breite kleiner als die Standardbreite von 800 Pixel ist. Egal wie groß der Benutzer das Browserfenster macht, der Hauptteil der Seite wird nie breiter als 680 Pixel, was garantiert, dass er passend zum Bild bleibt.

Die anderen offensichtlichen Änderungen sind die Reiter oben auf der Seite, die nun aus Bildern bestehen statt aus den Standardkästen in Plone. Die Reiter werden aus drei Bildern zusammengesetzt: einem Abstandshalter zwischen den Reitern sowie dem linken und dem rechten Teil eines Reiters. Die Kombination dieser drei Bilder ergibt den Effekt eines Reiters. Abbildung 7.15 zeigt diese drei Bilder.

Abbildung 7.15: Kombination dreier Bilder für den Reiter

Denken Sie beim Bearbeiten des CSS daran, dass jeder Reiter nichts weiter als ein Listeneintrag mit einem Link in einem Element mit der ID `portal-globalnav` ist. Um den Hintergrund-Abstandshalter zwischen den Reitern zu setzen, setzt die Skin zuerst den Hintergrund für das gesamte Element. Auch hier gilt, dass Sie durch das Setzen der Höhe auf 21 Pixel, d.h. auf die gleiche Größe wie beim Bild, garantieren, dass es ausreichend Platz für das Bild gibt. Der Code dafür ist folgender:

```
#portal-globalnav {
    background: url("http://mars.telascience.org/listspacer.gif") transparent;
    padding: 0;
    height: 21px;
    border: 0;
    margin: 0 0 1px 6px;
    clear: both;
}
```

Das Startbild verwenden Sie, um das Bild am linken Rand des Reiters zu setzen. Dieses setzen Sie, indem Sie den Wert des `li`-Elements statt des `anchor`-Elements wie folgt setzen:

```
#portal-globalnav li {
   display: block;
   float: left;
   height: 21px;
   background: url("/liststart.gif") transparent no-repeat;
   padding: 0 0 0 33px;
   margin: 0 0.5em 0 0;
}
```

Und schließlich setzen Sie den rechten Teil des Reiters, indem Sie ein Bild zum Ankerelement hinzufügen. Dazu ändern Sie das Ankerelement innerhalb des Reiters. Der folgende Code zeigt, wo Sie das Hintergrundbild als rechten Teil gesetzt haben:

```
#portal-globalnav li a {
    display: block;
    float: left;
    height: 21px;
    background: url("/listitem.gif") transparent right top;
    padding: 0 33px 0 0;
    border: 0;
    line-height: 2em;
    color: black;
    font-size: 90%;
    margin: 0;
}
```

Nun haben Sie die ziemlich standardmäßig aussehenden Plone-Reiter durch tolle Buttons ersetzt.

7.4.4 Eine Splash-Seite erstellen

Diese Site verfügt über ein weiteres Schlüsselmerkmal. Die Eingangsseite der Site ist eine *Splash-Seite*, die eine hübsche Grafik anzeigt und den Benutzer dazu einlädt, die Site zu betreten. Eine solche Seite können Sie hinzufügen, indem Sie ins ZMI gehen und das Objekt `index_html` entfernen, das normalerweise vorhanden ist. Erstellen Sie dann eine neue Datei namens `index_html`. Darin schreiben Sie einen eigenen Code, um die Homepage zu erzeugen, inklusive eines eigenen CSS. Das Hauptelement davon ist ein Bild, das mit folgendem CSS-Code dort platziert wird:

```
div {
    background: url(/splash.jpg) transparent no-repeat;
    width: 260px;
    height: 335px;
    position: absolute;
    ...
}
```

Das restliche CSS platziert den Text und die Links innerhalb des Bildes. Diese Seite enthält keinerlei Plone-Elemente, sondern ist statisches HTML.

7.4.5 Schlussfolgerung

Das sieht nach einer einigermaßen komplexen Site aus, wobei der relativ einfache CSS-Code die meiste Arbeit erledigt. Durch die Verwendung von CSS und mit einigen HTML-Kenntnissen haben Sie das Look-and-Feel von Plone verändert, ohne viel über Plone wissen zu müssen. Durch die Platzierung der Bilder mittels CSS haben Sie außerdem wesentliche Eigenschaften beibehalten, die für die Zugänglichkeit der Site sorgen.

Ein großes Dankeschön geht an die NASA und alle beteiligten Leute aus der Plone-Community für ihre Hilfe bei dieser Site und Fallstudie. Dazu gehören besonders, aber nicht ausschließlich, John Graham, Alma Ong, Joe Geldart, Michael Zeltner und Tom Croucher.

8 Workflows verwalten

Eine der vielen Stärken von Plone liegt in seiner Workflow-Komponente. Workflows sind ein zentrales Thema beim *Content-Management*, in dem es um die Trennung von Anwendungslogik, Inhalt und Präsentation geht. Deswegen wird in diesem Kapitel der Workflow in Plone detailliert behandelt.

Ich beginne mit einigen Schlüsseldefinitionen im Zusammenhang mit Workflows sowie mit den wichtigsten dabei beteiligten Werkzeugen zum Erstellen von Workflows. Nachdem diese Konzepte klar sind, werde ich dann beschreiben, wie Sie Ihre eigenen Workflows hinzufügen und bearbeiten können.

Über das gesamte Kapitel hinweg werde ich Ihnen einfache Änderungen zeigen, die Sie an jenem Workflow vornehmen können, der in Plone von vornherein enthalten ist. Außerdem werde ich auch eine Reihe von Beispielen zeigen, um Ihnen bei Aufgaben wie der Erstellung von Benachrichtigungen, beim Verschieben von Inhalten usw. zu helfen. Und schließlich werde ich einige fortgeschrittenere Eigenschaften bei der Entwicklung von Workflows sowie einige nützliche Werkzeuge dabei vorstellen.

8.1 Was ist ein Workflow?

Unter einem *Workflow* versteht man eine Kette von Aktionen oder Ereignissen, die sich abspielen, um ein bestimmtes Ziel zu erreichen. Oftmals wird eine Geschäftslogik mit einem Workflow ausgedrückt. Alle Geschäftsbereiche haben ihre eigenen Regeln und Vorschriften zu Vorgängen, die sich in einer Firma so ereignen. Hier sind einige Beispiele dafür:

- Bevor die Arbeitsstundenaufstellung eines Mitarbeiters genehmigt wird, muss sie von einem Vorgesetzten angeschaut und bestätigt werden.
- In einer Fabrik, die irgendwelche Teile herstellt, müssen die Benutzer für jedes zusammengesetzte Teil über die Bearbeitungsreihenfolge und Zustandsänderungen auf dem Weg durch die Fabrik benachrichtigt werden.

- Bevor eine Seite auf einer Website veröffentlicht wird, muss sie von der Marketingabteilung und vom Webmaster genehmigt und von einem Linguisten übersetzt werden.

Der Workflow kapselt die Geschäftslogik dieser Regeln und standardisiert die Art und Weise, über diese Vorgänge nachzudenken. Durch diese isolierte Geschäftslogik können Firmen eine Anwendung nun leicht gemäß ihrer Tätigkeit und den entsprechenden Prozessen abwandeln. Normale Anwendungen erzwingen oftmals einen Workflow in einem Geschäftsbereich, weil der Workflow darin fest kodiert ist.

8.2 Workflow in Plone

Das Workflow-Werkzeug von Plone bietet gewisse Eigenschaften und Beschränkungen, die man kennen muss, um Workflows in Plone zu verstehen. Das in Plone benutzte Workflow-Produkt ist DCWorkflow, ein Open Source-Produkt der Zope Corporation. Es sind auch weitere Workflow-Systeme verfügbar, und einige davon werden in Plone eingesetzt, z.B. CMFOpenFlow, ein aktivitätsbasierender Workflow (*http://www.reflab.it/community/Openprojects/CMFOpenflow*). Im Moment ist DCWorkflow jedoch mächtig und einfach genug, um alle von den meisten Benutzern gewünschten Eigenschaften zu bieten.

DCWorkflow geht davon aus, dass im System ein Objekt existiert, das als Ziel des Workflows fungiert, z.B. ein gewisser Inhalt oder ein Widget. Weiterhin wird angenommen, dass alle Objekte desselben Typs vom selben Workflow erfasst werden. Durch das Umwidmen (engl. Repurposing) von Inhalten (Details dazu folgen in Kapitel 11) können Sie ähnliche Inhalte mit verschiedenen Workflows bearbeiten.

Da DCWorkflow in Plone enthalten ist, gibt es nichts zusätzlich zu installieren. Im Zope Management Interface (ZMI) wird es vom Objekt `portal_workflow` dargestellt.

8.2.1 Konzipieren eines Workflows

Bevor ich einen Workflow erkläre, werde ich erst ein wenig die Terminologie dazu erklären: die Begriffe *Zustände* und *Übergänge*.

Ein *Zustand* (*state*) ist eine Information über ein Inhaltselement zu einem bestimmten Zeitpunkt. Beispielzustände sind *Privat*, *Veröffentlicht*, *Offen* und *Entwurf*. Alle Workflows verfügen über einen Ausgangszustand, in dem alle Inhalte starten. Der Workflow verschiebt den Inhalt dann durch eine Reihe von Zuständen, und zwar entweder durch eine Interaktion mit dem Benutzer oder durch einen automatisierten Prozess. Wenn der Inhalt einen Endzustand erreicht hat,

bleibt er für lange Zeit in diesem Zustand (normalerweise für immer). Während der Workflow aktiv ist, kann ein Inhalt auch einen von mehreren verschiedenen Endzuständen erreichen.

Damit dieser Inhalt von einem Zustand zum anderen gelangt, muss ein *Übergang* (engl. *Transition*) dazwischen vorhanden sein. Ein Übergang verbindet einen Start- mit einem Endzustand. Mit einem Übergang können viele verschiedene Eigenschaften verbunden sein, wie Sie später sehen werden, aber im Moment müssen Sie nur wissen, dass ein Übergang einen Inhalt zwischen zwei Zuständen bewegt. Normalerweise wird ein Übergang durch eine äußere Einwirkung ausgelöst, z.B. von einem Benutzer, der auf einer Webseite einen Button anklickt, oder von einem Script, das von einer Seite aus aufgerufen wird.

Die Visualisierung eines Workflows kann etwas verwirrend sein, besonders dann, wenn man über etwas so Vages wie einen Inhalt spricht. Daher ist es hilfreich, an ein alltägliches Beispiel zu denken. In diesem Fall zeigt das folgende Beispiel den Workflow meiner Kreditkartenrechnung, die mich einmal im Monat beglückt:

1. Die Kreditkartenfirma erstellt eine Rechnung und sendet sie per Post an mich.
2. Ich erhalte die Rechnung und lege sie auf meinen Schreibtisch. Manchmal liegt die Rechnung dort eine ganze Weile, während ich bis zum Monatsende warte. Gelegentlich muss ich bei Leuten wegen bestimmter Ausgaben nachfragen, z.B. »Was für Kleider hast du da gekauft?«
3. Alle ungeklärten Probleme gehen an die Kreditkartenfirma weiter, die möglicherweise eine neue Rechnung ausstellt (was allerdings sehr selten passiert).
4. Am Monatsende, wenn ich normalerweise die ganze Buchhaltung mache, bezahle ich dann die Rechnung.

Aus dieser Beschreibung können Sie nun ein paar Zustände ableiten. Wenn Sie sich die obigen Schritte anschauen, stellen Sie fest, dass Sie wirklich keinen Bedarf für verschiedene Zustände beim Erhalt der Rechnung haben, wozu auch das Öffnen und Ablegen auf meinen Schreibtisch gehört. Ebenso wenig müssen Sie sich mit allen Überprüfungen herumschlagen, die vorgenommen werden. Das sind zwar alles wichtige Schritte, die ausgeführt werden, aber einen Workflow für jeden einzelnen Schritt zu erstellen wäre bei weitem zu umständlich. Sie können stattdessen den Workflow mit den folgenden Zuständen zusammenfassen:

- **Entwurf**: Die Kreditkartenrechnung wurde erstellt und mir zugeschickt.
- **Überprüfung**: Die Kreditkartenrechnung wurde empfangen, liegt auf meinem Schreibtisch und wird geprüft.
- **Bezahlt**: Die Kreditkartenrechnung wurde bezahlt, ins Regal abgelegt und für immer vergessen.

Nun haben Sie die Zustände gefunden und können sich Gedanken zu den notwendigen Übergängen machen. Zu jedem Zustand gibt es mindestens einen Übergang, der die Rechnung von einem Zustand in einen anderen bewegt:

- **Verschicken**: Die Bank verschickt die Kreditkartenrechnung.
- **Bezahlen**: Ich bezahle die Kreditkartenrechnung.
- **Ablehnen**: Etwas auf der Rechnung stimmt nicht, und sie wird nicht genehmigt.

Abbildung 8.1 zeigt diesen Satz von Zuständen und Übergängen. In dieser Abbildung werden Zustände mit Namen von Kästen dargestellt, und Zustandsübergänge werden von Pfeilen mit kursiv geschriebenen Namen dargestellt.

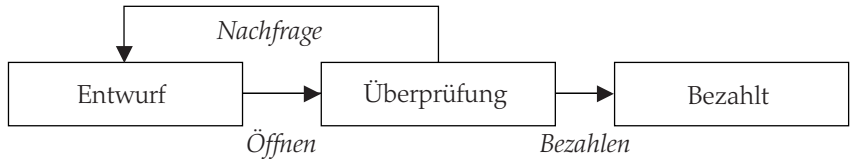


Abbildung 8.1: Einfacher Zustandsautomat bei der Bezahlung von Kreditkartenrechnungen

Jetzt haben Sie den Geschäftsvorgang der Bezahlung einer Kreditkartenrechnung in einem Workflow extrahiert. Der nächste Schritt besteht im Nachdenken über Rollen und Sicherheit für diese Kreditkartenrechnung. Dieser Workflow enthält nun die Geschäftslogik einer Anwendung zur Bearbeitung von Kreditkartenrechnungen.

8.2.2 Rollen und Sicherheit in Workflows

In jedem komplizierten System haben Sie Benutzer mit allen möglichen Rollen und Gruppen. Diese Rollen verleihen Plone eine große Flexibilität bei der Sicherheit, können es aber auch komplizierter machen. Kapitel 9 behandelt lokale Rollen und Gruppen, aber dieser Abschnitt behandelt einige wichtige Punkte dazu, wie diese Themen mit dem Workflow zusammenhängen.

Wenn ein Inhaltselement von einem Workflow-Zustand in einen anderen übergeht, kann der Workflow-Prozess die Sicherheitseinstellungen auf diesem Inhalt ändern. Diese Sicherheitseinstellungen bestimmen, welcher Benutzer welche Aktion auf welchem Inhalt ausführen kann. Durch die Manipulation der Sicherheitseinstellungen über den Workflow kann man die Sicherheit eines Inhaltselements über seinen Lebenszyklus hinweg ändern. Benutzer von statischen Systemen oder von Zope sind oftmals verwirrt, weil in Zope alle Inhaltselemente über ihren gesamten Lebenszyklus die gleichen Sicherheitseinstellungen haben.

Was das Kreditkartenbeispiel angeht, können Sie die Sicherheitseinstellungen für die Kreditkartenrechnung ableiten. Eine Möglichkeit, diese darzustellen, besteht darin, eine Tabelle anzulegen, die die Sicherheit für die Übergänge zwischen verschiedenen Zuständen allgemein erweitert, wie in Tabelle 8.1 zu sehen ist.

Zustand	Ich	Bank
Entwurf		Abschicken
Überprüfung	Bezahlen	Ablehnen
Bezahlt		

Tabelle 8.1: Übergänge und ihre erzeugenden Zustände

In diesem Stadium sehen Sie in Tabelle 8.1 die Übergänge und wer sie erzeugen kann. Sie haben aber noch nicht über den Zugriff nachgedacht, der immer dann erfolgt, wenn ein Benutzer eine Aktion auf einem Objekt ausführt. Wann kann jemand z.B. die Rechnung bearbeiten, und wann kann sie angezeigt werden? Plone verwendet hierfür den Begriff *Aktion*, wie in Tabelle 8.2 zu sehen ist. Hoffentlich habe nur ich Zugriff auf meine eigenen Kreditkartenauszüge! Ebenso kann die Bank die Kreditkartenrechnung jederzeit anzeigen und Fragen dazu beantworten.

Zustand	Ich	Bank
Entwurf		Anzeigen, Bearbeiten
Überprüfung	Anzeigen	Anzeigen
Bezahlt	Anzeigen	Anzeigen

Tabelle 8.2: Aktionen und ihre erzeugenden Zustände

Tatsächlich stellt sich heraus, dass ich meine Kreditkartenrechnung nicht bearbeiten kann, nur die Bank kann das. Ich kann meine Rechnung ablehnen und zurückschicken, aber die Bank will sicher nicht, dass ich die Rechnung bearbeiten kann. In dieser Situation nehmen Sie an, dass die Bank der Besitzer der Kreditkartenrechnung ist. Hiermit wird das Konzept eines *Besitzers* (engl. *Owner*) demonstriert. Ich habe evtl. mehrere Kreditkartenrechnungen von mehreren Banken, und bei jeder können Sie sich die Bank als deren Besitzer vorstellen. Jede Bank besitzt ihre eigenen Kreditkartenrechnungen, aber nicht die anderer Banken. Tabelle 8.3 kombiniert Übergänge und Aktionen, wobei die Begriffe *Ich* und *Bank* jeweils in *Bezahler* und *Besitzer* geändert wurden.

Zustand	Bezahler	Besitzer
Entwurf		Abschicken, Anzeigen, Bearbeiten
Überprüfung	Bezahlen, Ablehnen, Anzeigen	Anzeigen
Bezahlt	Anzeigen	Anzeigen

Tabelle 8.3: Übergänge und Aktionen kombiniert, plus Rollen der Leute

Natürlich ist das ein sehr gestelltes Beispiel, aber es illustriert, wie Sie einen Workflow auf einfache Zustände anwenden können. Es können hier auch weitere Übergänge erfolgen, z.B. wäre ich sehr froh, wenn meine Kreditkartenrechnungen von jemand anderem bezahlt würden, aber das ist leider so unwahrscheinlich, dass Sie das nicht zum Workflow oder zur Sicherheit aufnehmen sollten.

Bevor ich Ihnen zeige, wie Sie Workflows erstellen und bearbeiten, zeige ich Ihnen jetzt noch die Standard-Workflows, die in Plone enthalten sind.

8.2.3 Einführung in Plone-Workflows

Plone enthält bereits eine Reihe von Standard-Workflows für Ihre Plone-Site. Diese Workflows bieten eine logische Art und Weise, Inhalte durch eine Plone-Site zu schleusen. Eine Standard-Plone-Site enthält zwei Workflows: den Default-Workflow und den Ordner-Workflow (Folder-Workflow). In den folgenden Abschnitten werden beide vorgestellt.

Default-Workflow

In Kapitel 3 habe ich den Default-Workflow und die Standardeinstellungen bei der Veröffentlichung von Inhalten behandelt. Dort habe ich für jeden Zustand im Workflow dessen Sicherheit und Einstellungen beschrieben. Aber da ein Bild mehr sagt als tausend Worte, sehen Sie in Abbildung 8.2 diese Workflow-Zustände.

Hinweis

Der Besitzer des Inhalts ist jene Person, die den Inhalt ursprünglich erstellt hat. Ein Besitzer ist ein bestimmtes Mitglied einer Plone-Site. Zwar existieren auf einer Plone-Site viele Mitglieder, aber nur eine Person kann Besitzer eines Inhalts auf einer Plone-Site sein. Diese Besitzerrolle ist insofern speziell, als sie sich erst dann ergibt, wenn ein Objekt angelegt wird.

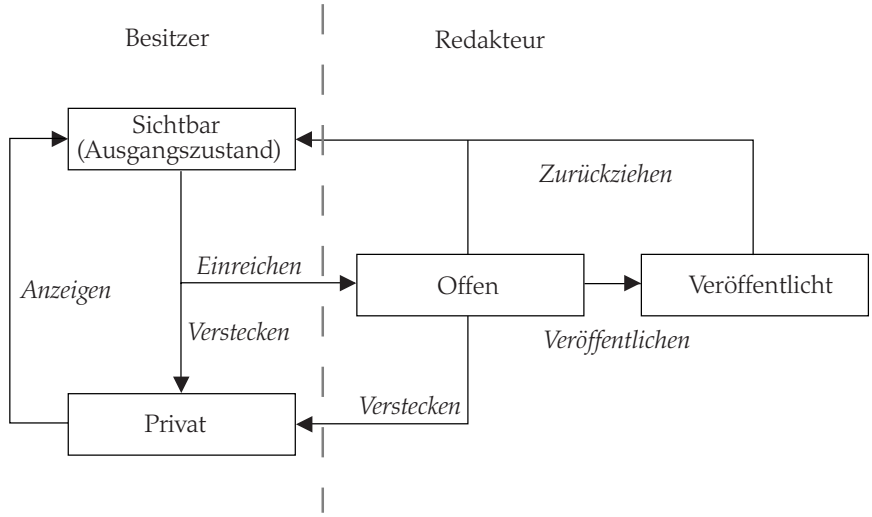


Abbildung 8.2: Default-Workflow in Plone für Inhalte

Abbildung 8.2 zeigt die Hauptzustände und -übergänge. Die grau gepunktete Linie in dieser Abbildung stellt eine Art Trennung zwischen Sicherheitsbereichen dar. Links von dieser Linie sind es die Besitzer von Inhalten, die damit interagieren, und rechts von der Linie interagieren normalerweise die Prüfer mit den Inhalten.

Wie beim Kreditkartenbeispiel ist auch hier ein zugehöriger Satz an Rechten für den Default-Workflow vorhanden. Tabelle 8.4 gibt alle Rechte und Zustände an.

Zustand	Anonym	Authentifiziert	Besitzer	Manager	Prüfer
Offen	Anzeigen	Anzeigen	Anzeigen	Bearbeiten	Bearbeiten
Privat			Bearbeiten	Bearbeiten	Anzeigen
Veröffentlicht	Anzeigen	Anzeigen	Anzeigen	Bearbeiten	Anzeigen
Sichtbar	Anzeigen	Anzeigen	Bearbeiten	Bearbeiten	Anzeigen
Anzeigen bezieht sich auf folgende Rechte: Access, Contents, Information und View					
Bearbeiten bezieht sich auf folgende Rechte: Modify, Portal und Content					

Tabelle 8.4: Rechte im Default-Workflow

Wie Sie in Tabelle 8.4 sehen können, werden Inhalte nur dann standardmäßig vor anderen verborgen, wenn sie im Zustand *Privat* sind. Inhalte im Zustand *Veröffentlicht* kann nur der Manager bearbeiten. Im Abschnitt »Rechte bearbeiten«, der weiter unten folgt, werde ich Ihnen zeigen, wie Sie diese Rechte ganz einfach über das Web ändern können.

Ordner-Workflow

Ebenfalls in Kapitel 3 habe ich den Ordner-Workflow beschrieben, als ich die Veröffentlichung von Inhalten behandelt habe. Aber wie ich dort bemerkt habe, gibt es für Ordner keinen offenen Zustand. Deswegen haben Sie einen etwas einfacheren Workflow, wie in Abbildung 8.3 gezeigt.

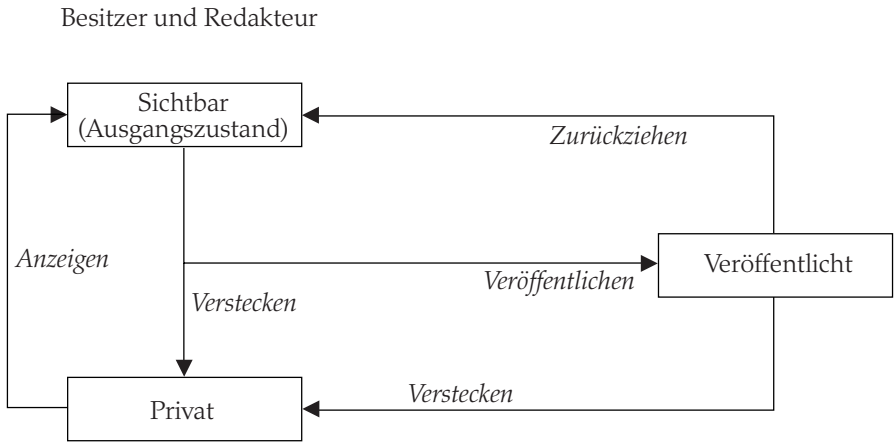


Abbildung 8.3: Ordner-Workflow in Plone für Inhalte

Weitere Workflows

Für eine Plone-Website sind zahlreiche Workflows verfügbar, darunter ein privater Workflow, ein Community-Workflow, ein One-Step Publication-Workflow usw. ZopeZen enthält einen Workflow, und PloneCollectorNG enthält ebenfalls einen Workflow. DCWorkflow enthält vier Workflows.

Zurzeit enthält das Produkt PloneWorkflows im Collective-Projekt auf SourceForge *(http://sf.net/projects/collective*) zwei Workflows: den Community-Workflow und den One-Step Publication-Workflow. Der Community-Workflow ähnelt bis auf ein paar Änderungen dem Plone-Workflow. Der One-Step Publication-Workflow verfügt über zwei Zustände: privat und veröffentlicht.

Im Moment gibt es keine einfache Möglichkeit, Workflows zu installieren bzw. zu deinstallieren, ebenso wenig wie es eine wirklich einfache Methode gibt, den Zustand von Inhalten zu wechseln. Wenn Sie z.B. den One-step-Publication-Workflow bei einem vorhandenen Zustand installieren, müssen Sie auch die Zustände für alle Objekte reparieren und sie in einen der neuen Zustände überführen. In diesem Fall ist das wahrscheinlich einfach: Alles im Zustand *Veröffentlicht* sollte so bleiben, wie es ist, und alles andere sollte in den Zustand *Privat* wechseln.

8.3 Workflows hinzufügen und bearbeiten

Nachdem ich den Default-Workflow beschrieben habe, komme ich zum Hauptpunkt, der Sie wahrscheinlich am meisten interessiert: Wie können Sie die Defaults ändern? Nun, wie bei den meisten Dingen in Plone können Sie alle Workflows über das ZMI erstellen, bearbeiten und löschen. Das Werkzeug, mit dem der Workflow gesteuert wird, lautet `portal_workflow`. In den folgenden Abschnitten behandle ich, wie Workflows zugewiesen werden, und gehe dann im Detail alle Einstellungen bei einem Workflow durch.

8.3.1 Workflows auf Inhaltstypen setzen

Nach einem Klick auf PORTAL_WORKFLOW sehen Sie eine Liste von Workflow-Zuweisungen. Ein Merkmal von DCWorkflow ist, das jedem Inhaltstyp genau ein Workflow zugewiesen ist. Abbildung 8.4 zeigt diese Zuweisungen.

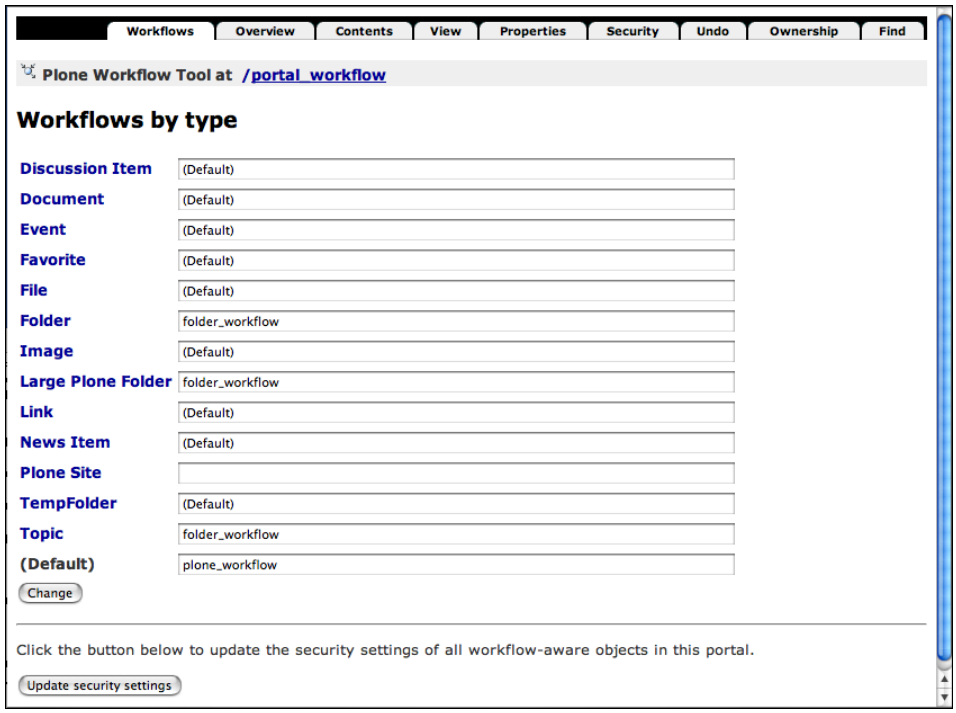


Abbildung 8.4: Die Workflow-Liste nach Typ

Auf dieser Seite sehen Sie eine Liste aller Inhaltstypen und des darauf angewendeten Workflows. Falls kein Workflow angegeben ist, d.h. falls der Wert leer ist, wird kein Workflow angewendet. Der Standardwert z.B. für den Typ Portal Site

(Plone Site) ist leer. Ganz bestimmt möchten Sie nicht den Zustand der Plone-Site selbst wechseln, sondern nur den Zustand der Objekte darin. Falls der Wert (Default) lautet, wird der Default-Workflow unten auf der Seite auf diesen Inhaltstyp angewendet. In Abbildung 8.4 wird der Workflow `folder_workflow` für Themen und Ordner benutzt, und bei allen anderen Inhaltstypen wird `plone_workflow` angewendet. Die Namen eines Workflows beziehen sich auf den Namen von Workflow-Objekten, die innerhalb des Workflow-Werkzeugs importiert oder erstellt werden. Weitere Informationen über verfügbare Workflows finden Sie, wenn Sie auf den CONTENTS-Reiter klicken. Danach wird eine Liste von Workflows geöffnet, die im System geladen sind, wie in Abbildung 8.5 zu sehen ist.

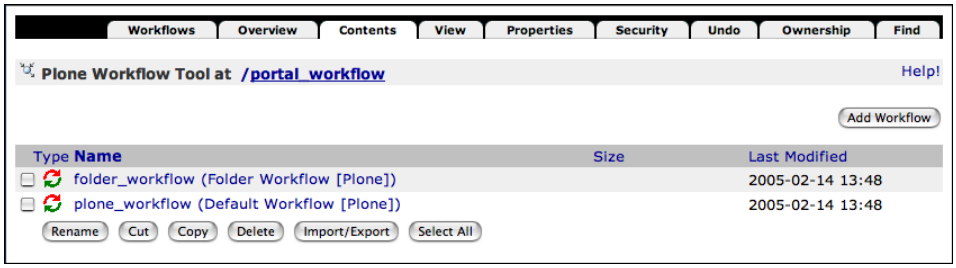


Abbildung 8.5: Verfügbare Workflows

Sie können Workflows hinzufügen, indem Sie auf den Button ADD WORKFLOW klicken. Dadurch wird eine Liste verfügbarer Workflows geöffnet. Um einen Workflow zu erstellen, wählen Sie einen Workflow-Typ und geben einen Workflow-Namen ein. Um einen leeren, aber über das Web konfigurierbaren Workflow zu erstellen, wählen Sie DC_WORKFLOW und geben einen passenden Namen ein, z.B. **mein_workflow**.

8.3.2 Bearbeiten eines Workflows

Unter dem CONTENTS-Reiter können Sie auf einen Workflow klicken, um zu den Management-Bildschirmen für alle Zustände, Übergänge und Eigenschaften zu diesem Workflow zu gelangen. Die Reiter oben auf der Seite beschreiben recht gut die Funktionalität eines Workflows: STATES (Zustände), TRANSITIONS (Übergänge), VARIABLES (Variablen), WORKLISTS, SCRIPTS und PERMISSIONS (Rechte). Ich werde all diese Reiter und einige andere verfügbare Optionen durchgehen. Falls nichts Gegenteiliges gesagt wird, sind alle folgenden Reiter von der Workflow-Hauptseite aus erreichbar.

Das Erstellen und Bearbeiten von Workflows verlangt mitunter viel Klickerei und kann ein wenig verwirrend sein. Wenn Sie als Entwickler scharf darauf sind, das Dateisystem zu benutzen, können Sie all das aus Python heraus tun, wenn Sie

möchten. Ich werde das im Abschnitt »Workflows in Python schreiben« gegen Ende des Kapitels behandeln.

Ausgangszustand setzen

Um den Ausgangszustand zu setzen, gehen Sie zum STATES-Reiter und prüfen die verfügbaren Zustände. Neben einem der Zustände werden Sie ein Sternchen wie in Abbildung 8.6 sehen.

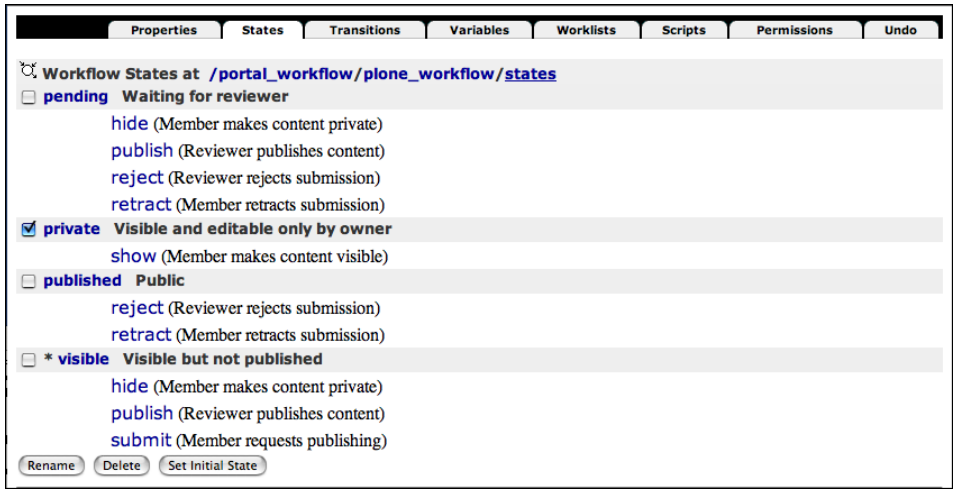


Abbildung 8.6: Ausgangszustand für diesen Workflow setzen

Auf dieser Seite setzen Sie den Ausgangszustand für Ihren Workflow, indem Sie das Kontrollkästchen neben dem Zustand markieren und dann auf SET INITIAL STATE klicken. Alle Inhalte, die diesen Workflow benutzen, werden in diesem Ausgangszustand erzeugt. Alle anderen schon erzeugten Inhalte bleiben in ihrem Zustand. Bei einer späteren Zustandsänderung wird dieser Zustand nicht verändert. Zu jedem Workflow können Sie nur einen Ausgangszustand einstellen.

Exkurs: Wie können Sie den Ausgangszustand auf Privat setzen?

Auf manchen Sites kann es Sinn machen, wenn gewisse Inhalte überhaupt nicht erscheinen oder für andere Benutzer als Administratoren und Besitzer erst dann zugänglich sind, wenn sie vollständig fertig sind. Das erreicht man am besten, indem man den Ausgangszustand des Objekts so wählt, das diese Sicherheit vorhanden ist, z.B. *Privat*. Im privaten Zustand können nur Redakteure, Manager und Besitzer das Objekt sehen.

Um den Ausgangszustand im ZMI auf *Privat* zu setzen, klicken Sie auf PORTAL_WORKFLOW und wählen den CONTENTS-Reiter. Klicken Sie dann auf PLONE_WORKFLOW, wählen Sie den STATES-Reiter, und wählen Sie anschließend den Ausgangszustand, indem Sie das Kontrollkästchen neben PRIVATE markieren. Am Ende klicken Sie auf den Button SAVE CHANGES. Nun werden neue Inhalte im Zustand *Privat* erstellt und sind für die Allgemeinheit nicht zugänglich.

Zustände bearbeiten

Unter dem STATES-Reiter werden die in diesem Workflow vorhandenen Zustände aufgelistet. Am Anfang dieses Kapitels habe ich erklärt, dass ein Zustand ein Objekt zu einem bestimmten Zeitpunkt darstellt. Jeder Zustand hat eine eindeutige ID, normalerweise ein einfaches Verb wie *pending* (offen) oder *published* (veröffentlicht). Um einen Zustand hinzuzufügen, geben Sie eine ID ein und klicken unten auf der Seite auf ADD.

Außerdem werden Sie die folgenden Optionen sehen:

- **Title**: Der Titel des Zustands wird in Ihrer Plone-Site angezeigt. Er ist ein benutzerfreundlicher Name für den Zustand.
- **Description**: Dies ist eine längere Beschreibung des Zustands. Momentan können die Benutzer diese Beschreibung nicht sehen, aber das kann sich in Zukunft noch ändern.
- **Possible transitions**: Hier werden die möglichen Übergänge aufgelistet, die aus diesem Zustand heraus erfolgen können. Diese Liste erscheint nur dann, wenn Sie tatsächlich einen Übergang im System haben. Wählen Sie einfach die Übergänge aus, die aus diesem Zustand erfolgen müssen. Durch die Wahl eines Übergangs für diesen Zustand machen Sie diesen Zustand zum Startpunkt des gewählten Übergangs.

Um einen Zustand zu ändern, geben Sie die Änderungen ein und klicken dann auf SAVE, um diese zu bestätigen. Der PERMISSIONS-Reiter wird geöffnet und zeigt die Rechte an, die auf dem Objekt in diesem Zustand angewendet werden. Das kann bedeuten, dass sich die Rechte des Objekts ändern, wenn es in diesen Zustand übergeht. Das Formular ist ziemlich selbsterklärend. Damit ein anonymer Benutzer das Objekt sehen kann, markieren Sie die Kästchen für VIEW und ANONYMOUS und klicken Sie auf SAVE, wie in Abbildung 8.7 zu sehen ist.

Wenn Sie die Rechte an einem bestimmten Workflow-Zustand ändern, haben Sie ein Problem, das Sie lösen müssen. Diese neuen Workflow-Rechte werden nicht auf den vorhandenen Inhalten in diesem Zustand gesetzt. Diese Inhalte haben weiterhin die alten Workflow-Rechte, und Sie müssen sie aktualisieren.

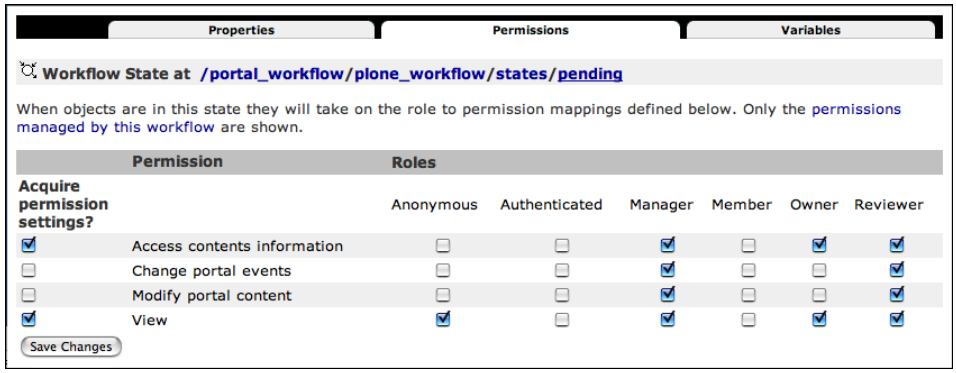


Abbildung 8.7: Seite für Zustandsrechte

Wenn Sie mit Ihren Änderungen fertig sind, gehen Sie in die Workflow-Ausgangsseite und klicken auf UPDATE SECURITY SETTINGS, wie in Abbildung 8.4 gezeigt wird. Es kann eine Weile dauern, diese Aktualisierung durchzuführen, je nachdem, wie viele Objekte davon betroffen sind.

Der VARIABLES-Reiter gestattet Ihnen die Zuweisung eines Wertes an eine Variable, falls das Objekt in diesem Zustand ist. Der Workflow bestimmt für jeden Zustand die Liste der verfügbaren Variablen. Weitere Informationen dazu finden Sie im Abschnitt »Variablen bearbeiten« weiter unten.

Übergänge bearbeiten

Der TRANSITIONS-Reiter listet die Übergänge auf, die in diesem Workflow erfolgen können. Zu Beginn dieses Kapitels habe ich Ihnen gezeigt, wie ein Übergang Änderungen darstellt, die auf dem Objekt erfolgen. Jeder Übergang hat einige Variablen, die auf der Zusammenfassungsseite angezeigt werden. Um einen Übergang hinzuzufügen geben Sie eine ID ein und klicken unten auf der Seite auf ADD.

Wenn Sie nun auf einen Übergang klicken, öffnen Sie die Details zu diesem Übergang, wie sie in Abbildung 8.8 zu sehen sind:

- TITLE: Dies ist der Titel des Übergangs.
- DESCRIPTION: Dies ist die detaillierte Beschreibung des Übergangs.
- DESTINATION STATE: Das ist der Zielzustand dieses Übergangs. Der Ursprungszustand wird bei der Zuweisung des Übergangs an einen Zustand definiert.
- TRIGGER TYPE: Hiermit wird angegeben, wodurch der Übergang ausgelöst wird. *Automatic* bedeutet, dass er sofort ausgelöst wird, wenn dieser Zustand erreicht wird. *Initiated by user action* ist die am häufigsten gewählte Einstellung und bedeutet, dass ein Benutzer den Übergang ausgelöst hat, indem er einen Link angeklickt hat.

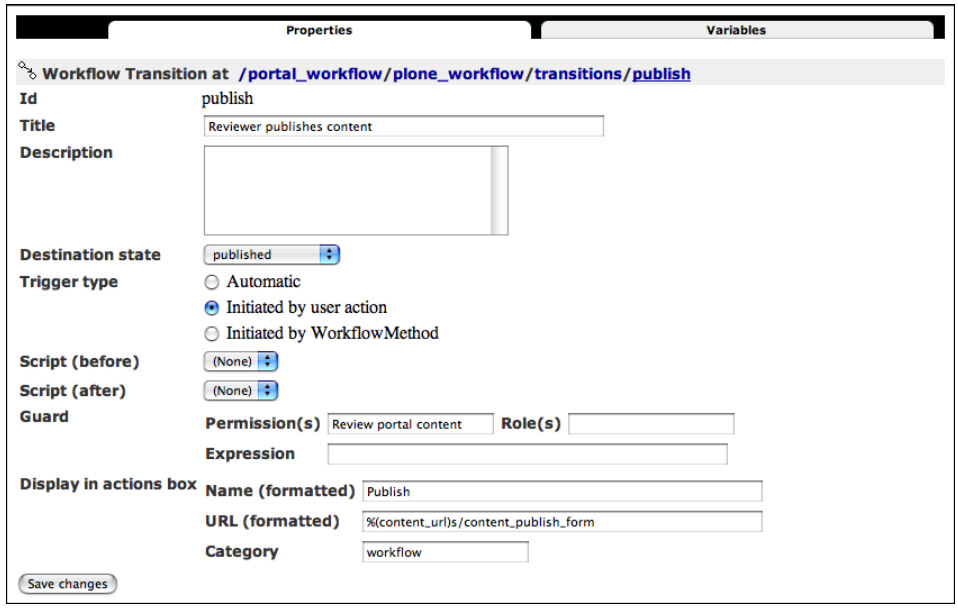


Abbildung 8.8: Seite mit Übergangsdetails

- SCRIPT (BEFORE): Dieses Script wird ausgeführt, bevor der Übergang erfolgt.
- SCRIPT (AFTER): Dieses Script wird ausgeführt, nachdem der Übergang erfolgt ist.
- GUARD: Dies ist der Wächter bei diesem Zustand (wird gleich erklärt).
- DISPLAY IN ACTIONS BOX: Hier wird angegeben, wie dieser Übergang in Plone dargestellt wird. Eine Eingabe an dieser Stelle garantiert, dass der Übergang auch als Aktion eingegeben wird. Sie erhalten diesen Übergang als Aktion, wenn Sie nach Aktionen fragen.

Unter diesen Werten ist der Zielzustand recht interessant. Ich habe zwar schon erwähnt, dass Übergänge normalerweise den Zustand wechseln, aber das muss nicht so sein. Da bei jedem Übergang Scripten ausgeführt werden können und etwas protokolliert werden kann, ist es manchmal nützlich, den Zustand *nicht* zu wechseln. Betrachten Sie als Beispiel hierfür den Abschnitt »Änderungen mit einem Workflow verfolgen« weiter unten in diesem Kapitel. Wenn Ihr Übergang den Zustand wechselt, wählen Sie den neuen Zustand als Zielzustand.

Ein Übergang kann mehrere Startpunkte, aber nur ein Ziel haben. Wenn Sie mehrere Ziele brauchen, müssen Sie mehrere Übergänge anlegen. Sie können auch Scripten angeben, die vor und nach diesem Übergang ausgeführt werden. Zwei häufige Beispiele hierfür sind das Verschieben eines Objekts im Workflow

und das Senden von Benachrichtigungen per E-Mail. Beide Beispiele werden im Abschnitt »Häufige Aufgaben und Beispiele« weiter unten behandelt.

Bevor ein Übergang erfolgt, wird er von einem Sicherheitswächter (Guard) überprüft, um sicherzustellen, dass der Benutzer auch das Recht hat, den Übergang auszuführen. Dieser Wächter besteht aus den folgenden drei Komponenten:

- PERMISSION(S): Dies sind die notwendigen Rechte. Mehrere Rechte sollten durch ein Semikolon (;) voneinander getrennt sein.
- ROLE(S): Dies sind die notwendigen Rollen. Mehrere Rollen sollten durch ein Semikolon (;) voneinander getrennt sein.
- EXPRESSION: Dies ist ein Workflow-Ausdruck. Weitere Informationen dazu finden Sie im Abschnitt »Workflow-Ausdrücke bearbeiten« weiter unten in diesem Kapitel. Für jeden angegebenen Wert muss der Wächter den Wert *logisch wahr* ergeben, bevor weitergemacht werden kann. Falls ein Test bei einem der Werte fehlschlägt, wird der Übergang nicht ausgeführt. Meistens sind bei einem Wächter nur ein oder zwei Werte angegeben.

Variablen bearbeiten

Der Variables-Reiter listet die Variablen auf, die im Workflow erzeugt und geändert werden. Bisher habe ich diese Variablen nicht groß erwähnt, sondern habe mich auf Zustände und Übergänge konzentriert. In diesem Abschnitt geht es nun um Variablen.

Es ist nicht immer möglich, alle in einem Workflow benötigten Informationen in Zuständen und Übergängen zu kapseln. Und ich rate Ihnen auch nicht, das unbedingt zu versuchen. Stattdessen können Sie Variablen verwenden, um einige Workflow-bezogene Informationen zu speichern. Im Kreditkartenbeispiel könnte die Rechnung z.B. auf mehrere Arten beglichen werden, z.B. per Online-Banking, Scheck usw. Sie könnten den Betrag (z.B. 100 Euro) in einer Variablen speichern. Sollte die Rechnung abgelehnt oder verändert werden, würde dieser Betrag aktualisiert. Der wesentliche Punkt bei einer Variablen ist der, dass man etwas hat, das sich zwischen den Zuständen und Übergängen ändern kann.

Klicken Sie nun wieder in der Workflow-Hauptseite auf den VARIABLES-Reiter, um eine Liste aller Variablen zu bekommen. Um eine Variable hinzuzufügen, geben Sie eine Variablen-*ID* ein und klicken unten auf der Seite auf ADD. Um zu bestimmen, in welchem Zustand ein Objekt zu einer bestimmten Zeit ist, speichert DCWorkflow den aktuellen Zustand in einer Variablen im Objekt. Standardmäßig lautet der Name dieser Variable `review_state`.

Hinweis

Sollten Sie diesen Namen ändern müssen, weil es einen Konflikt mit einem anderen Namen gibt, können Sie das unten auf der Seite tun. Dabei geht allerdings der Zustand all Ihrer aktuellen Objekte verloren. Seien Sie also bei einer solchen Änderung vorsichtig.

Alle Workflow-Variablen verfügen über die folgenden Eigenschaften:

- **Description**: Dies ist eine Beschreibung der Variablen.
- **Make available to catalog**: Diese Variablen werden in einer Liste platziert, auf die der Katalog zugreifen kann. Dabei werden kein Index oder Metadaten zum Katalog hinzugefügt. Das müssen Sie weiterhin von Hand machen.
- **Store in workflow**: Hiermit wird bestimmt, ob die Information im Workflow oder im Objekt gespeichert wird.
- **Variable update mode**: Dies bestimmt, wann die Variable aktualisiert wird.
- **Default value**: Das gibt einen Standardwert als String an.
- **Default expression**: Dies ist der Standardwert als Ausdruck. Wenn er vorhanden ist, wird er an Stelle des String-Wertes verwendet (weitere Informationen finden Sie im Abschnitt »Workflow-Ausdrücke bearbeiten« weiter unten in diesem Kapitel).
- **Info. guard**: Das sind Sicherheitseinstellungen beim Zugriff auf diese Variable. Diese Wächtereinstellungen ähneln jenen eines Übergangs. Hierbei wird der Wächter allerdings dann ausgeführt, wenn auf die Variable zugegriffen wird.

Worklists bearbeiten

Der WORKLISTS-Reiter bietet einen Zugriff auf alle Worklists, die in diesem Workflow zugewiesen sind. Eine *Worklist* ist eine Methode, den Workflow nach Informationen über die Anzahl von Objekten in diesem Workflow zu fragen. Ich möchte z.B. gern den Workflow einfach nach allen ausstehenden Kreditkartenrechnungen für mich fragen.

Um eine Worklist hinzuzufügen, geben Sie eine ID ein und klicken auf ADD. Alle Worklists verfügen über folgende Eigenschaften:

- **Description**: Dies ist eine Beschreibung der Worklist.
- **Cataloged variable matches**: Dies ist der Wert, auf den die Workflow-Variable eines Objekts passen muss, damit es zu dieser Worklist hinzugefügt wird. Die passende Variable ist die Workflow-Zustandsvariable, die in der Variablenliste angegeben ist (der Standardname dieser Variablen lautet review_state).

- **Display in actions box**: Diese Information soll in der Benutzerschnittstelle angezeigt werden. Die Eingabe eines Wertes an dieser Stelle stellt auch sicher, dass der Übergang als Aktion eingetragen wird. Sie erhalten diesen Übergang als Aktion, wenn Sie nach den Aktionen fragen.
- **Guard**: Dies ist ein Wächter für den Zugriff auf diese Worklist.

Kommen wir wieder zum Kreditkartenbeispiel zurück: Wenn ich gerne wüsste, welche Kreditkartenrechnungen ich alle überprüfen muss, könnte ich diese Information in einer Worklist angeben. Zuerst würde die Variable `review_state` den aktuellen Zustand jedes Elements enthalten. Alle zu überprüfenden Kreditkartenrechnungen wären im Zustand *review* (überprüfen). Zweitens würde ich eine Worklist namens `review_queue` hinzufügen, und der Variablenwert wäre `pending` (offen). Nun würde ich die Worklist nach allen Elementen in `review_queue` fragen.

Obwohl Worklists eine bequeme Art sind, diese Information zu speichern, benutzt Plone selbst keine Worklists. Stattdessen benutzt Plone direkt ZCatalog, um Objekte abzufragen, die sich in einem Workflow befinden. Da die DCWorkflow-Worklist das Katalog-Werkzeug verwendet, ist das Ergebnis das gleiche.

Scripten bearbeiten

Der SCRIPTS-Reiter listet die Scripten auf, die in diesem Workflow verfügbar sind. Diese Liste ist eigentlich ein normaler Ordner im ZMI, und Sie können darin fast alles hinzufügen. Der Hauptgrund dafür ist vermutlich der, ein Script hinzuzufügen, um eine kompliziertere Behandlung von Übergängen zu bewerkstelligen, d.h., Sie sollten an dieser Stelle nur Script(Python)-Objekte hinzufügen.

Um ein Script unter dem Scripts-Reiter hinzuzufügen, wählen Sie im ADD-Dropdown-Menü die Option SCRIPT (PYTHON) und geben dem Script eine ID. Dem Script wird genau ein Objekt übergeben, nämlich das Ausdrucksobjekt des Basis-Workflows. Weitere Informationen darüber finden Sie im Abschnitt »Workflow-Scripten bearbeiten« weiter unten in diesem Kapitel. Wenn Sie z.B. auf das eigentliche Objekt im Workflow zugreifen müssen, können Sie ein Python-Script wie folgt benutzen:

```
##parameters=state_change
obj = state_change.object
```

Was in diesem Script passiert, entscheidet dessen Entwickler. Sie können hier fast alles machen. Sie können Ereignisse auslösen, und Sie können auf andere Workflows und Übergänge zugreifen. Einige Beispielscripts finden Sie in den Abschnitten »Benachrichtigungen per E-Mail versenden« und »Objekte verschieben« im weiteren Verlauf dieses Kapitels. Wenn das Script ausgeführt wird, wird es unter dem Benutzer ausgeführt, der den Übergang eingeleitet hat. Sie könnten dem Script eine Proxy-Rolle zuweisen, wenn es unter einem anderen Benutzer ausgeführt werden muss. Was die Übergänge angeht, so können Sie dieses Script

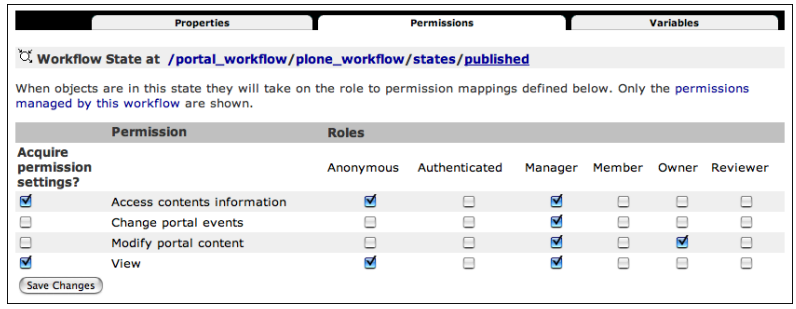
in den Einstellungen zu *script (after)* und *script (before)* an beliebig viele Übergänge zuweisen. Sie können das Script entweder vor oder nach einem Übergang ausführen.

Rechte bearbeiten

Der PERMISSIONS-Reiter listet die von diesem Workflow verwalteten Rechte auf. Diese Rechte haben Sie schon bei der Behandlung von Zuständen gesehen. Unter diesem Reiter setzen Sie die Liste der Rechte, die in diesen Zuständen verwaltet werden. Ein neues Recht fügen Sie hinzu, indem Sie es im Dropdown-Menü auswählen und auf ADD klicken.

Exkurs: Wie können Sie ein veröffentlichtes Dokument bearbeiten?

Nun, im Default-Workflow können Sie ein schon veröffentlichtes Dokument nicht bearbeiten, es sei denn, Sie haben die Rolle eines Managers. Wenn Sie dem Besitzer des Dokuments erlauben, es zu bearbeiten, sollten Sie es wirklich nochmal prüfen. Allerdings scheint das eine häufige Frage zu sein, und man kann es ganz schnell so einrichten, dass das geht. Klicken Sie im ZMI auf PORTAL_WORKFLOW, und wählen Sie den CONTENTS-Reiter. Dann klicken Sie auf PLONE_WORKFLOW und wählen den STATES-Reiter. Und schließlich klicken Sie auf PUBLISHED und wählen den PERMISSIONS-Reiter. Markieren Sie das Kästchen, das einem Besitzer erlaubt, Portal-Inhalte zu ändern.



Klicken Sie auf SAVE CHANGES, um Ihre Rechte zu speichern. Da Sie die Sicherheitseinstellungen aktualisiert haben, müssen Sie auf PORTAL_WORKFLOW klicken, den CONTENTS-Reiter wählen und auf UPDATE SECURITY SETTINGS klicken. Dadurch werden alle Objekte in Ihrer Site aktualisiert, und es wird garantiert, dass Ihre Rechte auf den vorhandenen Objekten angewendet wurden. Ab jetzt können Dokumente von ihren Besitzern bearbeitet werden, während sie im Zustand *Veröffentlicht* sind.

Workflow-Scripten bearbeiten

Scripten bieten dem Entwickler die Möglichkeit, bei der Ausführung eines Übergangs eine zusätzliche Anwendungslogik auszuführen. Das kann fast alles sein, was Sie tun möchten. Sie könnten z.B prüfen, ob gewisse Bedingungen erfüllt sind (z.B. ob die Rechtschreibung beim Dokument geprüft wurde) oder ob spezielle Aktionen ausgeführt wurden. Wenn ein Objekt seinen Zustand wechselt, wird das Script aufgerufen.

Wenn ein Script aufgerufen wird, wird ihm ein zusätzlicher Parameter übergeben. Dieser zusätzliche Parameter ermöglicht den Zugriff auf alle Arten von übergangsbezogenen Elementen und Attributen. Dieser Parameter wird auch `state_change`-Parameter genannt und hat folgende Attribute:

- **status**: Dies ist der Status des Workflows.
- **object**: Dies ist das Objekt, das im Workflow einen Zustandswechsel durchmacht.
- **workflow**: Dies ist das aktuelle Workflow-Objekt zu dem betroffenen Objekt.
- **transition**: Dies ist das aktuell ausgeführte Übergangsobjekt.
- **old_state**: Dies ist der Ursprungszustand des Objekts.
- **new_state**: Dies ist der Zielzustand des Objekts.
- **kwargs**: Dies sind die Schlüsselargumente, die an die Methode `doActionFor` übergeben werden.
- **getHistory**: Dies ist eine Methode, die keine Parameter hat und eine Kopie der Workflow-Historie des Objekts zurückgibt.
- **getPortal**: Dies ist eine Methode, die keine Parameter hat und das Plone-Wurzelobjekt zurückgibt.
- **ObjectDeleted(ordner)**: Dies sagt dem Workflow, dass das Objekt, dessen Zustand wechselt, gelöscht wurde. Diese Methode erwartet das Objekt, zu dem der Benutzer zurückkehren soll. Übergeben Sie der Ausnahme den Ordner, zu dem der Benutzer umgeleitet werden soll (siehe den Abschnitt »Objekte verschieben« weiter unten in diesem Kapitel).
- **ObjectMoved(neuesObjekt, neuesObjekt)**: Dies sagt dem Workflow, dass das Objekt, dessen Zustand wechselt, verschoben wurde. Übergeben Sie der Ausnahme den Ordner, zu dem der Benutzer umgeleitet werden soll (siehe den Abschnitt »Objekte verschieben« weiter unten in diesem Kapitel).
- **getDateTime**: Dies ist eine Methode, die keine Parameter hat und das zum Übergang gehörende `DateTime`-Objekt zurückgibt.

Um z.B. herauszufinden, zu welchem Zielzustand ein Übergang erfolgt und wann, können Sie das folgende Script(Python)-Objekt verwenden, das Ihnen genau diese Information liefert. Dieses Script schreibt die Information über den Übergang in die Protokolldatei:

```
##parameters=state_change
st = 'From %s to %s on %s' % (
    state_change.old_state,
    state_change.new_state,
    state_change.getDateTime())
context.plone_log(st)
```

Tipp

Wenn Sie ein Script(Python)-Objekt schreiben, müssen Sie eventuell etwas in die Protokolldatei ausgeben, um sich die Fehlersuche leichter zu machen. Ein Script namens `plone_log` tut genau das. Es erwartet einen String, den es an die Protokollfunktionen von Plone übergibt. Daher sind Aufrufe von `context.plone_log` bei der Fehlersuche sehr hilfreich.

Bei der Zuweisung eines Scripts an einen Übergang haben Sie zwei Möglichkeiten: `before` und `after`. Wie diese Namen schon besagen, wird ein Script, das an `before` zugewiesen wird, ausgeführt, bevor der Übergang erfolgt. Das eignet sich für Scripten, die prüfen, ob etwas noch vor dem Übergang passieren soll, die also z.B. testen, ob ein weiteres abhängiges Objekt oder eine Seite hochgeladen wurde oder ob etwas frei von Tippfehlern ist. Das an `after` zugewiesene Script wird ausgeführt, nachdem der Übergang beendet ist. Sollte aber eine irgendwann ausgelöste, nicht abgefangene Ausnahme in einem Script auftreten, dann führt das dazu, dass der Übergang fehlschlägt, das Objekt in seinem ursprünglichen Zustand bleibt und dem Benutzer die Ausnahme angezeigt wird.

Workflow-Ausdrücke bearbeiten

In diesem Kapitel haben Sie durchgehend Werte gesehen, die als Workflow-Ausdrücke ausgedrückt werden können. Der Wert beispielsweise, der einer Variablen zugewiesen wird, ist das Ergebnis eines Workflow-Ausdrucks. Dieser Ausdruck ist nichts Ungewöhnliches, sondern lediglich ein TAL-Ausdruck (Template Attribute Language) mit einigen anderen verfügbaren Variablen. TAL-Ausdrücke haben Sie bereits in Kapitel 5 kennen gelernt, d.h., Sie sollten mit diesen Ausdrücken und ihren Optionen wie Python-, String- und Pfadausdrücken vertraut sein.

Anders als bei den normalen TAL-Ausdrücken werden einige zusätzliche Parameter mit Bezug auf den konkreten Workflow über den Namespace übergeben. Der Namespace eines Workflow-Ausdrucks enthält Folgendes:

- **here**: Dies ist das Objekt, das im Workflow einen Zustandswechsel erfährt.
- **container**: Dies ist der Container des Objekts, das im Workflow den Zustandswechsel erfährt.
- **state_change**: Dies ist das Zustandswechselobjekt, das im Abschnitt »Workflow-Scripten bearbeiten« erwähnt wurde.
- **transition**: Dies ist der gerade ausgeführte Übergang; identisch mit `state_change.transition`.
- **status**: Dies ist der ursprüngliche Zustand, identisch mit `state_change.old_state`.
- **workflow**: Dies ist der Workflow für dieses Objekt.
- **scripts**: Dies sind die in diesem Workflow verfügbaren Scripten.
- **user**: Dies ist der Benutzer, der den Übergang ausführt.

8.4 Häufige Aufgaben und Beispiele

Ich werde Ihnen nun einige Aufgaben vorstellen, die Sie leicht mit einem Workflow bewerkstelligen können. Wenn ein Benutzer einen Übergang im Workflow bewirkt, wird dieser Übergang mit den Rechten dieses konkreten Benutzers ausgeführt. In vielen dieser Beispiele verfügt ein normaler Benutzer möglicherweise nicht über die notwendigen Rechte, um die Aufgabe auszuführen. Mitglieder haben z.B. normalerweise kein Recht, auf die Liste der Mitglieder zuzugreifen, sofern ihnen dieses Recht nicht explizit gewährt worden ist.

Um dieses Rechteproblem zu lösen, wurde da, wo es vermerkt ist, einigen der folgenden Script (Python)-Objekte eine leicht andere Rolle gegeben. Um auf einem Script eine Proxy-Rolle zu setzen, greifen Sie auf den PROXY-Reiter eines Objekts zu und wählen dann den Benutzer, der das Script ausführen soll, wie in Abbildung 8.9 zu sehen ist.

Natürlich würden Sie dabei dafür sorgen, dass Ihre Scripten mit den minimal notwendigen Rollen ausgeführt werden, je nachdem, was Ihr Script genau macht.

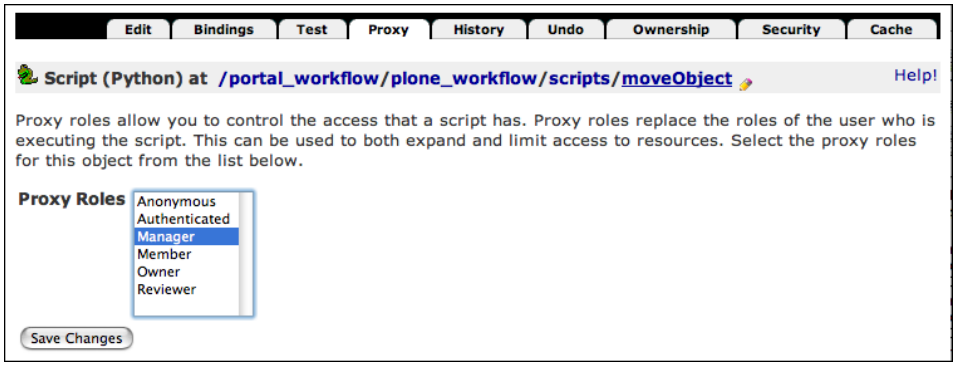


Abbildung 8.9: Proxy-Einstellungen eines Scripts setzen

8.4.1 Einführung in Workflow-Ausdrücke

Im Folgenden sehen Sie einige nützliche Beispiele von Workflow-Ausdrücken, die an verschiedenen Stellen verwendet werden können.

Benutzen Sie Folgendes, um die Kommentare eines Übergangs bzw. einen leeren String zu erhalten:

```
python:state_change.kwargs.get('comment', '')
```

Um den Titel eines Ordners zu erhalten, in dem das Objekt enthalten ist, benutzen Sie Folgendes:

```
container/Title
```

Um zu testen, ob der alte Zustand gleich *Offen* (*pending*) ist, schreiben Sie Folgendes:

```
python: state_change.old_state == 'pending'
```

Um an den Benutzer zu kommen, der den Übergang ausführt, benutzen Sie Folgendes:

```
user/getId
```

Wenn Sie also mitverfolgen möchten, welcher Benutzer zuletzt den Zustand eines Objekts verändert hat, könnten Sie eine Variable namens `last_user` zum Workflow hinzufügen. Das machen Sie, indem Sie zum Workflow gehen, dann auf den VARIABLES-Reiter klicken und anschließend die Variable `last_user` hinzufügen. Wenn Sie die Variable `Default expr` auf `user/getId` setzen, würde dieser Wert jedes Mal für Sie gespeichert, wenn das Objekt seinen Zustand ändert.

8.4.2 Änderungen mit einem Workflow verfolgen

Ein Kunde von mir wollte für eine spezielle Anwendung mitverfolgen, wann immer und aus welchem Grund ein Element bearbeitet wurde, so dass bei einer späteren Prüfung des Elements zu jeder Änderung ein Kommentar vorhanden wäre. Mit Hilfe eines Workflows war das recht einfach zu realisieren.

In diesem Fall bestand der Workflow aus nur einem Zustand, aber das würde sogar bei fast allen Workflows funktionieren. Bei diesem speziellen Workflow wurde ein Übergang namens `edit` hinzugefügt, der nicht wirklich den Objektzustand verändert. Der Zielzustand dieses Übergangs war `(Remain in state)`, d.h., es fand kein Wechsel statt.

Wenn ein Objekt bearbeitet wird, wird eine Methode aufgerufen, die den Übergang ausführt. Wenn z.B. ein Dokument bearbeitet wird, lautet die aufgerufene Methode `document_edit.cpy`. Dieses Script finden Sie, wenn Sie nacheinander `portal_skins`, `plone_form_scripts` und `document_edit` anklicken. Sie müssen lediglich eine Zeile vor der letzten Zeile dieses Scripts hinzufügen:

```
context.portal_workflow.doActionFor(new_context,
 'edit', comment='')
```

Die Methode `doActionFor` in `portal_workflow` führt den gegebenen Übergang (in diesem Fall `edit`) auf dem übergebenen Objekt durch (in diesem Fall `context`). Jedes Mal, wenn das Objekt bearbeitet wird, wird dieser `edit`-Übergang ausgelöst. Das führt dann dazu, dass eine Zeile zur Kommentarliste hinzugefügt wird, in der neben einem Kommentar steht, wer das Objekt bearbeitet hat und wann es hinzugefügt wurde.

Eigentlich gibt es keinen Kommentar, wenn ein Objekt bearbeitet wird. Das heißt, um es ganz richtig zu machen, müssten Sie das Bearbeiten-Template des Dokuments so modifizieren, dass es ein Kommentarfeld enthält. Dann könnten Sie mit dem `content_status_history`-Formular auf diese Kommentarliste zugreifen, wo unten die Liste der Änderungen angezeigt wird. Dieses Formular erreichen Sie, indem Sie im Workflow-Menü auf ERWEITERT klicken.

8.4.3 Objekte verschieben

Eine nützliche Fähigkeit ist die, ein Objekt im Workflow zu verschieben. Sie könnten z.B. immer dann, wenn Sie eine Pressemitteilung veröffentlichen, alle Pressemitteilungen in einen Ordner namens `Press Release` verschieben. Inhalte könnten irgendwo erstellt und bearbeitet werden, um dann bei ihrer Veröffentlichung in diesen Ordner verschoben zu werden. Das Beispielscript in Listing 8.1 verschiebt das vom Workflow erfasste Objekt in den `Members`-Ordner. Um dieses Script hinzuzufügen, gehen Sie im ZMI zum Workflow-Werkzeug und klicken

auf den SCRIPTS-Reiter. Dann wählen Sie SCRIPT (PYTHON) im Dropdown-Menü aus. Nun geben Sie dem neuen Objekt den Namen **moveObject** und geben dann den Code aus Listing 8.1 in diesem Script ein.

Listing 8.1: Ein Objekt verschieben

```
##parameters=state_change
# get the object and its ID
obj = state_change.object
id = obj.getId()

# get the src folder and the destination folder
dstFldr = context.portal_url.Members
srcFldr = obj.aq_parent

# perform the move
objs = srcFldr.manage_cutObjects([id,])
dstFldr.manage_pasteObjects(objs)

# get the new object
new_obj = dstFldr[id]

# pass new_obj to the error, *twice*
raise state_change.ObjectMoved(new_obj, new_obj)
```

Sie müssen noch ein paar andere Dinge machen. Zuerst weisen Sie dieses Script einem Übergang zu. Ich benutze ein solches Script normalerweise im Publish-Übergang. Dazu gehen Sie zum Übergang und weisen `moveObject` den Wert `script (after)` zu.

Zweitens gibt es noch ein kleines Problem: Dieses Script verschiebt Objekte in den `Members`-Ordner. Wahrscheinlich schwebt Ihnen ein besserer Ort dafür vor. Um ein solches Verschieben vorzunehmen, muss ein Benutzer die geeigneten Rechte haben, um Objekte zwischen diesen Ordnern zu bewegen. Normalerweise kann nur ein Manager Objekte in den `Members`-Ordner verschieben. Daher müssen Sie dem Script die Proxy-Rolle eines Managers geben. Das tun Sie, indem Sie erst auf SCRIPTS und dann auf MOVEOBJECT klicken und anschließend den PROXY-Reiter wählen. Weisen Sie diesem Script dann die Manager-Rolle zu. Weitere Informationen über Sicherheit und lokale Rollen finden Sie in Kapitel 9.

Wenn Sie sich den Code anschauen, sehen Sie, dass das Script zuerst das Objekt und seine Objekt-ID aus dem Namespace des Übergangs holt. Dann holt es sich den Ursprungs- und Zielordner. Danach benutzt es die `ObjectManager`-API von Zope zum Kopieren und Einfügen. Natürlich könnten Sie diese Ordner auch im Programm selbst bestimmen, beispielsweise abhängig vom Benutzer, der den Übergang durchführt, oder vom Typ des verschobenen Inhalts. Schließlich haben Sie das Objekt und übergeben es an die Ausnahme `ObjectMoved`.

Die Ausnahme (Exception) `ObjectMoved` ist eine spezielle Ausnahme in DCWorkflow. Indem das neue Objekt zweimal als Parameter an die Ausnahme übergeben wird, wird dieses neue Objekt an das Plone-Frontend weitergegeben. Hier muss man aufpassen, denn wenn der Benutzer als Folge der Änderung zum Objekt geleitet wird, ist es das Objekt an seinem neuen Zielort und nicht das Objekt am Ursprungsort. Natürlich könnten Sie auch eine Funktion schreiben, die das Objekt wieder zurückbewegt, nachdem es abgelehnt wurde – vielleicht in den Startordner des Benutzers.

Ein weiterer spezieller und eher ungewöhnlicher Fall ist der, ein Objekt im Workflow zu löschen. Normalerweise ist das Löschen eines Objekts eine Aktion des umgebenden Objekts, daher ist es im Workflow ungewöhnlich. Für diese Aufgabe können Sie die Ausnahme `ObjectDeleted` auslösen. Listing 8.2 zeigt das Script, mit dem man ein Objekt löschen kann.

Listing 8.2: Ein Objekt löschen

```
##parameters=state_change

# get the object
obj = state_change.object
id = obj.getId()

# get the parent folder, delete the object
srcFldr = obj.aq_parent
srcFldr.manage_delObjects([id,])

# raise the object deleted method and pass
# the folder you want to return to
raise state_change.ObjectDeleted(srcFldr)
```

Dieses Script könnten Sie `deleteObject` nennen und Objekte damit erfolgreich im Workflow löschen. Noch einmal: Durch das Auslösen der Ausnahme weiß Plone, was es zu tun hat. In diesem Fall bringt es den Benutzer zu dem Ordner, der das Objekt enthält.

8.4.4 Benachrichtigungen per E-Mail versenden

Wenn Sie eine Website haben, die von ihren Benutzern nicht so oft besucht wird, dann ist es sinnlos, darauf Informationen darüber anzuzeigen, welche Inhalte geprüft werden müssen. Sie können den Workflow aber zu einem einfachen Benachrichtigungssystem machen, indem Sie damit E-Mails an die Benutzer verschicken. Der Nachrichtenkanal über E-Mail ist nur ein Beispiel. Es könnte auch einer mit Instant Messages, SMS-Mitteilungen per Telefon usw. sein. Ich über lasse es Ihnen, sich weitere Möglichkeiten auszudenken.

In diesem Beispiel senden Sie E-Mails über das `MailHost`-Objekt auf dem Server an alle Benutzer mit der Rolle eines Redakteurs im System. Mit diesen E-Mails teilen Sie ihnen mit, dass neue Elemente für die Überprüfung bereitstehen. Dieses Script ist etwas komplizierter als die anderen, die ich Ihnen bisher gezeigt habe, da es mehrere Schritte durchmacht: Definieren der Variablen, Finden der Kontonamen aller Redakteure, Finden und Senden einer E-Mail. Sie sehen das Script in Listing 8.3.

Listing 8.3: Eine Benachrichtigung per E-Mail versenden

```
##parameters=state_change
# the objects we need
object = state_change.object
mship = context.portal_membership
mhost = context.MailHost
administratorEmailAddress = context.email_from_address

# the message format, %s will be filled in from data
message = """
From: %s
To: %s
Subject: New item submitted for approval - %s

%s

URL: %s
"""
```

Hiermit werden die Nachricht und die benötigten Objekte aufgesetzt. Neben dem Objekt, das einen Zustandswechsel erfährt, benötigen Sie auch eine Referenz auf das Mitglieder-Werkzeug `portal_membership` und den SMTP-Server (Simple Mail Transfer Protocol) über `MailHost`. Die Nachricht lässt sich leicht so konfigurieren, dass Sie eine E-Mail in einem beliebigen Format schicken können.

Dann benutzen Sie die Methode `listMembers` des Objekts `portal_membership`, um eine Liste der Mitglieder zu erhalten. Bei jedem Mitglied können Sie dann nachprüfen, ob in der Liste der Rollen zu diesem Benutzer die Redakteursrolle enthalten ist, indem Sie die Methode `getRoles` aufrufen:

```
for user in mship.listMembers():
    if "Reviewer" in mship.getMemberById(user.id).getRoles():
```

Aufmerksame Leser werden bemerken, dass eine Iteration über alle Mitglieder einer Plone-Site etwas langsam sein könnte, falls Sie Tausende von Benutzern haben sollten. Im nächsten Kapitel werden Sie das Script modifizieren, um eine Liste von Benutzern aus einer speziellen Gruppe zu erhalten.

Eine E-Mail zu verschicken macht nur dann Sinn, wenn Sie die E-Mail-Adresse eines Benutzers haben, d.h., hier prüfen Sie zuerst, ob es eine gültige E-Mail-Adresse gibt. Nun muss die E-Mail nur noch formatiert und verschickt werden. Dazu können Sie die String-Ersetzungsfunktionen von Python benutzen und vier Parameter übergeben, die dem `%s` in der Variable `message` entsprechen, die am Anfang des Scripts gesetzt werden. Nach dieser Ersetzung enthält die Variable `msg` die E-Mail, die Sie verschicken möchten. Um sie abzuschicken, rufen Sie einfach die Methode `send` von `MailHost` auf und übergeben dabei den String mit der E-Mail:

```
if user.email:
    msg = message % (
         administratorEmailAddress,
         user.email,
         object.TitleOrId(),
         object.Description(),
         object.absolute_url()
         )
    mhost.send(msg)
```

Dadurch wird die folgende E-Mail verschickt:

```
From: administrator@agmweb.ca
To: andy@agmweb.ca
Subject: New item submitted for approval - Plone's great

We all know Plone is a great product, but with the newest release
it's gotten even better...

URL: http://agmweb.ca/Members/andym/News_Item_123
```

Anhang B zeigt das vollständige Listing für dieses Script.

8.4.5 PloneCollectorNG verwenden

PloneCollectorNG ist ein Bugtracking-Programm für Plone. Sie werden noch viele weitere solcher Programme finden, aber dieses eine benutze und empfehle ich im Zusammenhang mit Plone. Tatsächlich scheinen Entwickler sehr häufig solche Fehlerverwaltungsprogramme zu schreiben. Eine sehr nette Eigenschaft von Workflows ist, dass Ihre Benutzer damit ganz wesentlich die Art und Weise ändern können, wie eine Anwendung funktioniert. Die Entwicklung von Produkten, die mit DCWorkflow zusammenarbeiten, ermöglicht es, dass Ihre Anwendung flexibel bleibt. Sie finden PloneCollectorNG unter *http://www.zope.org/Members/ajung/PloneCollectorNG*.

Das Produkt fügt bei der Installation eine Reihe von Inhaltstypen hinzu, darunter PloneIssueNG, also einen Fehlerbericht (*issue*). Anstatt genau festzuschreiben, wie der Bericht durch die Datenbank geht, wird ihm ein eigener Workflow zugewiesen. Dieser Workflow enthält die passenden Zustände, Übergänge, Variablen und Worklists.

Sie können zu jedem Zeitpunkt herausfinden, in welchem Zustand ein Objekt ist, indem Sie die Methode `getInfoFor` in `portal_workflow` aufrufen. Diese nützliche Methode erwartet ein Objekt und die Variable, die gesucht werden soll. Im Workflow von PloneCollectorNG lautet diese Variable `state`, und im Plone-Workflow lautet sie `review_state`. Um z.B. den Zustand eines Objekts herauszufinden, benutzen Sie Folgendes:

```
portal_workflow.getInfoFor(obj, "state")
```

Die möglichen Zustände eines Objekts finden Sie heraus, indem Sie das Zustandsobjekt direkt im Workflow wie folgt untersuchen:

```
portal_workflow['pcng_workflow'].states._mapping.keys()
```

Wenn Ihre Benutzer ein einfaches Fehlerberichtssystem haben möchten, dann können Sie hiermit recht einfach diesen Workflow über das Web entsprechend anpassen (falls bei der Entwicklung der Anwendung die Workflow-Werkzeuge berücksichtigt wurden). Vergleichen Sie das einmal mit Bugzilla, einem anderen populären Bugtracking-System, wo zum Ändern eines Zustands oder eines Übergangs viele Stunden an Arbeitszeit eines Perl-Programmierers nötig sind, um alle festkodierten Referenzen auf Zustände von Fehlern herauszufinden.

8.4.6 Workflows verteilen und schreiben

Wenn Sie einen tollen Workflow für Ihre Anwendung haben, dann stehen Ihnen einige verschiedene Möglichkeiten offen, den Workflow zu schreiben und zu verteilen. Die folgenden Abschnitte stellen einige dieser Möglichkeiten vor und beschließen damit die Diskussion zu Workflows.

Schreiben über das ZMI

Der vielleicht einfachste, aber arbeitsintensivste Weg, einen Workflow zu schreiben, ist der über das ZMI. Auch, wenn das ZMI viele Leute in den Wahnsinn treibt, ist es doch eine einfach Art, die Optionen einzustellen. Leider ist es aber so, dass Sie, nachdem Sie mit dem Schreiben im ZMI angefangen haben, in diesem Paradigma gefangen sind. Mit anderen Worten: Es gibt keinen einfachen Weg, diesen Workflow im Dateisystem zu bearbeiten oder zu verändern. Natürlich habe ich aber weiter oben in diesem Kapitel beschrieben, wie Sie einen Workflow

über das Web bearbeiten können. Eine andere Alternative ist, den Workflow in einem UML-Tool als Zustandsdiagramm zu erstellen und mit ArchGenXML (siehe Kapitel 13) zu generieren.

Um einen Workflow aus dem ZMI zu exportieren, klicken Sie auf PORTAL_WORKFLOW und dann auf den CONTENTS-Reiter. Wählen Sie die gewünschten Workflows aus, indem Sie die Kästchen links im ZMI markieren, und klicken Sie dann auf IMPORT/EXPORT. Im oberen Teil der Exportseite wählen Sie *Download to local machine* und klicken auf EXPORT. Es wird eine Datei mit der Erweiterung `.zexp` erzeugt, die gespeichert und weiterverteilt werden kann. Bei der Wahl von *XML Format* hat die Datei das Format XML (Extensible Markup Language) mit der Erweiterung `.xml`.

Wenn Sie einen Workflow mit der Erweiterung `.zexp` oder `.xml` erhalten, können Sie diesen Workflow sehr einfach in Ihr Plone importieren. Kopieren Sie die Datei im Dateisystem ins Import-Verzeichnis von Zope. Das kann das Verzeichnis von der Wurzelinstanz oder von Zope sein.

Klicken Sie dann auf PORTAL_WORKFLOW, wählen Sie den CONTENTS-Reiter, und klicken Sie auf IMPORT/EXPORT. Im unteren Teil der Seite werden Sie ein kleines Formular sehen, in dem man einen Import-Dateinamen angeben kann. Geben Sie hier den Dateinamen ein, und lassen Sie die Option *Take ownership of imported objects* markiert. Klicken Sie auf den IMPORT-Button, um den Workflow zu importieren. Nun wird der Workflow importiert und bekommt den Namen, der beim Export angegeben wurde.

Workflows in Python schreiben

Der vermutlich beliebteste Weg, Workflows zu schreiben, ist der, es direkt in Python zu tun, weil man dies vollständig in Python machen und leicht verteilen kann. Erstellen Sie zuerst ein Python-Modul im Dateisystem. Ganz oben in der Datei importieren Sie die passenden Werkzeuge wie folgt:

```
from Products.CMFCore.WorkflowTool import addWorkflowFactory
from Products.DCWorkflow.DCWorkflow import DCWorkflowDefinition
```

Als Zweites erstellen Sie eine Funktion, die den Workflow erzeugt. Anhang A listet das API zum Schreiben eines Workflows etwas detaillierter auf. Aber Sie könnten auch mogeln und sich all die tollen Beispiele anschauen, die im PloneWorkflow-Projekt der Collective-Sammlung verfügbar sind *(http://sf.net/projects/collective*) oder sogar die, die in Plone enthalten sind. Beispiel:

```
def sample(id):
    """ Sample workflow """
    ob = DCWorkflowDefinition(id)
```

```
    ob.states.addState('private')
    ob.states.addState('public')
    # add transitions
    return ob
```

Und schließlich registrieren Sie den Workflow wie folgt im System:

```
addWorkflowFactory(sample,
                   id='sample_workflow',
                   title='Sample workflow')
```

Dieses Script muss nun Teil einer Produkt-Installation werden. Kapitel 12 behandelt das Schreiben und die Installation von Produkten.

Nun gibt es dafür natürlich auch eine Abkürzung namens DCWorkflowDump. Dabei wird für Sie der Code aus dem ZMI genommen und in ein Python-Modul geschrieben. Den Quellcode für DCWorkflowDump finden Sie im Collective-Projekt unter *http://sf.net/projects/collective*, aber Sie finden auch eine Zip-Datei des Codes auf der Website zum Plone-Buch unter *http://plone-book.agmweb.ca*.

Um DCWorkflowDump zu installieren, packen Sie die Datei aus und kopieren das Verzeichnis namens `DCWorkflowDump` in das `Products`-Verzeichnis Ihrer Plone-Installation. Dass Sie im richtigen Verzeichnis sind, merken Sie, wenn das `Products`-Verzeichnis auch das `DCWorkflow`-Verzeichnis und andere Dinge enthält. Dann starten Sie Ihre Plone-Instanz neu.

Nachdem Sie Plone neu gestartet haben, gehen Sie im ZMI zu dem bestimmten Workflow, wo Sie einen neuen Reiter namens DUMP bemerken werden. Klicken Sie auf diese Seite, um zu diesem Dump-Schirm zu gelangen, und klicken Sie dann auf DUMP IT!, um den Workflow auf den Bildschirm zu schreiben. Dabei wird Ihr Workflow für Sie in Python formatiert. Speichern Sie die Datei in Ihr Produkt, und Sie haben eine Python-Datei, die Sie manipulieren können. Dieses Produkt ist ein tolles Werkzeug, weil es Ihnen erlaubt, den Workflow im ZMI zu erstellen und dann mit Hilfe von Python zu verteilen und zu ändern.

9 Sicherheit und Benutzer einstellen

Plone verfügt über ein mächtiges und feinmaschiges Sicherheitsmodell. Es enthält eine Unmenge an Optionen für die Sicherheit auf allen Ebenen, d.h., jedes Objekt kann eigene Sicherheitseinstellungen für einen Benutzer, eine Rolle, eine Gruppe usw. haben.

Um dieses Kapitel in einen größeren Zusammenhang zu setzen, möchte ich Sie mit dem folgenden interessanten Zitat bekannt machen:

> *»Sicherheit ist schwer zu erreichen.«*
> *Jim Fulton , Chefentwickler von Zope*

Das Sicherheitsmodell von Plone ist so mächtig und vielschichtig, dass es sehr schwierig sein kann, es zu verwalten und Fehler darin zu finden. Vermutlich sind aber die richtigen Sicherheitseinstellungen das Wichtigste an einer Plone-Site. Eine Sicherheitslücke auf Ihrer Site ist daher vielleicht der schlimmste Patzer, den Sie machen können. Aus diesem Grund behandle ich die Sicherheit in Plone sehr umfassend.

In diesem Kapitel behandle ich zuerst die gesamte Terminologie und die wichtigsten Schnittstellen, mit denen es Ihre Benutzer zu tun haben werden. Dann zeige ich Ihnen, wie Sie Benutzer und Gruppen über die Plone-Schnittstelle hinzufügen und bearbeiten können. Anschließend gehe ich die wichtigsten Werkzeuge und APIs (Application Programming Interfaces) zur Verwaltung von Benutzer- und Sicherheitseinstellungen durch. Dann behandle ich, wie man Python-Werkzeuge verwendet, um per Script Änderungen an Benutzern und ihren Eigenschaften vorzunehmen. Und am Ende behandle ich die Sicherheit auf Servern und gehe ausführlicher auf die Benutzerauthentifikation ein. Dabei kommt auch ein detailliertes Beispiel dafür vor, wie man Benutzer von einem LDAP-Server (Lightweight Directory Access Protocol) einbindet.

9.1 Benutzer verwalten

Eine der häufigsten Aufgaben für den Administrator einer Plone-Site besteht in der Verwaltung der Mitglieder der Site. Zur Administration zählt dabei normalerweise die Wiedergewinnung von Passwörtern und Änderung von Mitgliedereinstellungen. Über das Web können Sie ziemlich viele Aufgaben erledigen, aber der beste Freund eines jeden Administrators ist natürlich eine Scriptsprache wie Python, um Änderungen en masse durchzuführen. Wenn Sie es mit sehr vielen Benutzern zu tun haben, wird der Abschnitt »Scripten von Benutzern« weiter unten in diesem Kapitel von besonderem Interesse für Sie sein.

9.1.1 Benutzer, Rollen und Gruppen

Zu den wichtigsten Konzepten von Plone gehören Benutzer, Rollen und Gruppen. Bevor ich Ihnen zeige, wie Sie diese bearbeiten können, möchte ich etwas detaillierter darstellen, worum es dabei jeweils geht.

Benutzer

Eine Person, die eine Plone-Site benutzt, wird auch als *Benutzer* bezeichnet. Der Benutzer wird von Plone authentifiziert oder auch nicht, wobei nicht authentifizierte Benutzer auch *anonyme Benutzer* genannt werden. Authentifizierte Benutzer sind mit einem vorhandenen Benutzerkonto angemeldet. Sollten sie kein Konto haben, dann können sie sich normalerweise selbst eines erstellen.

Anonyme Benutzer stellen insofern die *niedrigste* Stufe von Benutzern dar, als für sie die meisten Einschränkungen gelten. Sobald sich Benutzer anmelden, erhalten sie die Rollen zugewiesen, die ihnen ihr Konto zuordnet. Benutzer werden mit einem kurzen Bezeichner identifiziert, z.B. `andym`. Standardmäßig werden in Plone keine Benutzer für Sie erstellt, bis auf den einen, der vom Installationsprogramm in Zope hinzugefügt wird, damit Sie einen Administratorzugriff haben. Dessen Name hängt davon ab, was Sie im Installationsprogramm angegeben haben. Meistens ist das `admin`.

Rollen

Zu einer Plone-Site gehört eine Reihe von *Rollen*, womit eine logische Kategorisierung von Benutzern gemeint ist. Anstatt die Rechte aller Benutzer individuell einzustellen, werden die Rechte an jede Rolle einzeln zugewiesen. Jedem Benutzer können keine oder mehrere Rollen zugewiesen werden. Ein Benutzer kann z.B. ein Mitglied und ein Manager sein. Jede Rolle wird mit einem einfachen eindeutigen Namen, z.B. `Member`, bezeichnet.

Eine Plone-Site enthält fünf vordefinierte Rollen, die in zwei Gruppen unterteilt sind: zuweisbare und nicht zuweisbare Rollen. Zuweisbare Rollen können Sie an

Benutzer zuweisen, die diese Rollen ab dem Zeitpunkt ihrer Anmeldung erhalten. Nicht zuweisbare Rollen sind solche, die bereits in einer Plone-Site vorhanden sind und nicht einem bestimmten Benutzer zugewiesen werden können. Die Rolle eines anonymen Benutzers weisen Sie einem Benutzer beispielsweise nicht zu.

Es gibt folgende nicht zuweisbare Rollen:

- **Anonymous**: Dies ist ein Benutzer, der nicht in der Site angemeldet ist. Das könnte ein Benutzer sein, der kein Konto hat, oder einer, der noch nicht angemeldet ist.
- **Authenticated**: Diese Rolle haben alle in der Site angemeldeten Benutzer, egal, welche Rolle sie sonst noch haben. Per definitionem ist ein Benutzer entweder anonym oder authentifiziert, aber nicht beides gleichzeitig. Da ein authentifizierter Benutzer sehr wenige Abstufungsmöglichkeiten bietet, wird bei den meisten Anwendungen vom Gebrauch dieser Rolle abgeraten.

Folgende zuweisbare Rollen gibt es:

- **Besitzer (Owner)**: Diese spezielle Rolle erhalten Benutzer, wenn sie ein Objekt erzeugen. Sie gilt für einen Benutzer nur bei diesem Objekt, und die entsprechende Information wird auf dem Objekt selbst gespeichert. Normalerweise weisen Sie diese Rolle nicht explizit zu, sondern Plone macht das für Sie.
- **Mitglieder (Members)**: Dies ist die Standardrolle von Benutzern, die sich auf Ihrer Site registriert haben. Alle, die sich mit dem MITGLIED WERDEN-Button über die Plone-Schnittstelle registrieren, haben diese Rolle.
- **Redakteure (Reviewer)**: Dies ist ein Benutzer mit mehr Rechten als ein Mitglied, aber weniger Rechten als ein Manager. Redakteure sind Benutzer, die von anderen Mitgliedern eingegebene Inhalte bearbeiten oder prüfen können. Sie können aber nicht die Konfiguration der Site oder ein Benutzerkonto verändern.
- **Manager**: Manager dürfen auf einer Plone-Site fast alles machen, d.h., Sie sollten diese Rolle nur an vertrauenswürdige Entwickler und Administratoren vergeben. Ein Manager darf Inhalte löschen oder bearbeiten, Benutzer entfernen, die Site-Konfiguration ändern und sogar Ihre Plone-Site löschen.

Gruppen

Das Konzept von Gruppen unterscheidet sich von dem von Rollen. Rollen bedeuten, dass ein Benutzer mit einer Rolle andere Rechte hat als jemand mit einer anderen Rolle, während eine *Gruppe* eine logische Kategorisierung von Benutzern darstellt. Die Marketing-Abteilung könnte z.B. eine Gruppe sein und die Entwicklungsabteilung eine andere. Jeder Benutzer kann zu keiner oder vielen Gruppen gehören. Gruppen sind optional, d.h., Sie müssen keine Gruppen ver-

wenden, aber das Plone-Team fand Gruppen sehr nützlich und hat sie daher eingebaut.

Die Entwickler einer Site können Gruppen nach Belieben benutzen, beispielsweise, um eine Abteilung oder eine bestimmte Art von Benutzern zu gruppieren. Den meisten Benutzern, die Plone zum ersten Mal einsetzen, rate ich, die Gruppen unverändert zu lassen. Per Voreinstellung werden keine Gruppen für Sie erzeugt.

Hinweis

Gruppen implementieren Sie mit dem *Group User Folder* (GRUF). Diese Gruppen sind kein Teil von Zope, sondern ein extra Werkzeug für Plone. GRUF wurde von Ingeniweb entwickelt und zur Verfügung gestellt.

9.1.2 Der Zugriffsrechte-Reiter

Als ich in Kapitel 3 die Veröffentlichung von Dokumenten behandelt habe, habe ich den ZUGRIFFSRECHTE-Reiter übersprungen, weil er eine Einrichtung für Fortgeschrittene ist, die Sie vermutlich nicht immer verwenden möchten. Der ZUGRIFFSRECHTE-Reiter ist eine Aktion in portal_actions, d.h., wenn Sie nicht möchten, dass er erscheint, gehen Sie im Zope Management Interface (ZMI) zu diesem Werkzeug und deaktivieren die Option *visible*. Der ZUGRIFFSRECHTE-Reiter ist aber recht nützlich, weil Sie damit Benutzern und Gruppen verschiedene lokale Rollen an einem Objekt in Plone zuweisen können.

Wenn Sie einen Inhalt in einer Plone-Site hinzugefügt haben und Sie möchten, dass eine andere Person diesen Inhalt bearbeiten kann, müssen Sie ihr mehr Rechte an diesem einen Objekt geben. Das bezeichnet man als eine *lokale Rolle*, die Ihnen erlaubt, einem Benutzer weitergehende Rechte an einem Objekt zu geben. Wenn ich ein Dokument in Plone schreibe, werde ich zum Besitzer dieses Dokuments und erhalte gewisse Rechte. Wenn ich vor der Veröffentlichung mit meinem Kollegen Ralf an diesem Dokument zusammenarbeiten möchte, dann muss ich Ralf erweiterte Rechte geben, damit er das Dokument bearbeiten darf. Dazu gehe ich zum ZUGRIFFSRECHTE-Reiter und gebe meinem Kollegen dort die nötigen Rechte.

Hinweis

Lokale Rollen können Sie auf einem Ordner oder einem Dokument vergeben. Wenn Sie einem Benutzer eine lokale Rolle an einem Ordner geben, dann erhält er diese lokale Rolle auf allen Objekten in diesem Ordner.

Der ZUGRIFFSRECHTE-Reiter erscheint nur dort, wo Sie das Recht haben, Zugriffsrechte zu verändern, dazu gehört ihr eigener Ordner. Klicken Sie erst auf MEIN ORDNER und dann auf ZUGRIFFSRECHTE. Abbildung 9.1 zeigt, wie das Formular für den ZUGRIFFSRECHTE-Reiter aussieht. Es besteht aus drei Hauptkomponenten, in denen Sie einem Benutzer oder einer Gruppe eine lokale Rolle an diesem Objekt zuordnen können. Hier sehen Sie auch, wer bereits bestimmte Rollen hat.

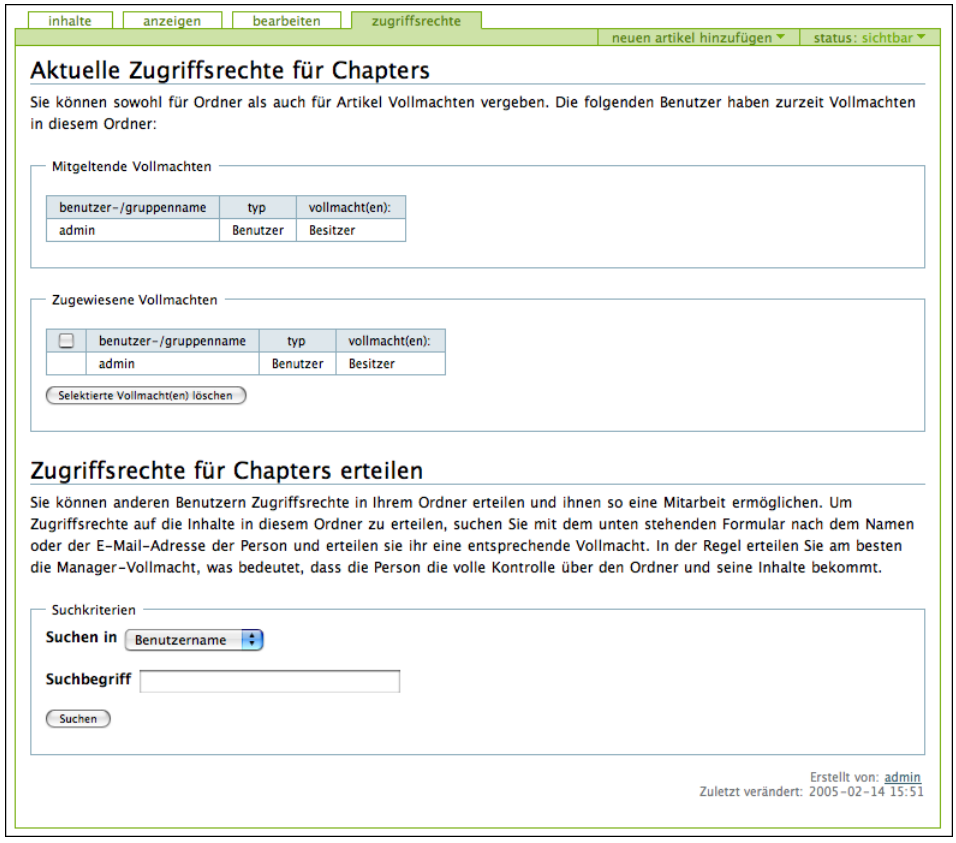


Abbildung 9.1: Zugriff auf den Zugriffsrechte-Reiter

Um einen Benutzer zu finden, dem die Rolle zugewiesen werden soll, geben Sie einen Suchbegriff ein, z.B. *Gavin*, wobei eine Liste von Benutzern geöffnet wird, die Ihrem Suchkriterium entsprechen. Dann können Sie den Benutzer anklicken und im Dropdown-Menü die gewünschte Rolle auswählen. In Abbildung 9.2 gebe ich dem Benutzer Gavin die Rolle des Besitzers an diesem Ordner.

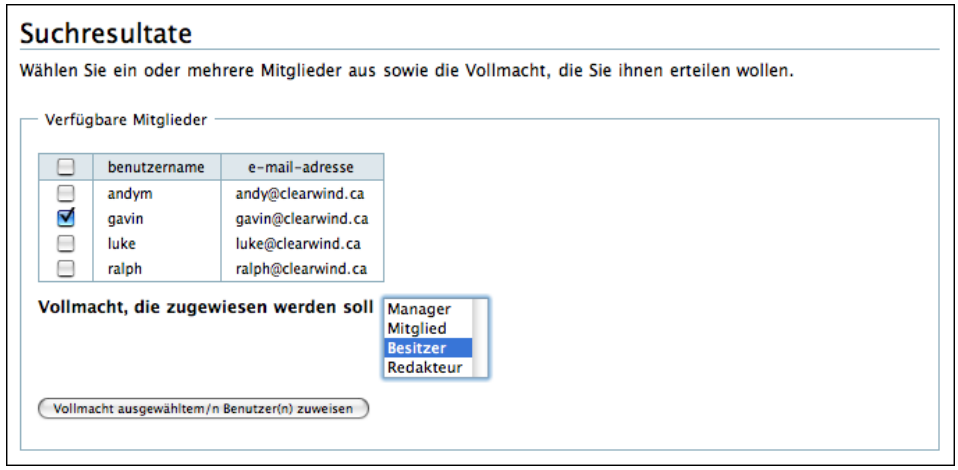


Abbildung 9.2: Zuweisen einer Rolle an einen Benutzer

In meinem vorigen Beispiel wollte ich Rechte an einen einzelnen Benutzer zuweisen, aber bei einer großen Anzahl von Benutzern kann das sehr umständlich sein, es sei denn, Sie haben ihnen Gruppen zugewiesen. Wenn ich möchte, dass die gesamte Marketing-Abteilung mein Dokument bearbeiten darf, kann ich das damit tun. Um die verfügbaren Gruppen zu erhalten, klicken Sie auf VIEW GROUPS, wonach eine Liste von Gruppen für diese Site geöffnet wird und Sie eine lokale Rolle an eine Gruppe zuweisen können. In Abbildung 9.3 gebe ich der Entwicklungsabteilung die Rolle des Besitzers an diesem Ordner.

In Abbildung 9.4 können Sie schließlich sehen, welche Benutzer und Gruppen Rollen mit Rechten an dieser Seite haben, um sie bei Bedarf zu entfernen. Sobald Sie jemand anderem einmal eine lokale Rolle an einem Objekt gegeben haben, erlauben Sie ihm, den ZUGRIFFSRECHTE-Reiter zu öffnen. Dann hindert ihn nichts mehr daran, Rollen für Sie aus dem Inhalt zu entfernen.

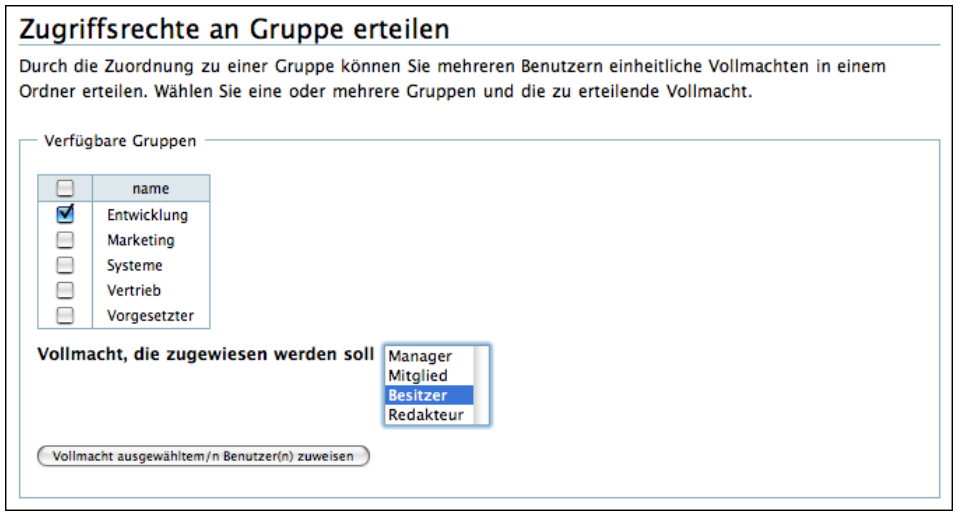


Abbildung 9.3: Zuweisen einer Rolle an eine Gruppe

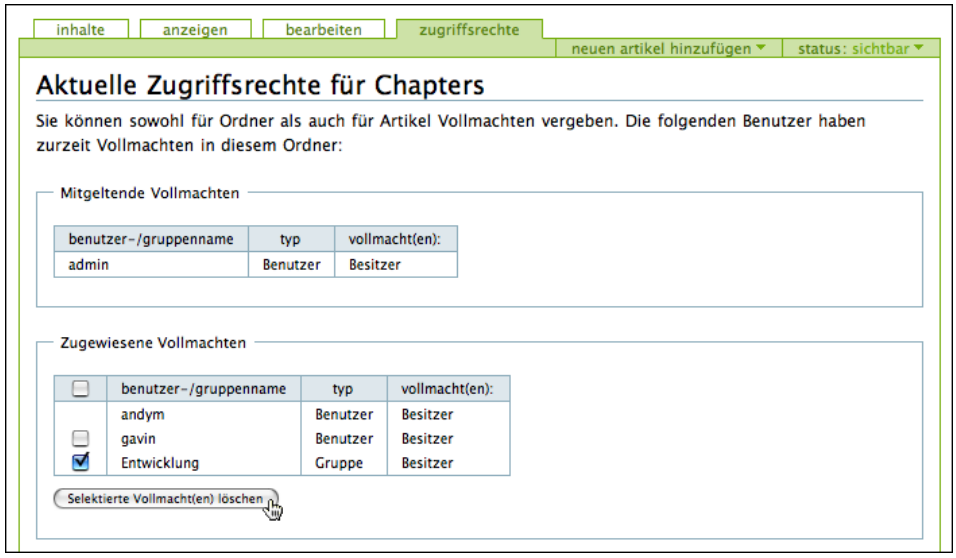


Abbildung 9.4: Rollen anzeigen und entfernen

9.1.3 Administration über das Web

Über die Plone-Schnittstelle können Sie sehr einfach den Benutzer verändern, der gewissen Gruppen zugewiesen worden ist, Benutzerangaben ändern, Gruppen hinzufügen usw. Das meiste hiervon können Sie über das Plone-Control Panel erledigen. Klicken Sie einfach auf PLONE KONFIGURATION, und wählen Sie dann

BENUTZER- UND GRUPPENVERWALTUNG. Dann sehen Sie zwei Reiter: BENUTZER und GRUPPEN.

Klicken Sie auf den BENUTZER-Reiter, um an die Liste der Benutzer auf dem System zu kommen. Das Formular ist ziemlich selbsterklärend: Sie können einen Benutzer entfernen, ein Passwort neu setzen (dadurch erhält der Benutzer ohnehin eine E-Mail) oder eine E-Mail ändern, und all das aus dem Formular heraus, das Sie in Abbildung 9.5 sehen.

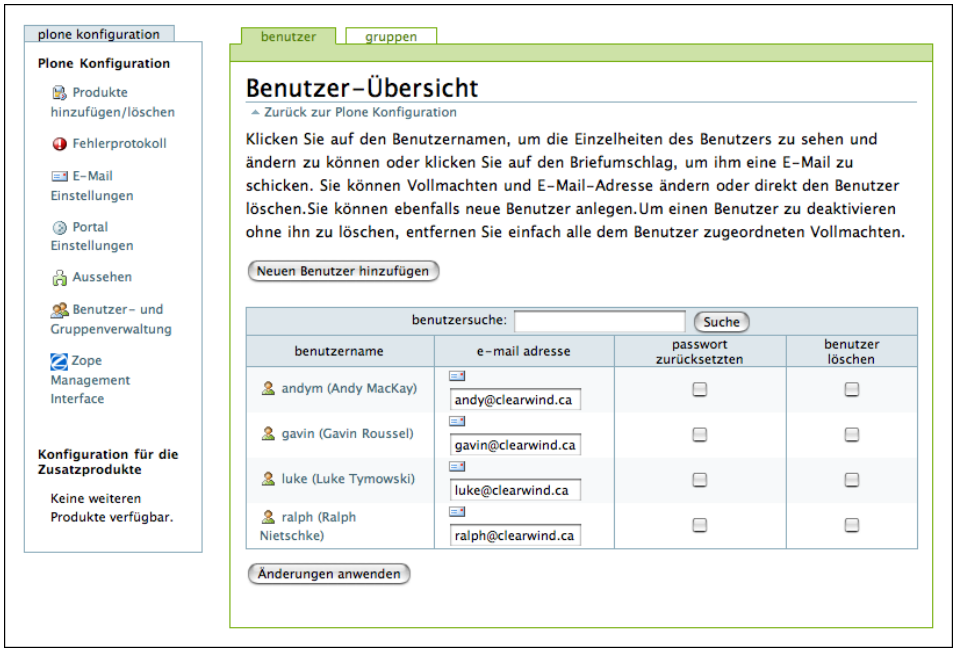


Abbildung 9.5: Benutzer bearbeiten

Durch einen Klick auf einen Benutzer kommen Sie zum Formular mit den Benutzereinstellungen dieses Benutzers, können dort Änderungen vornehmen und dann auf SPEICHERN klicken. Um einen neuen Benutzer hinzuzufügen, klicken Sie auf NEUEN BENUTZER HINZUFÜGEN. Danach wird das Formular für die Registrierung des Benutzers geöffnet, wo Sie auch Angaben zu diesem Benutzer ändern können. Da die Anzahl von Benutzern auf einer Site ziemlich groß werden kann, werden die Daten in der vertrauten Plone-Art auf mehrere kurze Listen aufgeteilt. Sie können einen Suchbegriff eingeben, mit dem über alle Benutzer hinweg nach jenen mit den passenden Namen und E-Mail-Adressen gesucht wird.

Wenn Sie auf den GRUPPEN-Reiter klicken, können Sie Gruppen hinzufügen, bearbeiten und löschen. Eine Gruppe fügen Sie hinzu, indem Sie auf den Button NEUE GRUPPE HINZUFÜGEN klicken. Danach wird ein Formular für eine Gruppe

geöffnet. Darin ist NAME das einzige Feld, das ausgefüllt werden muss. Dieser Name sollte ein kurzer beschreibender Name der Gruppe sein. Normalerweise hat eine Gruppe immer direkt mit einer Geschäfts- oder Site-Aktivität zu tun.

Jetzt, wo Sie eine Gruppe mit einigen Benutzern erstellt haben, können Sie Benutzer und Gruppen einander zuordnen, und zwar mit dem Plone-Control Panel. Sie können entweder einen Benutzer anklicken und diesem einige Gruppen zuordnen oder eine Gruppe anklicken und ihr Benutzer zuordnen.

Wann benutzt man Gruppen?

Gruppen sind optional, und vielleicht werden Sie nie einen Bedarf für Gruppen haben. Ein wichtiger Anwendungsfall von Gruppen besteht allerdings im Anlegen eines *Workspaces*. In einer einfachen Plone-Site können Benutzer Inhalte in ihrem eigenen Ordner anlegen und bearbeiten. Jedes dort erstellte Element hat diejenige Person als Besitzer, die es erstellt hat. Aber leider skaliert das nicht besonders gut. Und schließlich geht es eigentlich darum, dass einige Leute ein Dokument zusammen bearbeiten und austauschen können!

An dieser Stelle kommen Gruppen und Workspaces ins Spiel. So wie es einen Ordner für Mitglieder gibt, der alle Benutzerordner der Mitglieder enthält, gibt es auch einen Ordner namens `GroupWorkspaces`. Dieser wird standardmäßig immer dann erstellt, wenn eine Gruppe hinzugefügt wird, und in diesem Ordner gibt es für jede Gruppe einen weiteren Ordner. Wenn Sie also eine Gruppe namens *Marketing* hinzufügen, können Sie einen Ordner unter `GroupWorkspaces/Marketing` finden. Alle Benutzer in der Marketing-Gruppe haben das Recht, Inhalte im `Marketing`-Workspace zu erstellen, zu bearbeiten und zu löschen. Mit anderen Worten: Sie haben nun einen Ordner für diese Gruppe. Das ist gleichbedeutend damit, eine Gruppe hinzuzufügen und ihrem Ordner dann eine lokale Rolle zuzuordnen.

Das ist nur ein Beispiel dafür, wie nützlich eine Gruppe sein kann. Ein anderes wäre die Benutzung von Gruppen beim Workflow. Im vorigen Kapitel habe ich den Workflow erläutert und gezeigt, wie Sie eine E-Mail an bestimmte Leute schicken können, wenn irgendetwas passiert. Wenn z.B. ein Mitglied der Marketing-Gruppe ein Objekt hinzufügt, können Sie eine E-Mail an alle Benutzer dieser Gruppe schicken, statt einfach an alle Benutzer. Im Abschnitt »Die anderen Benutzer einer Gruppe bestimmen« weiter unten lernen Sie, wie Sie das machen.

Auf der Plone-Website z.B. sind die Benutzer verschiedenen Entwicklergruppen zugeordnet, die für verschiedene Teile von Plone verantwortlich sind, z.B. dem Release-Team und dem Dokumentations-Team.

Gruppen verwalten

Aus dem Plone-Control Panel heraus können Sie Gruppen auf zwei verschiedene Arten verwalten. Entweder gehen Sie zu einem Benutzer und klicken auf seine Gruppen, oder Sie gehen zu einer Gruppe und klicken auf ihre Benutzer. Auf beide Arten können Sie ganz einfach Gruppen zu einem Benutzer hinzufügen und entfernen. Um einen Benutzer zu einer Gruppe hinzuzufügen, gehen Sie besser zur Benutzersuchseite und klicken erst auf einen Benutzer und dann den Gruppen-Reiter, der die entsprechenden Gruppen anzeigt. In Abbildung 9.6 sehen Sie z.B. die Gruppen des Benutzers `andym`.

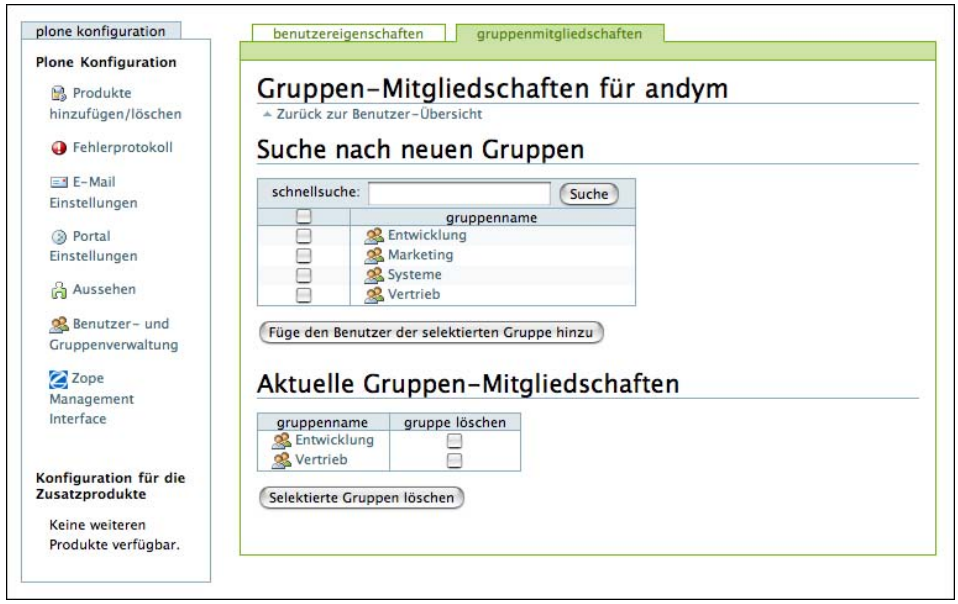


Abbildung 9.6: Gruppen für diesen Benutzer

Um den Benutzer zu einer neuen Gruppe hinzuzufügen, markieren Sie das Kästchen für die Gruppe und klicken dann auf FÜGE DEN BENUTZER DER SELEKTIERTEN GRUPPE HINZU.

Umgekehrt können Sie einen Benutzer aus einer Gruppe entfernen, indem Sie das Kästchen neben der Gruppe markieren und auf SELEKTIERTE GRUPPEN LÖSCHEN klicken. Eine ähnliche Schnittstelle sehen Sie bei der Gruppenverwaltung, wenn Sie auf PLONE KONFIGURATION, GRUPPEN- UND BENUTZERVERWALTUNG und GRUPPEN klicken. Klicken Sie auf eine Gruppe und dann auf GRUPPENMITGLIEDER, und Sie erhalten eine Liste von Mitgliedern in dieser Gruppe, in der Sie Mitglieder hinzufügen und löschen können.

Rollen an Gruppen vergeben

Nun haben Sie gesehen, dass sowohl Benutzer als auch Gruppen Rollen haben können. Das mag Ihnen seltsam vorkommen, aber denken Sie z.B. an eine Gruppe von Vorgesetzten, die alles mit dem Inhalt tun können muss, der von einem ihrer Mitarbeiter erstellt worden ist. Um das auf der Site zu machen, müssen Sie die Rolle eines Redakteurs haben. Um eine Gruppe von Vorgesetzten einzurichten, klicken Sie auf PLONE KONFIGURATION, BENUTZER- UND GRUPPENVERWALTUNG, GRUPPEN und dann auf NEUE GRUPPE HINZUFÜGEN. Nennen Sie diese Gruppe **Vorgesetzter**, und füllen Sie das Formular vollständig aus. Im nächsten Formular erhalten Sie eine Liste der Gruppen und der ihnen zugeordneten Rollen. Um dieser Gruppe die Rolle des Redakteurs zuzuweisen, wählen Sie die Kontrollkästchen, die der Redakteursrolle für diese Gruppe entsprechen, wie in Abbildung 9.7 zu sehen ist.

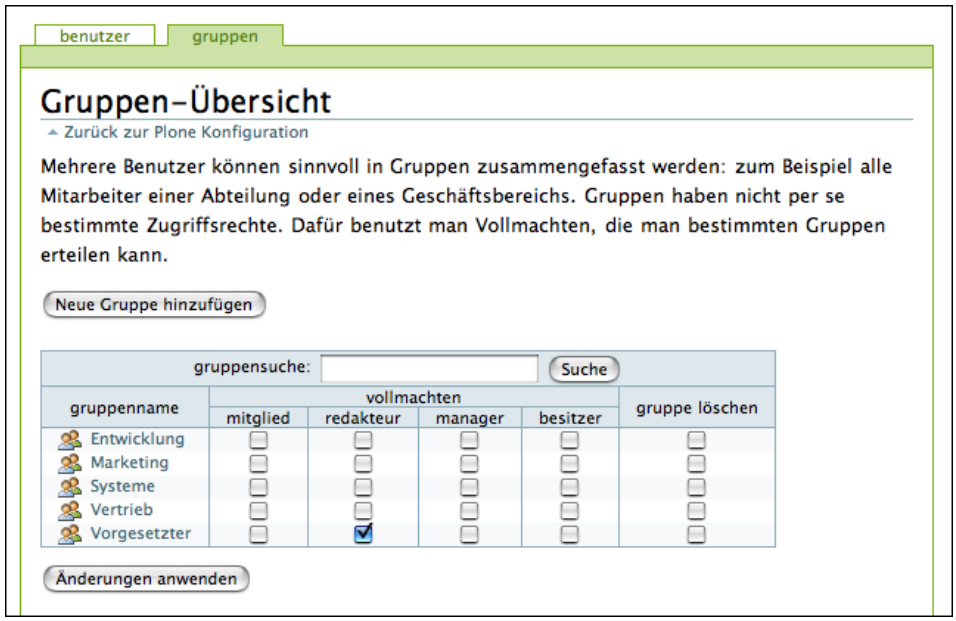


Abbildung 9.7: Einrichten der Redakteursrolle für die Gruppe Vorgesetzter

Sie haben es leicht gemacht, den Benutzern die Redakteursrolle zuzuweisen, und nun können Sie die Redakteure über die Plone-Schnittstelle verwalten. Außerdem ist es einfach, die Redakteure mit einem Programm zu bestimmen, weil Sie die Gruppe untersuchen und eine Liste ihrer Mitglieder erhalten können.

Die Idee von Gruppen mit Rollen ist tatsächlich ein kleiner Paradigmenwechsel, verglichen mit der normalen Entwicklung unter Zope, weil Sie dort an individuelle Benutzer gewöhnt sind, denen Rollen zugeordnet sind. Das können Sie in

Plone natürlich weiterhin haben, aber die Zuweisung von Rollen an Gruppen ist in Plone sehr einfach.

Hinweis

Per definitionem gilt: Bei der Bestimmung der Rechte eines Benutzers an einem Objekt werden verschiedene Faktoren berücksichtigt. Erstens werden die Rollen bestimmt, die einem Benutzer zugewiesen sind. Zweitens werden die Rollen bestimmt, die ein Benutzer aus seinen Gruppen erhält. Daraus ergibt sich die Gesamtmenge an Rollen, über die ein Benutzer verfügt.

9.2 Registrierungswerkzeuge für Benutzer

Um Mitglied auf Ihrer Site zu werden, müssen sich Benutzer darauf erst einmal registrieren. Benutzer können sich sehr leicht selbst registrieren, indem sie in der oberen rechten Ecke einer Plone-Site auf MITGLIED WERDEN klicken. Das habe ich detailliert am Anfang von Kapitel 3 behandelt. Der Registrierungsvorgang für Benutzer ist ziemlich einfach, aber es sind einige Optionen dabei verfügbar. Dieser Vorgang wird von drei wichtigen Werkzeugen gesteuert: `portal_registration`, `portal_memberdata` und `portal_membership`, die nun in den folgenden Abschnitten vorgestellt werden.

9.2.1 Portal-Registrierung

Das Werkzeug `portal_registration` bietet eine wichtige Aktion in Plone an, nämlich ein Mitglied zu werden. Ein Klick auf diesen Link öffnet das Mitglied-Formular. Standardmäßig können alle noch nicht angemeldeten Benutzer (auch anonyme) diesen Link anklicken, um Mitglied zu werden.

Wenn sich Benutzer mit dem Mitglied-Formular registrieren, haben sie zwei einfache Möglichkeiten für eine Plone-Site: mit und ohne Validierung einer E-Mail-Adresse. Die einzig wahre Methode, eine E-Mail-Adresse zu validieren, besteht darin, eine Nachricht an diese Adresse zu schicken, um zu sehen, ob eine entsprechende Antwort zurückkommt. Die Validierung von E-Mails ist standardmäßig ausgeschaltet, d.h., wenn Benutzer sich registrieren, geben sie per Voreinstellung ihren Namen, E-Mail und Passwort in Plone ein. Dann können sie sich anmelden und die Site ganz normal benutzen. Dieses Formular haben Sie in Kapitel 3 gesehen. Falls aber die Validierung von E-Mail-Adressen eingeschaltet ist, können die Benutzer nur einen Namen, einen Benutzernamen und eine E-Mail-Adresse eingeben, wie in Abbildung 9.8 zu sehen ist.

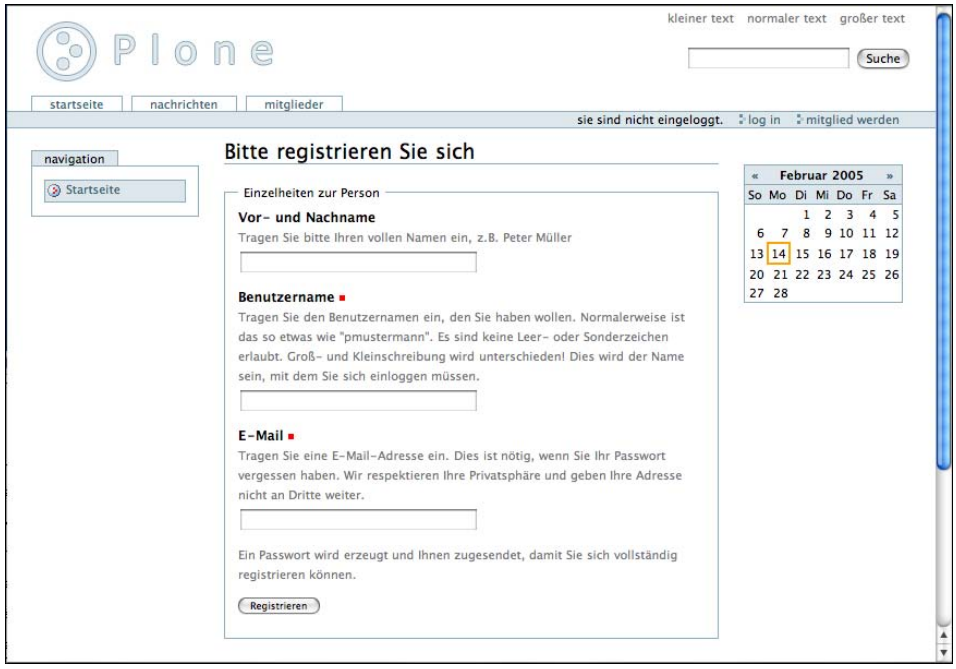


Abbildung 9.8: Benutzer-Registrierung mit eingeschalteter E-Mail-Validierung

Nach einem Klick auf den Link in der erhaltenen E-Mail, gelangen die Benutzer zu einem Anmeldeschirm, und der Registrierungsvorgang wird normal fortgesetzt.

Exkurs: Hinzufügen einer Aktion für Benutzer

Wenn Sie weitere Aktionen für Benutzer hinzufügen möchten, bevor diese Mitglied werden, ist das der beste Ort dafür. Wenn Sie z.B. eine Seite hinzufügen möchten, die Angaben zum Datenschutz macht, wäre hier ein guter Platz dafür. Fügen Sie dazu erst die Seite mit allen erforderlichen Angaben hinzu. Es wäre sinnvoll, dieser Seite eine nützliche ID zu geben, z.B. `datenschutz.html`, und sie in dem Wurzelobjekt Ihrer Plone-Site zu platzieren.

Gehen Sie im ZMI zu `portal_registration`, und fügen Sie eine neue Aktion mit den folgenden Angaben hinzu:

```
Name: Datenschutz
Id: datenschutz
Action: string: ${portal_url}/datenschutz.html
Condition: not: member
Permission: Add portal member
Category: user
Visible: angehakt
```

Nun erhalten Sie einen Link auf Ihre Datenschutz-Seite, wenn Sie noch nicht angemeldet sind. Durch die Wahl von *user* als Kategorie stellen Sie sicher, dass der Link in der persönlichen Leiste erscheint.

Um die Validierung in der Plone-Schnittstelle zu aktivieren, klicken Sie auf PLONE KONFIGURATION und PORTAL EINSTELLUNGEN. Unter PASSWORTEIGENSCHAFTEN wählen Sie *Erzeuge für das Mitglied ein Passwort und sende es ihm per E-Mail zu*. und klicken auf SPEICHERN, um die Änderung zu bestätigen.

Wenn Sie die an Benutzer verschickte E-Mail anschauen oder bearbeiten möchten, können Sie das Page Template bearbeiten, aus dem sie erzeugt wird. Sie finden das Template, indem Sie auf PLONE_SKINS, PLONE_TEMPLATES und REGISTERED_NOTIFY_TEMPLATE klicken.

9.2.2 Mitgliederdaten

Das Werkzeug `portal_memberdata` enthält die Mitgliederdaten aller Benutzer. Zu einem Plone-Benutzer gibt es eine Reihe von Angaben, z.B. zu Skins, zum Zeitpunkt seiner letzten Anmeldung, zu seinem WYSIWYG-Editor (What You See Is What You Get) usw. Wenn ein Benutzer Mitglied auf einer Site wird, wird ein Standarddatensatz in `portal_memberdata` erzeugt. Die tatsächlich darin erzeugten Eigenschaften sehen Sie mit diesem Werkzeug. Klicken Sie auf PORTAL_MEMBERDATA, und wählen Sie PROPERTIES, um die Eigenschaften im Standarddatensatz zu sehen. In Plone sind das folgende:

- **e-mail**: Die E-Mail-Adresse des Benutzers.
- **portal_skin**: Diese Eigenschaft ist veraltet und sollte ignoriert werden.
- **listed**: Zeigt diesen Benutzer im `Members`-Ordner an (boolescher Wert). Ist standardmäßig eingeschaltet.
- **login_time**: Das Datum der letzten Anmeldung in dieser Sitzung des Benutzers.
- **last_login_time**: Das Datum der letzten Anmeldung des Benutzers.
- **fullname**: Der vollständige Name des Benutzers.

- **error_log_update**: Wird vom Fehlerprotokollformular verwendet, bitte ignorieren.
- **formtooltips**: In älteren Versionen von Plone gab es Optionen für die Anzeige von Hilfen in Formularen. Heute ist das nicht mehr relevant, daher bitte ignorieren.
- **visible_ids**: Zeigt die IDs (oder Namen) von Objekten an. Wenn aktiviert, ist das erste Feld im BEARBEITEN-Formular aller Inhaltstypen immer der Name. Durch eine Änderung des Namens können Benutzer ein Objekt umbenennen. Ist per Voreinstellung aktiviert.
- **wysiwyg_editor**: Der Editor, der in Formularen benutzt werden soll.

Über die Zope-Schnittstelle können Sie Einträge in dieser Liste hinzufügen oder löschen. Allerdings wird dabei nicht automatisch das Formular der Benutzerschnittstelle neu erstellt, das die Benutzer tatsächlich bearbeiten. In Kapitel 3 haben Sie gesehen, dass Benutzer durch einen Klick auf MEINE EINSTELLUNGEN auf die meisten dieser Eigenschaften zugreifen und sie ändern können. Wenn Sie diese Einstellungen ändern möchten, müssen Sie dieses Formular anpassen. Die in diesen Feldern angegebenen Werte sind voreingestellte Werte für einen neu registrierten Benutzer. Standardmäßig werden z.B. alle Mitglieder unter dem Mitglieder-Reiter aufgelistet, außer die Benutzer entscheiden sich explizit dagegen.

Wenn Sie z.B. nicht möchten, dass alle Mitglieder standardmäßig bei der Suche aufgelistet werden, müssen Sie die Einstellung in diesem Formular ändern. Finden Sie im Formular `portal_memberdata` die Eigenschaft `listed`, und entfernen Sie die Markierung für den Wert im Formular. Klicken Sie dann auf SAVE CHANGES, und alle neuen Benutzer werden nicht mehr aufgelistet.

Das Werkzeug `portal_groupdata` enthält die entsprechenden Daten für Gruppen. Die Standardeigenschaften bei einer Gruppe lauten wie folgt:

- **title**: Ein Titel der Gruppe
- **description**: Eine Beschreibung der Gruppe
- **email**: Eine E-Mail-Adresse
- **listed**: Angabe, ob die Gruppe für Benutzer aufgelistet wird

Diese Werkzeuge speichern Benutzer- und Gruppendaten in den Werkzeugen selbst und nicht im `acl_users`-Ordner. Wenn Sie Benutzerinformationen von einem Plone-Server zum anderen verschieben möchten, müssen Sie diese Werkzeuge ebenfalls verschieben. Nur den `acl_users`-Ordner zu verschieben reicht nicht aus. Das können Sie tun, indem Sie diese Werkzeuge importieren und exportieren. Vor dem Import auf der neuen Plone-Site müssen Sie allerdings das existierende Werkzeug löschen, sonst wird ein Fehler ausgelöst.

9.2.3 Mitgliedschaften

Das Werkzeug `portal_membership` verwaltet einige weitere Eigenschaften. Insbesondere verknüpft es die Mitgliederdaten mit den Mitgliedern. Beim Zugriff per ZMI auf `portal_membership` haben Sie eine große Anzahl von Optionen, von denen folgende die wichtigsten sind:

- **Set members folder**: Dies ist der Ordner, der die Mitglieder-Ordner enthält. Dieser Ordner muss vorhanden sein. Per Voreinstellung hat er den Namen Members.
- **Control creation of member areas**: Standardmäßig wird für jeden Benutzer ein Mitgliederbereich erstellt, wenn er Mitglied wird, was allerdings optional ist. Klicken Sie auf TURN FOLDER CREATION OFF, um das abzuschalten. Standardmäßig werden diese Bereiche erstellt.

Unter dem ACTIONS-Reiter finden Sie eine Reihe von Aktionen, die bei angemeldeten Benutzern Anwendung finden, z.B. *Meine Favoriten*, *Meine Einstellungen* usw. Diese haben alle die Kategorie *user*, damit die Aktionen in der oberen rechten Ecke erscheinen.

Das Werkzeug `portal_groups` bietet ähnliche Möglichkeiten wie `portal_membership`, aber für Gruppen. Auch hier gilt, dass bei der Erzeugung einer Gruppe ein Gruppen-Workspace erstellt wird, in dem alle Mitglieder dieser Gruppe Inhalte erstellen und bearbeiten können.

9.2.4 Nützliche APIs

Das Werkzeug `portal_membership` enthält einen der meistverwendeten Sätze an API-Funktionen. Oftmals möchten Sie wichtige Informationen herausfinden, wie z.B. den gerade angemeldeten Benutzer, ob der Benutzer anonym ist usw. Das Werkzeug `portal_membership` bietet Ihnen hierfür die nötigen Methoden, von denen die folgenden am wichtigsten sind:

- **isAnonymousUser()**: Gibt `True` zurück, falls der Benutzer anonym ist.
- **getAuthenticatedMember()**: Gibt den gerade angemeldeten Benutzer mitsamt seinen Eigenschaften aus `portal_metadata` zurück. Wenn keiner angemeldet ist, wird ein spezieller Benutzer `nobody` ohne `portal_metadata`-Eigenschaften zurückgegeben.
- **listMemberIds()**: Gibt die IDs aller Benutzer zurück.
- **listMembers()**: Gibt alle Benutzerobjekte zurück.
- **getMemberById(id)**: Gibt das Benutzerobjekt zu einer gegebenen ID zurück.

- **getHomeFolder(id=None)**: Gibt den Startordner zu einer gegebenen ID zurück. Die ID ist optional. Ohne ID wird der Startordner des aktuellen Mitglieds zurückgegeben.
- **getHomeUrl(id=None)**: Gibt eine URL zum Startordner des Mitglieds zurück. Die ID ist optional. Ohne ID wird die URL des Startordners des aktuellen Mitglieds zurückgegeben.

Der von diesen Funktionen zurückgegebene Benutzer wird mit Daten aus dem Werkzeug `portal_memberdata` versehen, so dass diese Eigenschaften Attribute des Benutzerobjekts werden. Das folgende Beispiel ist ein kleines Script (Python)-Objekt, um an die E-Mail-Adresse des Benutzers `Andy` zu gelangen:

```
##parameters=
u = context.portal_membership.getMemberById("andy")
return u.email
```

9.2.5 Cookie-Authentifikation

Plone verwendet standardmäßig eine Cookie-Authentifikation bei der Identifikation seiner Benutzer, d.h., Benutzer müssen Cookies in ihrem Browser eingeschaltet haben, damit sie sich anmelden können. Diese Authentifikation erfolgt auf einer Plone-Site mit dem Objekt `cookie_authentication`, das die notwendige Funktionalität für die Anmeldung von Benutzern enthält. Falls Sie wirklich eine HTTP-Authentifikation (Hypertext Transfer Protocol) benutzen möchten, können Sie dieses Objekt einfach löschen. Allerdings möchte ich das wirklich nicht empfehlen, denn bei den meisten Sites eignet sich eine HTTP-Authentifikation nicht.

Dieses Objekt bietet folgende Eigenschaften, die Sie im ZMI bearbeiten können:

- **Authentication cookie name**: Der Name des Cookies, das für die persistente Benutzer-Authentifikation benutzt wird. Das funktioniert über ein persistentes Token für den Benutzer, das die Anmeldung des Benutzers enthält. Der Standardwert hierfür lautet `__ac`.
- **User name form variable**: Der Name der Variablen im Anmeldeformular, die den Benutzernamen enthält. Der Standardwert hierfür ist `__ac_name`.
- **User password form variable**: Der Name der Variablen im Anmeldeformular, die das Passwort enthält. Standardwert ist `__ac_password`.
- **User name persistence form variable**: Der Name der Variablen im Anmeldeformular, die das persistente Token enthält. Standardwert ist `__ac_persistent`.

- **Login page ID**: Wenn ein Benutzer sich anmelden muss, ist dies die Seite, zu der er gelangt, um die Anmeldung abzuschließen. Standardwert ist `require_login`.
- **Logout page ID**: Wenn ein Benutzer dabei ist, sich abzumelden, gelangt er zu einer netten Seite mit einer Nachricht darauf. Diese Seite hat diese ID. Der Standardwert ist `logged_out`.
- **Failed authorization page ID**: Diese Seite erscheint, wenn die Autorisierung fehlschlägt. Standardmäßig ist sie leer, da Plone etwas anderes macht.
- **Use cookie paths to limit scope**: Setzt das Cookie lokal zum aktuellen Ordner und allen Ordnern darunter. Lassen Sie hier den Standardwert (leer) stehen, damit Sie sich für die gesamte Site authentifizieren, unabhängig davon, wo Sie auf EINLOGGEN klicken.

Um andere Cookies als die Standard-Cookies zu verwenden, ändern Sie einfach den Wert in diesem Formular und klicken auf SAVE. Aber seien Sie sich bewusst, dass, wenn Sie den Namen des Cookies ändern, alle vorhandenen Cookies auf den Rechnern Ihrer Benutzer ignoriert werden, d.h., die Benutzer müssen sich alle neu anmelden. Wenn Sie gern eine andere Anmeldeseite hätten, könnten Sie entweder das Page Template `require_login` anpassen oder den Wert dieser Variablen ändern.

9.2.6 Der eigentliche Benutzerordner

Wenn Sie im ZMI auf den Ordner `acl_users` klicken, erhalten Sie Zugriff auf den eigentlichen Benutzerordner einer Plone-Site. Dabei wird die Schnittstelle zum Gruppenbenutzerordner (GRUF, Group User Folder) geöffnet, in der Sie eine Vielzahl von Optionen haben.

Die GRUF-Schnittstelle ist im Prinzip den Benutzeroptionen recht ähnlich, die Sie im Plone-Control Panel haben. Sie können Benutzer und Gruppen über eine recht einfach zu verstehende Schnittstelle hinzufügen und bearbeiten. Mit einem Klick auf BENUTZER und GRUPPEN können Sie diese Elemente bearbeiten. Wenn Sie auf den CONTENTS-Reiter klicken, erhalten Sie eine Auswahl zwischen Benutzern und Gruppen. Klicken Sie auf USERS und dann auf ACL_USERS. Damit kommen Sie schließlich zum aktuellen Benutzerordner eines Benutzers. Dieser sieht wie der Standard-Benutzerordner aus. Sie sehen eine Liste von Benutzern, und um einen davon zu bearbeiten, klicken Sie einfach auf den Benutzernamen, wie in Abbildung 9.9 zu sehen ist.

An dieser Stelle können Sie das Passwort eines Benutzers oder seine Rollen verändern. Sie werden bemerken, dass hier die Gruppe MANAGEMENT tatsächlich als Rolle dargestellt ist, um sicherzugehen, dass keine Namenskollisionen entste-

hen. Der Name wird zu `group_Management` verändert. Wenn Sie aus einem Benutzer ein Mitglied dieser Gruppe machen möchten, könnten Sie das hier tun. Aber Sie können hier nicht viel machen, was Sie nicht auch auf höchster Ebene tun könnten, daher würde ich nicht bis auf diese Stufe hinunter gehen, es sei denn, Sie müssen hier z.B. das Passwort ändern oder eine Domain setzen.

Abbildung 9.9: Bearbeiten des Benutzerdatensatzes

9.3 Rechte setzen

Nun haben Sie Benutzer, Rollen und Gruppen kennen gelernt, aber es gibt noch mehr. Die niedrigste Stufe bei Sicherheitseinstellungen sind die Rechte. Wie aus der Bezeichnung hervorgeht, bedeutet, einem Benutzer ein *Recht* zu geben, dass er die Möglichkeit hat, etwas zu tun, z.B. ein Objekt anzuzeigen, ein Dokument zu erstellen, eine Liste mit dem Inhalt eines Ordners zu bekommen usw. Jedes Recht wird mit einem sinnvollen Namen eindeutig bezeichnet, z.B. *View*, *Add portal content* oder *List folder contents*.

Rechte werden nicht an einen einzelnen Benutzer vergeben, sondern an eine Rolle. Jede Rolle erhält bestimmte Rechte, und dann bekommt der Benutzer bestimmte Rollen. Alle Sicherheitseinstellungen von Zope finden Sie über das ZMI unter dem SECURITY-Reiter. Dazu gehören die Plone-Site, das Zope-Wurzelobjekt, alle Objekte und Inhalte in einer Plone-Site sowie die Skins. Nach einem Klick auf den SECURITY-Reiter können Sie in einem Raster alle Rechte sehen und die Rollen, auf die sie abgebildet werden (siehe auch Abbildung 9.10).

In Abbildung 9.10 sehen Sie, dass dieses Objekt eine Reihe von Sicherheitseinstellungen hat. Angezeigt werden sie in Form eines Rasters mit Kontrollkästchen darin. Links stehen die Rechte in alphabetischer Reihenfolge, und oben stehen die Rollen, ebenfalls in alphabetischer Reihenfolge. Diese Seite ist ziemlich groß und unpraktisch, aber es gibt zwei nützliche Abkürzungen. Klicken Sie auf das Recht, um alle Rollen zu diesem Recht zu erhalten. In Abbildung 9.11 sehen Sie z.B. die Einstellungen zu dem Recht *Access future portal content*.

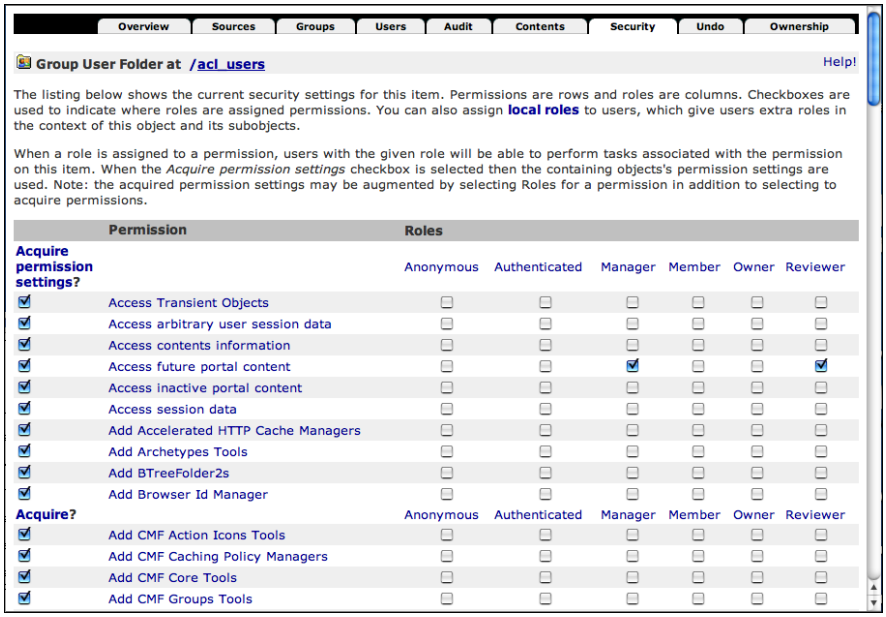


Abbildung 9.10: Sicherheitseinstellungen

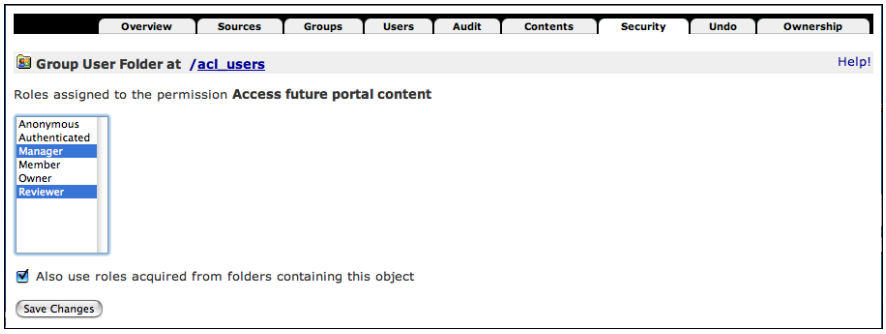


Abbildung 9.11: Rechte-Einstellungen

Und Sie können auf eine Rolle klicken, um alle Einstellungen zu dieser Rolle zu erhalten, was viel einfacher ist, als eine lange Liste (siehe Abbildung 9.12).

Bei all diesen Rechten muss man einfach nur die Kästchen für die gewünschten Rechte markieren oder die Optionen im Auswahlmenü wählen und auf SAVE klicken. Falls die Einstellung *Acquire Permission* gewählt ist, werden die Sicherheitseinstellungen für dieses Recht akquiriert, sonst werden die Rechte nicht akquiriert. *Akquisition* bezeichnet die Möglichkeit eines Objekts, in der Objekthierarchie nach Rechten zu suchen und die gefundenen Rechte zu den Gesamtrechten zu kombinieren.

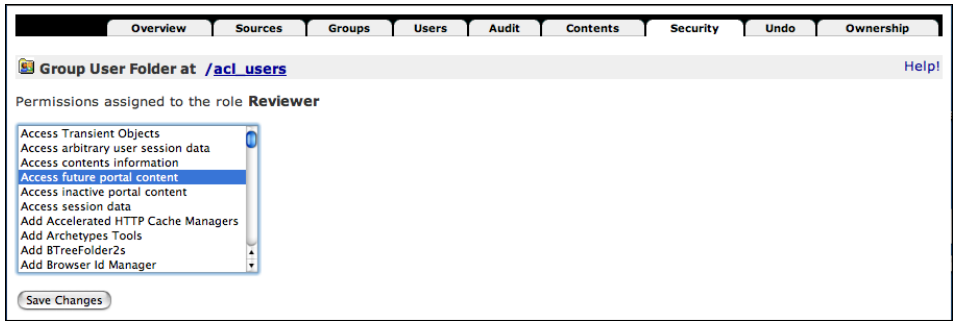


Abbildung 9.12: Einstellungen für die Redakteursrolle

Hinweis

Die Rechte-Seite schaltet für Sie die Berechtigungen für den Manager ein. Ihren Manager auszuschließen wäre sehr schlecht, daher ist es nicht verkehrt, sie standardmäßig eingeschaltet zu haben.

Betrachten Sie nun das Recht *Access contents information*. Gehen Sie im ZMI zum Plone-Wurzelobjekt, und klicken Sie auf den SECURITY-Reiter. In der Standardeinstellung für dieses Recht sind keine Rollen aktiviert, d.h., die Einstellungen sind für alle Benutzer leer. Allerdings ist die Option ACQUIRE SETTINGS markiert, d.h., Sie müssen in den Elternobjekten in der Hierarchie nachschauen, um die Rechte dieses Objekts zu bestimmen. Gehen Sie nun zum Zope-Wurzelordner, und klicken Sie auf den SECURITY-Reiter. Danach wird die Liste der Rechte für den Wurzelordner geöffnet, in der es ganz bestimmt einige Einstellungen für das Recht *Access contents information* in diesem Ordner gibt, und zwar haben die Rollen *Anonymous* und *Manager* dieses Recht.

Da Rechte akquiriert werden, erhalten alle Unterordner ebenfalls diese Sicherheitseinstellungen. Das heißt, die Plone-Site und jedes Objekt in der Plone-Site wird diese Rechte haben. Wenn Sie also eine Sicherheitseinstellung für die gesamte Site festlegen möchten, müssen Sie lediglich das Recht in dem Plone-Wurzelobjekt konfigurieren, und die meisten Objekte werden diese Rechte akquirieren.

Hinweis

Eine Ausnahme bilden Objekte im Workflow, die die Akquisition explizit ausschalten. Das wird später im Abschnitt »Sicherheit und Workflow« weiter unten in diesem Kapitel beschrieben.

In Zope können Sie Rechte auf allen Objekten über das ZMI setzen. Das kann das Zope-Wurzelobjekt, eine Plone-Site, ein Ordner wie z.B. der `Members`-Ordner oder sogar ein gewisser Inhalt sein. Jedes Objekt hat seinen eigenen Satz an Rechten, aber nicht alle Objekte haben die gleiche Auswahl an Rechten. Das Recht *Add...* haben beispielsweise alle Ordner. Aber da diese Rechte in einem Objekt keinen Sinn machen, das kein Ordner ist (ein Objekt muss per definitionem ein Ordner sein, damit man darin Elemente hinzufügen kann), sind sie nicht vorhanden.

Alle Produkte oder Teile von Python-Code in Ihrer Zope-Site können ihre eigenen Sicherheitsrechte definieren, daher kann es etwas schwierig werden, genau zu definieren, was Sie mit einem bestimmten Recht tun dürfen. Tabelle 9.1 beschreibt einige der wichtigsten Rechte und was sie tun.

Recht	Beschreibung
Access contents information	Dieses Recht erlaubt den Zugriff auf ein Objekt, ohne das Objekt notwendigerweise anzuzeigen. Ein Benutzer möchte z.B. den Titel des Objekts in einer Ergebnisliste sehen, obwohl er den Inhalt der Datei nicht sehen kann.
Add...	Es gibt zahlreiche Rechte beim Hinzufügen, abhängig vom Typ des Objekts, das ein Benutzer erstellen möchte. Bei einer normalen Plone-Site sind alle Rechte zusammen unter *Add portal content* gruppiert.
Add portal member	Hiermit hat man die Möglichkeit, Mitglied einer Plone-Site zu werden und ein Benutzerkonto zu bekommen.
Copy or Move	Hiermit darf man ein Objekt kopieren oder verschieben. Auch wenn Benutzer dieses Recht haben, müssen Sie dennoch das Recht haben, das Objekt an irgendeinem Ort einzufügen.
Delete objects	Hiermit hat man das Recht, ein Objekt zu löschen. Im normalen Zope wird dieses Recht im Ordner geprüft. In Plone wird diese Prüfung bei allen Objekten vorgenommen.
List folder contents	Hiermit bekommt man eine Liste des Inhalts eines Ordners. Dabei wird nicht geprüft, ob Sie das Recht haben, das aufgelistete Objekt anzuzeigen.

Tabelle 9.1: Häufige Plone-Rechte

Recht	Beschreibung
List portal members	Hiermit hat man das Recht, eine Mitgliederliste der Site zu sehen und nach Mitgliedern zu suchen.
Modify portal content	Dies ist ein zusammengefasstes Recht für beliebige Änderungen an einem Objekt, z.B. Ändern des Inhalts, seiner Stichwörter oder anderer Eigenschaften. Dieses Recht gilt für fast alle Objekte.
Set own password	Hiermit hat man das Recht, sein eigenes Passwort in einer Plone-Site zu ändern.
Set own properties	Hiermit hat man das Recht, seine eigenen Eigenschaften in einer Plone-Site zu ändern.
View	Hiermit dürfen Benutzer das fragliche Objekt anzeigen. Dabei bedeutet *View* nicht nur Anzeigen als HTML, sondern auch via FTP (File Transfer Protocol), WebDAV und über andere Zugriffsformen.

Tabelle 9.1: Häufige Plone-Rechte (Forts.)

9.3.1 Rollen hinzufügen

Rollen an Benutzer zu vergeben bedeutet, dass Sie einen derart kompatiblen Satz von Rechten für jede Rolle finden müssen, dass die Gruppierung der Rechte Sinn macht. Das ist nicht immer möglich. Manchmal braucht ein bestimmter Benutzer etwas anderes als andere Benutzer.

Aus Entwicklersicht ist allerdings alles umso einfacher, je weniger und einfacher Sie Ihre Rollen halten können. Es ist nicht sehr kompliziert, aber es empfiehlt sich nicht, gleich für jede denkbare Sicherheitsoption eine Rolle anzulegen. Sie bringen sich damit selbst sehr schnell in ein großes Schlamassel. Ich möchte Sie stattdessen dazu bringen, die Zahl Ihrer Rollen klein und allgemein genug formuliert für die gesamte Site zu halten.

Um eine Rolle hinzuzufügen, gehen Sie zum Plone-Wurzelordner, klicken auf den SECURITY-Reiter und scrollen zum unteren Seitenende (ein langer Weg). Unten ist ein einfaches Formular, mit dem man weitere Rollen hinzufügen oder Rollen löschen kann. Fügen Sie den Namen einer neuen Rolle hinzu, und klicken Sie auf ADD ROLE.

9.3.2 Häufige Aufgaben erledigen

Einige Sicherheitseinstellungen können Sie schnell und einfach setzen, um häufige Aufgaben zu erledigen. Bevor Sie viele Änderungen an den Sicherheitseinstellungen vornehmen, empfehle ich Ihnen dringend, ein Backup Ihrer Plone-Site anzulegen. Wie Sie das machen, zeige ich Ihnen in Kapitel 14.

Verhindern weiterer Site-Mitgliedschaften

Um zu verhindern, dass weitere Benutzer Mitglied auf Ihrer Site werden, benutzen Sie das Recht *Add portal member* in Ihrer Zope-Wurzel für anonyme Benutzer. Dieses können Sie entweder dort für anonyme Benutzer deaktivieren, oder Sie gehen zu Ihrer Plone-Site und schalten die Einstellung *Acquire Permission* ab.

Blockieren der Suche auf der Site

Um zu verhindern, dass Benutzer auf Ihrer Site suchen können, benutzen Sie in der Wurzel einer Plone-Site das Recht *Search ZCatalog* für anonyme Benutzer. Ändern Sie also dort das Recht, indem Sie Anonymous oder einen anderen Benutzer deaktivieren.

Site-Zugriff für anonyme Benutzer völlig blockieren

Nun ja, anonyme Benutzer gänzlich von Ihrer Site fernzuhalten ist etwas trickreich, weil es ziemlich kompliziert ist, den anonymen Zugriff auf Ihre Site völlig zu blockieren. Die Benutzer müssen weiterhin auf Ihre Site zugreifen können, um sich anzumelden! Was Sie in dieser Situation wirklich möchten, ist den Zugriff auf Ihren Inhalt einzuschränken. Das können Sie mit einer Einschränkung der Rechte in Ihrem Workflow erreichen.

9.3.3 Sicherheit und Workflow

Wie ich in Kapitel 8 dargestellt habe, wird im Workflow die Sicherheit aller Objekte verwaltet. Das geschieht, indem der Workflow die aktuellen Rechte eines Objekts ändert. Eben habe ich Ihnen gezeigt, wie Sie sich die Sicherheitseinstellungen aller Objekte ansehen können, d.h., Sie können nun sehen, dass sich die Sicherheitseinstellungen von Objekten in einem Zustand von denen in einem anderen Zustand unterscheiden können. Wenn Sie auf PORTAL_WORKFLOW klicken, den CONTENTS-Reiter wählen, auf PLONE_WORKFLOW und dann den STATES-Reiter klicken, werden Sie alle verfügbaren Zustände sehen. Klicken Sie auf einen Zustand, und wählen Sie dann PERMISSIONS, um die Rechte für diesen Zustand zu sehen (siehe Abbildung 9.13).

Wenn ein Objekt in den Zustand *Veröffentlicht* kommt, erhalten anonyme Benutzer, wie Sie sehen, die Rechte *Access contents information* und *View*. Das heißt, Leute können den Inhalt sehen. Sie werden bemerken, dass Mitglieder oder Besitzer ihren eigenen Inhalt nicht bearbeiten können, weil sie dieses Recht nicht haben. Die vom Workflow angewendeten Rechte werden im PERMISSIONS-Reiter gesetzt, wo Sie alle vom Workflow verwalteten Rechte setzen können.

Nachdem Sie die Sicherheitseinstellungen geändert haben, müssen Sie zum Werkzeug PLONE_WORKFLOW gehen und UPDATE SECURITY SETTINGS klicken, sonst sind Sicherheitseinstellungen und Workflow des Objekts verschieden.

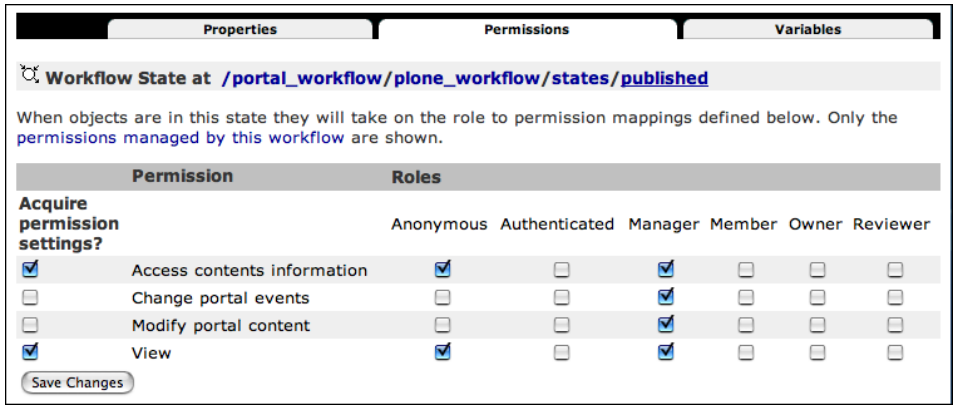


Abbildung 9.13: Rechte für den Zustand Veröffentlicht

Hinweis

Weil sich die Rechte ändern, wenn das Objekt in einen anderen Zustand übergeht, werden andere Rechteänderungen am Objekt, die Sie evtl. über das ZMI veranlassen, entfernt, wenn diese Rechte vom Workflow verwaltet werden (und nur dann). Aus diesem Grund sollten Sie immer dem Drang widerstehen, kleine Änderungen im ZMI an der Sicherheit von Inhaltstypen vorzunehmen. Bleiben Sie dabei, das Plone-Site-Objekt und den Workflow zu ändern.

Wächter

Zu allen Übergängen gibt es Wächter, mit denen der Administrator die erlaubten Rechte auswählen kann, bevor ein Benutzer den Übergang ausführen darf. Die Prüfung, ob ein Benutzer den Übergang ausführen darf, findet in dieser Reihenfolge statt: erst die Rechte, dann die Rollen und dann der Ausdruck. Sobald eine dieser Prüfungen positiv ist, wird der Übergang ausgeführt.

Es folgen die Einstellungen eines Wächters:

- **Permission**: Eine Liste der erlaubten Rechte, mit Semikola (;) voneinander getrennt, z.B. *Review portal content; Modify portal content*.
- **Roles**: Eine Liste von erlaubten Rollen bei diesem Übergang, mit Semikola voneinander getrennt, z.B. *Manager; Reviewer*.
- **Expression**: Ein Workflow-TALES-Ausdruck (Template Attribute Language Expression Syntax), mit dem Sie eine bestimmte Bedingung ausdrücken können. Der folgende Übergang z.B. findet nur dann statt, wenn er im Ordner namens `Members` ist. Das ist kein echtes Recht, aber es ist ein netter Trick:

```
python: if 'Members' in state_object.getPhysicalPath()
```

Hinweis

`getPhysicalPath` ist eine Methode bei allen Objekten in Zope, die den Ort innerhalb der Zope-Objekthierarchie zurückgibt, wobei ein eventuelles virtuelles Hosting ignoriert wird.

9.3.4 Proxy-Rollen

Im vorigen Kapitel habe ich einige nette Methoden vorgestellt, um Benutzer zu benachrichtigen und um Inhalte zu verschieben, die von einem Workflow erfasst sind. Wenn so etwas passiert, wird das Script dann ausgeführt, wenn der Benutzer die Workflow-Transaktion ausführt. In diesem Fall könnte Ihr Script etwas tun, wozu Ihr Benutzer ein Recht hat oder auch nicht. Vielleicht möchten Sie z.B. nicht, dass ein Benutzer etwas in einen Ordner namens `public` hinzufügen darf, außer über den Workflow. Das stellt ein Problem dar, denn dann müssen Sie sicherstellen, dass das Script mit einer übergeordneten Rolle ausgeführt werden kann.

Eine *Proxy-Rolle* ist etwas, womit Ihre Benutzer nicht in Berührung kommen, und wahrscheinlich werden sie auch nichts davon wissen, aber für Sie ist es eine Methode, dieses Problem zu umgehen. Angenommen, Sie möchten z.B., dass ein Benutzer einen anderen Benutzer aus der Liste aller Site-Benutzer finden kann. Sie möchten dem Benutzer nicht das Recht geben, alle Benutzer anzuzeigen, sondern nur dazu, die Benutzer in diesem bestimmten Kontext aufzulisten. Um das Script auszuführen, braucht ein Benutzer das Recht *List portal members*, damit er eine Mitgliederliste bekommt, aber das möchten Sie nicht an anonyme Benutzer vergeben.

Das Script, das diesen Befehl ausführt, muss eine übergeordnete Proxy-Rolle bekommen, wahrscheinlich *Member*. Gehen Sie dazu im ZMI zu dem Script, klicken Sie auf den PROXY-Reiter und dann auf MEMBER. Falls dieses Script als Datei im Dateisystem vorliegt, kann diese Information in der Metadaten-Datei hinzugefügt werden. Die `.metadata`-Datei hätte z.B. die folgende Zeile: `proxy = Member`. Nun würde dieses Script als Mitglied ausgeführt werden, wodurch Ihr Sicherheitsproblem gelöst ist!

9.4 Scripten von Benutzern

Jetzt haben Sie also eine Menge Benutzer in Ihrer Plone-Site, und daher benötigen Sie einige Scripts auf dieser Site, um Ihnen die Verwaltung der Benutzer zu erleichtern. Ab einigen hundert Benutzern kann es sehr schwer werden, Änderungen über das Web vorzunehmen, daher enthalten die folgenden Abschnitte ein paar Beispielscripts, die einige wichtige Aufgaben erledigen.

9.4.1 Benutzer en masse registrieren

Wenn Sie eine große Anzahl von Benutzern registrieren müssen, benötigen Sie ein Script, um sie zu importieren. Diese Benutzer können aus einem beliebigen System stammen, das Sie mit Plone ersetzen. Wenn Sie die Benutzer allerdings schon in LDAP, einer relationalen Datenbank oder einer anderen externen Quelle haben, können Sie diese Benutzer von dort aus direkt integrieren.

Vorläufig nehmen Sie eine Anzahl von mit Kommata getrennten Benutzern in einer Datei mit folgendem Inhalt: Benutzername, vollständiger Name, E-Mail und Gruppen. In diesem Beispiel werden Sie durch diese Liste durchgehen, um alle Benutzer mit diesen Einstellungen hinzuzufügen und dann ihre Eigenschaften so zu verändern, dass sie die richtigen Einstellungen haben. Die `.csv`-Datei sieht daher etwa wie folgt aus:

```
"Benutzername", "Vollständiger Name", "E-Mail", "Gruppen"
"Andy", "Andy Mckay", "andy@enfoldsystems.com", "Systeme,Vertrieb,Entwicklung"
...
```

Eine `.csv`-Datei enthält pro Zeile Werte, die mit Kommata voneinander getrennt sind, und kann mit den meisten Tabellenkalkulationsprogrammen erstellt und bearbeitet werden, darunter auch Microsoft Excel oder OpenOffice.org. Anschließend können Sie die Datei im CSV-Format exportieren und schließlich in Plone importieren. Weil dazu eine Menge von Methoden aufgerufen werden müssen, deren Nutzung nicht für alle Benutzer möglich ist, müssen Sie deswegen eine externe Methode daraus machen:

```
# An external method to import user
import csv

# the full path to your csv file
fileName = "/var/zope.zeo/Extensions/test.csv"

def importUsers(self):
    reader = csv.reader(open(fileName, "r"))
    pr = self.portal_registration
    pg = self.portal_groups
```

```
    out = []

    # if your csv file contains a header line that
    # explains the contents of each column
    ignoreLine = 1
```

Dies ist nur der Code zum Einstellen. Mit anderen Worten: Er stellt alle Variablen ein, die Sie in diesem Script benutzen werden. Am Anfang importieren Sie das `csv`-Modul, das in Python 2.3 enthalten ist und ein schnelles Parsen von `.csv`-Dateien ermöglicht. Die `.csv`-Datei steht in der Variablen `fileName` und ist der vollständige absolute Pfad zur Datei. Wenn Sie einen relativen Pfad angeben, sucht Plone die Datei möglicherweise am falschen Ort. Wie Sie vorher gesehen haben, wird `self` an die Methode übergeben, und von dort aus können Sie zu den zwei benötigten Werkzeugen gelangen: `portal_registration` für den Zugriff auf die Registrierungs-API und `portal_groups` für den Zugriff auf die Gruppen-API:

```
for row in reader:
    if ignoreLine:
        ignoreLine = 0
        continue

    # check we have exactly 4 items
    assert len(row) == 4
    id, name, email, groups = row
    groups = groups.split(',')

    # make a password
    password = pr.generatePassword()
```

Als Nächstes gehen Sie alle Zeilen durch und bekommen ID, Namen, E-Mail und Gruppen. Dann generieren Sie ein zufälliges Passwort, indem Sie `generatePassword` aufrufen. Dabei wird ein zufälliges Passwort aus sechs Zeichen (Klein- und Großbuchstaben und Zahlen) erzeugt. Wenn Sie die ID oder das Passwort von einer Information wie dem Benutzernamen oder der E-Mail abhängig machen möchten, können Sie das hier tun. In diesem Fall habe ich alle Gruppen mit Kommata getrennt ins gleiche Feld eingetragen (z.B. `"Vertrieb,Marketing"`). Daher muss ich das z.B. wie folgt in eine Liste einzelner Namen trennen:

```
    try:
        # add in member
        pr.addMember(id = id,
            password = password,
            roles = ["Member",],
            properties = {
                'fullname': name,
                'username': id,
```

```
                'email': email,
                }
            )
        # groups are separated by commas
        for groupId in groups:
            group = pg.getGroupById(groupId)
            group.addMember(id)

        out.append("Added user %s" % id)

    except ValueError, msg:
        # if we skipped this user for a reason, tell the person
        out.append("Skipped %s, reason: %s" % (id, msg))

# return something
return "\n".join(out)
```

Da Sie nun alle für die Registrierung des Benutzers nötigen Angaben haben, können Sie die eigentliche Registrierung vornehmen. Das machen Sie, indem Sie die Funktion `addMember` von `portal_registration` aufrufen, die einen Benutzer registriert. Dieser Funktion wird ein Dictionary mit Schlüssel/Wert-Paaren wie E-Mail und Name übergeben. Dann rufen Sie für jede Gruppe `getGroupById` auf, um die Gruppe zu bekommen und darauf `addMember` aufzurufen. Wie der Name schon andeutet, wird dabei der Benutzer mit dieser Gruppe registriert. Wenn Sie fertig sind, müssen Sie nur noch etwas für die Person ausgeben, die den Import ausführt.

Um das auf Ihrer Site auszuführen, müssen Sie das Script in das Verzeichnis `Extensions` Ihres Plone-Servers kopieren und es `import_users_with_groups.py` nennen. Dann müssen Sie von Hand die Gruppen hinzufügen, die Sie auf Ihrer Site haben werden. Dieses Script erzeugt nicht die Gruppen für Sie. Dann bereiten Sie die `.csv`-Datei vor. Wenn Ihre Benutzer in einem anderen System gespeichert sind, müssen Sie einen Weg finden, sie in dieses Format zu übertragen. Ändern Sie den Dateinamen im Script auf den Namen Ihrer neuen Datei. Fügen Sie dann eine externe Methode mit den folgenden Werten zu Ihrer Plone-Site hinzu:

- **ID**: `import_users_with_groups`
- **Module name**: `import_users_with_groups`
- **Function name**: `importUsers`

Nachdem Sie diese externe Methode hinzugefügt haben, klicken Sie auf TEST, um die Methode auszuführen, und Sie werden das Ergebnis bekommen!

9.4.2 Benutzereinstellungen ändern

Wenn Sie ein neues Produkt installieren oder eine Einstellung neu vornehmen, kann es notwendig sein, Benutzer-Metadaten en masse zu verändern. Wenn Sie z.B. einen neuen WYSIWYG-Editor installieren und möchten, dass dieser zum Standardeditor für jeden Benutzer wird, müssen Sie zwei Dinge tun:

Ändern Sie die Standardeinstellung für alle neuen Benutzer. Dazu klicken Sie auf PORTAL_METADATA und wählen den PROPERTIES-Reiter. Setzen Sie dort die Voreinstellung, und alle neuen Benutzer erhalten diesen Wert.

Ändern Sie die Einstellungen für alle vorhandenen Benutzer. Das kann nur mit der folgenden externen Methode durchgeführt werden:

```
def fixUsers(self):
    pm = self.portal_membership
    members = pm.listMemberIds()

    out = []
    for member in members:
        # now get the actual member
        m = pm.getMemberById(member)
        # get the editor property for that member
        p = m.getProperty('wysiwyg_editor', None)

        out.append("%s %s" % (p, member))
        if p is not None and p != 'Epoz':
            m.setMemberProperties({'wysiwyg_editor': 'Epoz',})
            out.append("Changed property for %s" % member)
    return "\n".join(out)
```

Kopieren Sie diesen Code in ein Python-Modul ins `Extensions`-Verzeichnis Ihrer Plone-Instanz, und nennen Sie das Modul `fixUserScript.py`. Fügen Sie dann im ZMI eine externe Methode mit den folgenden Parametern hinzu:

- **ID**: `fixUsers`
- **Module name**: `fixUserScript`
- **Function name**: `fixUsers`

Klicken Sie auf den TEST-Reiter, um den Code auszuführen. Er läuft über alle Mitglieder Ihrer Site und setzt den Wert des WYSIWYG-Editors auf `"Epoz"`. Dazu wird zuerst die Liste aller Mitglieder geholt, und zwar mit einer Methode in `portal_membership` namens `listMemberIds`, die das für Sie erledigt. Für jedes Mitglied wird die von Plone verwendete Eigenschaft untersucht, um den Editor zu bestimmen (in diesem Fall die Eigenschaft `wysiwyg_editor`). Falls diese Eigenschaft verschieden von `"Epoz"` ist, wird `setMemberProperties` aufgerufen, um sie zu ändern.

Das ist eine hilfreiche Methode, um über all Ihre Mitglieder zu iterieren. Mit den Methoden `setMemberProperties` und `getProperty` können Sie eine beliebige Eigenschaft eines Mitglieds untersuchen oder ändern.

9.4.3 Die anderen Benutzer einer Gruppe bestimmen

Ich habe bereits die Möglichkeit erörtert, eine E-Mail an alle Leute einer Arbeitsgruppe zu einem Objekt zu schicken. Das könnten Sie zum Workflow hinzufügen, aber zuerst brauchen Sie ein Script, um das zu tun. Dieses Beispiel verwendet ein paar Funktionen, um an die Benutzer heranzukommen. Das folgende Script `getGroupUsers` nimmt ein Objekt und gibt eine Liste von Benutzern zurück:

```
##parameters=object=None
# object is the object to find all the members of the same group for
users = []
# get the creator
userName = object.Creator()
user = context.portal_membership.getMemberById(userName)
pg = context.portal_groups

# loop through the groups the user is in
for group in user.getGroups():
  group = pg.getGroupById(group)

   # loop through the users in each of those groups
  for user in group.getGroupUsers():
    if user not in users and user != userName:
      users.append(user)

return users
```

In diesem Script erhalten Sie ein Objekt, dessen Erzeuger Sie mit der Methode `Creator` finden müssen. Sobald Sie diesen Benutzer haben, können Sie `getGroups` aufrufen, und eine Methode des Benutzerobjekts listet die Namen aller Gruppen auf, in denen der Benutzer Mitglied ist. Danach erhalten Sie all diese Gruppen, und aus dieser Liste erhalten Sie die Benutzernamen zu einer Gruppe. So haben Sie schließlich alle Benutzernamen. Nun möchten Sie darunter keine Duplikate und auch nicht die ursprüngliche Person, die das Objekt geändert hat. Die Benutzerliste enthält alle anderen Benutzer in den gleichen Gruppen wie die Person, die das Objekt besitzt.

Das sollten Sie in Ihr Workflow-E-Mail-Benachrichtigungsscript aus Kapitel 8 einsetzen, um es zu verbessern. Wenn Sie sich erinnern, dann haben Sie darin Folgendes gemacht:

```
for user in mship.listMembers():
    if "Reviewer" in mship.getMemberById(user.id).getRoles():
```

Das iteriert über alle Benutzer und prüft, ob sie die Mitgliederrolle haben. Das vorangegangene Script hieß `getGroupUsers` und befand sich im Ordner `portal_skins/custom`, d.h., Sie können per Akquisition über den `context`-Namespace darauf zugreifen. Kurz gesagt: mit `context.getGroupUsers(object)` bekommen Sie die Benutzer:

```
users = context.getGroupUsers(object)
for id in users:
    user = mship.getMemberById(id)
```

Nun senden Sie eine E-Mail an alle in der Gruppe statt an alle Redakteure!

9.4.4 Benutzerangaben in Page Templates

In Kapitel 6 haben Sie ein Page Template erstellt, mit dem ein Benutzer Feedback über ein Formular an den Site-Administrator geben konnte. In diesem Formular konnte der Benutzer in einem Eingabefeld eine E-Mail-Adresse eingeben, die Sie dann validiert haben. Wenn ein Benutzer aber angemeldet ist und Sie seine E-Mail-Adresse kennen, wäre es nett, dieses Feld für den Benutzer automatisch auszufüllen.

Der vorhandene Code für das Eingabefeld lautet wie folgt:

```
<input type="text" name="email_address"
       tal:attributes="tabindex tabindex/next;
                       value request/email_address|nothing" />
```

Wenn nun im Anfrageformular schon ein Wert für diese E-Mail-Adresse aus einem vorangegangenen Ausfüllversuch existiert, sollten Sie ihn anzeigen. Wenn nicht, dann können Sie nachsehen, ob eine E-Mail-Adresse zum aktuellen Benutzer existiert. Die folgenden Änderungen am Formular sorgen dafür, dass die E-Mail-Adresse ausgefüllt ist:

```
<input type="text" name="email_address"
tal:define="user context/portal_membership/getAuthenticatedMember;
            email user/email|nothing"
       tal:attributes="tabindex tabindex/next;
                       value request/email_address|email|nothing" />
```

9.4.5 Fehlersuche und Hintergründe zur Sicherheit

Ich habe festgestellt, dass Sicherheit nicht nur einer der am schwersten zu verstehenden Bereiche von Plone ist, sondern auch einer, bei dem die Fehlersuche und das Testen sehr schwierig sind. Da das Modell granular aufgebaut und kompliziert ist, kann es extrem schwierig sein herauszufinden, warum und wo ein Fehler auftritt. Manchmal ist die angegebene Fehlermeldung oder Information schwer zu entziffern, oder es ist schon schwer, überhaupt eine Information darüber zu bekommen.

Sicherheitstests sind ebenfalls ein schwieriges Unterfangen, da Sie in Sites mit vielen Rollen einen vollständigen Regressionstest für alle Rollen in allen Situationen durchführen sollten. Wegen der damit verbundenen Kosten werden diese umfangreichen Regressiontests aber oftmals nicht durchgeführt. Außerdem ist ein Sicherheitsfehler das Schlimmste, das auf einer Site passieren kann, wenn dadurch vertrauliche Informationen verloren gehen. Plone lässt Sie hier machen, was immer Sie wollen, und hält Sie nicht davon ab, sich selbst ins Knie zu schießen, also passen Sie auf!

VerboseSecurity

VerboseSecurity ist ein Zusatzprodukt, das von den Installationsprogrammen gleich mitinstalliert wird. Sie können VerboseSecurity auch unter *http://hathaway.freezope.org/Software/VerboseSecurity* herunterladen. Wie der Name schon klarmacht, bietet es detaillierte Fehlermeldungen, wenn Sie etwas in Plone nicht tun dürfen, weil Sie keine Berechtigung dazu haben. Wenn Sie jedoch zu lasche Sicherheitseinstellungen benutzen, kann Ihnen dieses Produkt auch nicht helfen.

VerboseSecurity kann auf einem Plone-Server ohne Performance-Einbußen laufen, d.h., Sie können das Produkt problemlos auf Ihren Produktions- und Entwicklungsservern betreiben. Ein bisschen Leistung mag dann verloren gehen, wenn jemand kein Recht hat, etwas zu tun, ein Fehler ausgelöst wird und die neuen Sicherheitsmodule einspringen.

Da die Fehlermeldung aber sehr detailliert ist, möchten Sie diese sicher nicht den Benutzern vorsetzen. Sie enthält wesentlich mehr Informationen über Ihr System, als ein Benutzer jemals wissen sollte! Passwörter sind jedoch niemals enthalten, sondern nur Informationen über die Benutzer, Rollen und Rechte. Natürlich wird Ihr Produktionsserver immer perfekt laufen, d.h., Sie werden keinen Bedarf haben, VerboseSecurity auf Ihrem Produktionsserver zu installieren.

Die ursprüngliche Implementierung der Routinen, die die Rechte prüfen, war in Python geschrieben. Mit zunehmend stabilerem API und nachdem die Entwickler die Einsicht gewonnen hatten, welchen Overhead die Sicherheitsmaßnahmen mit sich brachten, wurde sie in C neu geschrieben. Standardmäßig läuft die

schnellere C-Implementierung, was aber bedeutet, dass VerboseSecurity das Rechte-Modul nicht ändern kann, um es ausführlicher zu machen. Allerdings musste ich mich nur selten auf diese Detailstufe begeben. Normalerweise bekomme ich so schon genügend Informationen. Wenn Sie jedoch weitere Informationen benötigen, müssen Sie Plone mit der folgenden Umgebungsvariablen ausführen:

```
ZOPE_SECURITY_POLICY=PYTHON
```

Um VerboseSecurity zum Laufen zu bringen, müssen Sie lediglich sicherstellen, dass VerboseSecurity in Ihrem `Products`-Verzeichnis liegt (weitere Details dazu finden Sie in Kapitel 10) und dann Plone neu starten. Gehen Sie zum `cookie_authentication`-Objekt, das die Liste der Optionen für Ihre Site-Authentifikation enthält, und ändern Sie im Formular die Option für `login_page` von `require_login` auf einen leeren Wert, wie in Abbildung 9.14 zu sehen ist.

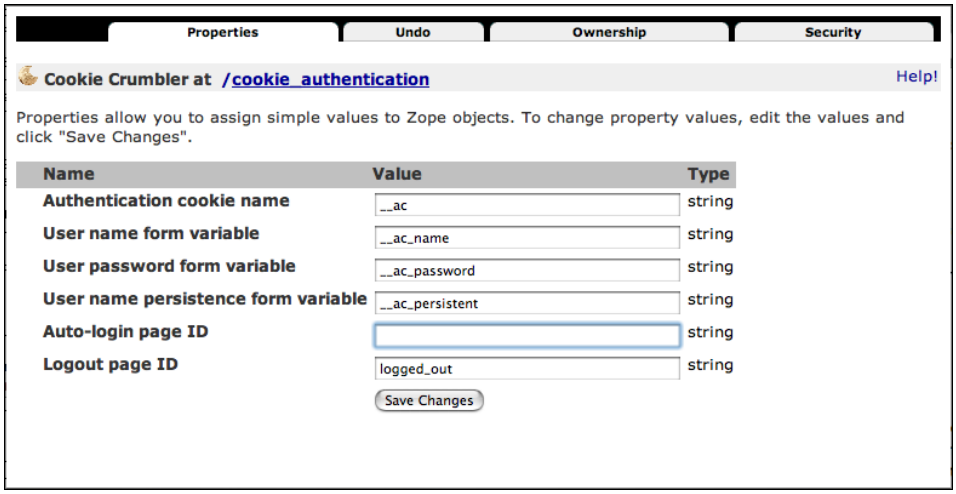


Abbildung 9.14: Ändern der Anmeldeeinstellungen für Ihre Site

Nun können Sie die Umstände neu wiederherstellen, unter denen der Fehler aufgetreten ist, den Sie suchen. *Denken Sie daran, sich als der Benutzer anzumelden, der den Fehler bekam!* An dieser Stelle ist es sehr praktisch, zwei verschiedene Browser für den Zugriff auf Ihre Plone-Site zu haben: einen für die Verwaltung und einen zum Testen. Wenn ein Fehler auftritt, wird ein HTTP-Authentifikationsdialog auf dem Bildschirm angezeigt. An diesem Punkt klicken Sie auf ABBRECHEN, und nun sollten Sie eine detaillierte Fehlermeldung bekommen, wie sie in Abbildung 9.15 zu sehen ist.

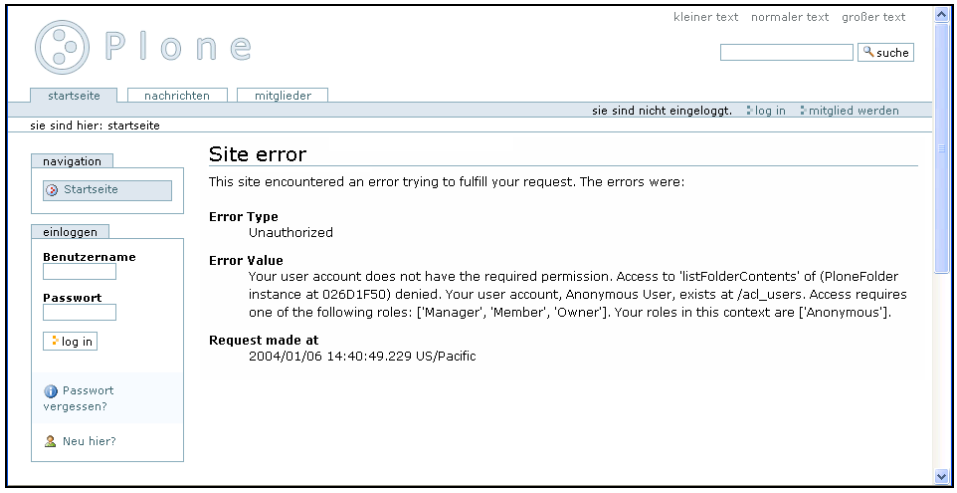


Abbildung 9.15: Eine schön detaillierte Fehlermeldung

Diese Meldung ist recht lang und selbsterklärend. An dieser Stelle gehe ich normalerweise zu dem anderen Browser und untersuche die Rechteeinstellungen der beteiligten Objekte, um zu sehen, was die Ursache sein könnte.

Häufige Probleme

Ein paar Probleme lassen sich leicht feststellen, wenn man es mit Plone zu tun hat. Das erste hat nichts speziell mit Plone zu tun, sollte hier aber dennoch wiederholt werden: Prüfen Sie, dass der Benutzer, der den Fehler erzeugen kann, wirklich der ist, den Sie annehmen. Ich habe oft genug von Leuten Dinge gehört wie »Funktioniert in einem Browser, aber nicht in einem anderen.« Das kommt normalerweise daher, das Sie beim Browser-Wechsel auch den Benutzer wechseln.

Um mit dem Offensichtlichen weiterzumachen, überprüfen Sie zweimal, dass Ihr Benutzer auch die erwartete Rolle hat! Das kann bedeuten, in `acl_users` nachzusehen, welche Benutzerrolle er hat, und zu prüfen, dass die erwartete Rolle vorhanden ist. Als Nächstes denken Sie an mögliche Gruppen, in denen der Benutzer enthalten sein könnte. Wieder verschafft Ihnen ein Blick in `acl_users` Klarheit, da Benutzer über eine Gruppe zu weiteren Rollen kommen können. Und denken Sie schließlich daran, dass die Rolle eines Benutzers auch von lokalen Rollen in Ordnern oder Objekten verändert werden kann. Das ist etwas schwieriger festzustellen, weil es keine einfache Möglichkeit gibt festzustellen, welcher Ordner oder welche Objekte lokale Rollen haben.

Sobald Sie sich sicher sind, wer der Benutzer ist und welche Rolle er an einem Objekt hat, können Sie herausfinden, welche Rechte an einem Objekt wirklich

vorhanden sind. Wie Sie gesehen haben, können zwei ähnliche Objekte (z.B. zwei Dokumente) verschiedene Rechte und Rollen haben. Der Benutzer, der das Dokument erstellt, wird auf dem Dokument die Rolle des Besitzers haben, während ein anderer Benutzer nur die Mitgliederrolle daran haben wird. Die Rechte an einem Dokument werden auch vom Workflow verändert, während es durch die verschiedenen Workflow-Zustände geht.

Plone zusperren

Plone kann man nicht so leicht zusperren, weil es kein Konzept einer »gesperrten« Site gibt. Das Grundprinzip ist aber, dass Benutzer das minimal Notwendige dessen tun können sollen, was sie tun müssen, und nicht mehr. Das heißt, Sie sollten die Standardeinstellungen überprüfen und die nicht benötigten Einstellungen entfernen.

Wirklich paranoide Zeitgenossen können auch Eigenschaften aus der Benutzerschnittstelle entfernen, um Benutzer vom Herumwandern abzuhalten, indem sie den CSS-Code verändern (Cascading Style Sheets). Aber denken Sie daran, dass allein das Entfernen eines Reiters zu einer Aktion oder die Verweigerung des Zugriffs an einem Page Template nicht ausreicht, wenn ein Benutzer z.B. weiterhin ein Dokument bearbeiten kann. Mit dem entsprechenden Wissen über Plone könnten Benutzer die Seite aus einem Script oder über einen anderen bösartigen Mechanismus ausführen. Wenn Sie es nur intensiv genug versuchen, werden Sie sehen, dass Sie in Plone zur Bearbeiten-Seite eines Dokuments kommen, das Sie gerade anzeigen, indem Sie die entsprechende URL hacken. Allerdings wird es Ihnen nicht gelingen, die Seite wirklich zu bearbeiten; Sie können lediglich das Bearbeiten-Formular aufrufen.

Wenn Ihr Server ohne Zugangsbeschränkung in der Wildnis läuft, sollten Sie sicherstellen, dass Sie einen weiteren Webserver vor Zopes ZServer laufen lassen. Wie ich in Kapitel 10 beschreiben werde, ist der im Paket enthaltene ZServer eine einfache Implementierung ohne all die Überprüfungen und Sicherheiten, die ein echter Webserver braucht. Erwägen Sie nach Möglichkeit den Einsatz von Proxies für andere Zope-Services wie FTP und WebDAV, falls Sie Benutzern den Zugriff auf diese Dienste ermöglichen, denen Sie nicht trauen können (was normalerweise nicht der Fall ist).

9.5 Plone mit anderen Diensten integrieren

Die folgenden Abschnitte behandeln die Sicherheit außerhalb einer Plone-Instanz, z.B. alle Sicherheitseinstellungen, die Sie benötigen, um Plone auf einem Server zu betreiben. Anschließend behandle ich den Einsatz von Plone mit LDAP, damit Sie Benutzer von einem externen Server in Plone benutzen können.

9.5.1 Sicherheit auf Ihrem Server

Die Sicherheit von Benutzern innerhalb eines Plone-Systems habe ich behandelt, aber es gibt einen weiteren wichtigen Aspekt: die Sicherheit und Einrichtung Ihres Plone-Servers in Ihrem Betriebssystem. Wie bei allen Webanwendungen ist es ein wichtiger Schritt, die Sicherheit Ihres Servers zu gewährleisten, bevor er für die ganze Welt freigeschaltet wird. Der Installationsvorgang für Zope 2.7 ist ziemlich gut und macht das meiste hiervon für Sie bereits richtig. Aber auf einige Dinge sollten Sie hingewiesen werden, was ich nun im Folgenden tun möchte.

Der Benutzer, der Zope laufen lässt

Sie sollten sicherstellen, dass der Benutzer, der Zope laufen lässt, ein Minimum der notwendigen Sicherheitsrechte hat, um die Aufgabe zu erledigen. Der Benutzer, der Zope ausführt, benötigt Lese- und Schreibrechte an allen Zope-Verzeichnissen im Dateisystem. Der Benutzer muss in den Verzeichnissen schreiben, die die Protokolle und die Datenbank Ihrer Zope-Instanz enthalten. Das sind die Verzeichnisse `var` und `log` Ihrer Zope-Instanz.

Auf Linux macht man das am besten so, dass ein spezieller Benutzer, z.B. mit Namen plone, erstellt wird, der das übernimmt. Dann können Sie die Zugriffsmöglichkeiten dieses Benutzers für den unwahrscheinlichen Fall beschränken, dass Plone gehackt wird.

Wenn Sie mit Plone unter Linux Ports mit kleinen Nummern (unter 1024) wie 21 oder 80 belegen möchten, müssen Sie Plone normalerweise als Root ausführen. Es belegt diese Ports dann als Root und wechselt anschließend zu einem anderen effektiven Benutzer. Dazu müssen Sie einen Wert für `effective-user` in der Konfigurationsdatei `zope.conf` angeben. Dann belegt es die Ports und wechselt zu diesem Benutzer. Ein Beispiel dafür ist `effective-user zope`. Die beste Alternative dazu ist nicht etwa die, das gar nicht zu machen, sondern Zope stattdessen auf einem höheren Port, z.B. 8080, auszuführen. Diesen Port können Sie dann in der Firewall schützen und Apache oder einen anderen Webserver benutzen, der auf Port 80 als Proxy zu Port 8080 läuft. Kapitel 10 enthält weitere Details zu diesem Thema.

Das Äquivalent dazu unter Windows ist der Benutzer, der den Dienst ausführt, standardmäßig das Konto `LocalSystem`. Auch hier können Sie den Benutzer ändern, unter dem Plone läuft. Wenn Sie versuchen, Plone auf einem Windows-Rechner auszuführen, der keine Dienste hat (was ich nicht empfehlen oder unterstützen kann), läuft Plone lokal unter dem Benutzer, der den Server manuell gestartet hat.

Manche Produkte verlangen eventuell die Installation zusätzlicher Software, die z.B. Möglichkeiten der Bildverarbeitung, Dokumentkonvertierung usw. bietet.

Wenn Sie solche Werkzeuge installiert haben, sollten Sie daran denken, dass eventuell ein wenig Arbeit damit verbunden ist, damit sie mit Ihrer Plone-Site erfolgreich zusammenarbeiten. Unter Windows habe ich z.B. `pdftohtml` für die Konvertierung von PDF-Dokumenten (Portable Document Format) installiert, aber damit der Befehl ausgeführt werden kann, musste ich den Dienst unter einem Benutzer mit zusätzlichen Privilegien ausführen, damit Zope mit dieser Software interagieren kann. In diesem Fall war das kein Problem, weil der Server hinter einer Firewall lag.

Zugriff in Notfällen

Wenn Sie eine Plone-Site haben, aber nicht auf das ZMI zugreifen können, weil Sie das Passwort nicht kennen oder vergessen haben, können Sie sich ein Notfall-Konto verschaffen. Dazu benötigen Sie den Zugriff per Dateisystem auf die Instanz Ihrer Plone-Site. Wenn Sie diesen nicht haben, müssen Sie zuerst einen Weg finden, diesen zu erhalten.

Gehen Sie dazu zu Ihrer Instanzwurzel, und starten Sie das Script `zpasswd.py`, das Sie in Ihrem Zope-Verzeichnis (in `ZOPE_HOME`) finden. Auf meinem Rechner befindet sich das Script `zpasswd.py` in `/opt/Zope-2.7/bin/zpasswd.py`. Um also das Passwort zu erzeugen, machen Sie Folgendes:

```
$ cd /var/zope
$ python /opt/Zope-2.7/bin/zpasswd.py access

Username: emergency
Password:
Verify password:

Please choose a format from:

SHA - SHA-1 hashed password
CRYPT - UNIX-style crypt password
CLEARTEXT - no protection.

Encoding: SHA
Domain restrictions:
```

Damit wird eine Datei namens `access` in Ihrer Zope-Instanz erzeugt. Starten Sie Zope nun erneut, und melden Sie sich im ZMI mit dem Benutzernamen und Passwort an, das in diesem Script eingegeben wurde. Dieser Benutzer hat eine besondere Bedeutung in Plone und wird auch *Notfallbenutzer* genannt. Wenn Sie einmal als Notfallbenutzer angemeldet sind, können Sie keine Objekte erstellen, aber Sie können einen neuen Benutzer für sich selbst anlegen und sich als dieser Benutzer anmelden. Aus Sicherheitsgründen sollten Sie dann die Datei `access` löschen.

Zugriff in Notfällen unter Windows

Die Windows-Installation von Plone bietet eine grafische Benutzerschnittstelle, mit der man sehr leicht einen Notfallzugriff bekommt. Wählen Sie START – PROGRAMME – PLONE – PLONE und klicken Sie auf EMERGENCY USER. Damit können Sie einen neuen Benutzer erzeugen, das Passwort des aktuellen Notfallbenutzers ändern oder einen Notfallbenutzer löschen, wie Abbildung 9.16 zeigt.

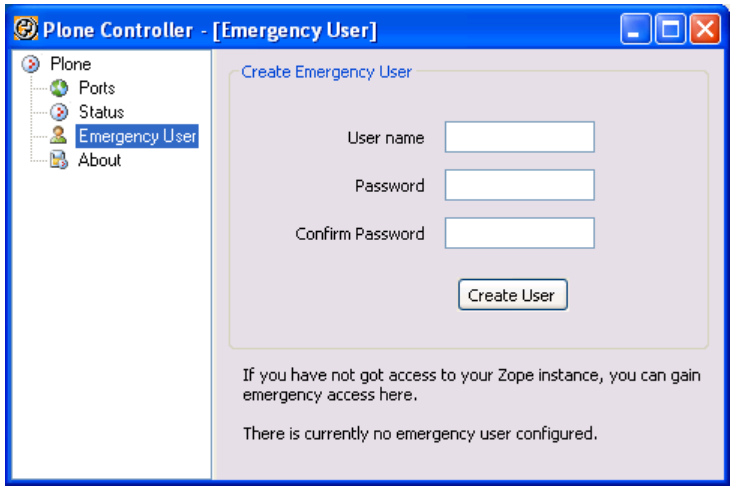


Abbildung 9.16: Erstellen eines neuen Notfallbenutzers

Um einen neuen Benutzer zu erstellen, klicken Sie auf CREATE USER. Geben Sie im dann erscheinenden Dialogfeld einen Benutzernamen und ein Passwort ein. Dann wird im Dateisystem eine Datei erzeugt, die den Benutzernamen und das Passwort enthält. Ein Klick auf CHANGE PASSWORD ändert das Passwort des Benutzers. Nach dem Erstellen oder der Änderung eines Passworts müssen Sie Zope neu starten. Um Plone neu zu starten, gehen Sie zum CONTROL-Reiter, klicken auf STOP und klicken dann auf START. Dann klicken Sie auf MANAGE ROOT und geben den zuvor gewählten Benutzernamen und Ihr Passwort ein. Dann sind Sie als Notfallbenutzer angemeldet, d.h., Sie können keine Objekte erstellen, aber Sie können einen neuen Benutzer für sich selbst erstellen und sich als dieser Benutzer anmelden.

9.5.2 Externe Authentifizierungssysteme verwenden

Wie Sie in Kapitel 8 gesehen haben, speichert Plone all seine Benutzer in der Zope-Objektdatenbank in einer eigenen Benutzerliste. Das ist, wie alles, nicht perfekt, und ab einem gewissen Punkt möchten Sie vielleicht einen anderen Dienst verwenden, um Ihre Benutzer zu authentifizieren. Das am weitesten ver-

breitete Alternativsystem ist LDAP oder Microsofts Active Directory, das mit LDAP kommuniziert.

Möglicherweise möchten Sie aber mit einer anderen Anwendung zusammenarbeiten, die ihre Benutzer in einer relationalen Datenbank speichert. Während ich dieses Buch schrieb, verwendete die ASPN-Site von ActiveState Zope für alle Inhalte, aber dessen Benutzer können sich auch mit Microsofts Passport-System authentifizieren. Zusätzliche Schemata für die Benutzerauthentifizierung zu installieren ist dank der exzellenten Arbeit vieler Entwickler sogar sehr einfach. Bei der Einrichtung dieses Vorgangs empfand ich das Übersetzen der Software und die Integration der Systeme als den schwierigsten Teil.

Achtung

Im nächsten Abschnitt werden Sie mit den `acl_users`-Ordnern einer Plone-Site herumspielen. Sie sollten niemals den `acl_users`-Ordner in der Wurzel Ihrer Zope-Instanz löschen oder ändern. Wenn Sie das tun und Ihr Benutzerordner geht aus irgendwelchen Gründen kaputt (z.B. wenn der Server herunterfährt), ist Ihre gesamte Site blockiert und Sie werden keinerlei Zugriff mehr darauf bekommen, nicht einmal als Administrator. Stellen Sie sicher, dass Sie nur den Benutzerordner der Plone-Site ändern!

LDAP verwenden

Zuerst müssen Sie einen LDAP-Server einrichten, oder etwas, das mit LDAP kommuniziert, z.B. Active Directory (obwohl Active Directory anscheinend einige Macken hat). In diesem Beispiel habe ich openLDAP auf meinem Red Hat-Server und auf meinem Windows-Server installiert. Für Windows finden Sie eine vorkompilierte Version unter *http://www.zope.org/Members/volkerw/LdapWin32*. Diese habe ich mit Python 2.3 getestet.

Laden Sie die Datei herunter, packen Sie sie aus, und kopieren Sie den Inhalt nach `C:\Programme\Plone\Python\lib\site-packages`. Dann installieren Sie *LDAP-UserFolder*.

Für Linux finden Sie die openLDAP-Downloads unter *http://www.openldap.org*. Die getestete Version enthält die RPMs 2.0.27-2.8.0 und 2.0.27-2.8.0. Nach dem Übersetzen mit den folgenden Anweisungen habe ich unter *http://python-ldap.sourceforge.net* die passenden LDAP-Bibliotheken für Python heruntergeladen und übersetzt. In meinem Fall war die getestete Version `2.0.0pre05-1.i386.rpm`. Passen Sie auf, dass Sie denselben Python-Interpreter verwenden, mit dem auch Plone läuft.

Wenn Sie durch diese Schritte durch sind, müssen Sie sicherstellen, dass das Modul `_ldap.so` von Python importiert werden kann. Am leichtesten prüft man das mit dem folgenden Befehl:

```
$ python -c "import _ldap"
```

Wenn Sie keine Fehlermeldungen erhalten, dann wurde es korrekt importiert. Wenn Sie doch Fehler bekommen, müssen Sie Ihre Schritte noch einmal durchgehen. Holen Sie sich dann *LDAPUserFolder* unter *http://www.dataflake.org/software/ldapuserfolder*. Die hier getestete Version war 2.1 beta 2. Laden Sie die Datei herunter, packen Sie sie aus, und verschieben Sie sie ins Verzeichnis `Products` Ihrer Zope-Installation. Beispiel:

```
$ tar -zxf LDAPUserFolder-2_1beta2.tgz
$ mv LDAPUserFolder /var/zope/Products
```

Nun starten Sie Plone neu, gehen zum Control Panel und vergewissern sich, dass das Produkt auf der Produktseite korrekt angezeigt wird. Weitere Details dazu gibt es in Kapitel 10.

Anschließend sollten Sie in Plone in der Lage sein, auf `acl_users` und SOURCES zu klicken und dann nach unten bis zur Option `Users source #1` zu scrollen. Wählen Sie dann *LDAPUserFolder*, und markieren Sie *I'm sure*, wie in Abbildung 9.17 gezeigt. Das erzeugt einen neuen Benutzerordner und ersetzt den vorhandenen. Passen Sie also auf, dass Sie nichts Wichtiges verlieren. Tatsächlich ist jetzt ein guter Zeitpunkt für ein Backup. Anschließend klicken Sie auf OK.

Abbildung 9.17: Hinzufügen eines LDAPUserFolder

Nehmen Sie in den Einstellungen zu *LDAPUserFolder* die Änderungen vor, die Ihren vorhandenen LDAP-Einstellungen entsprechen. Nun sollten Sie auf den USERS-Reiter klicken und nach Benutzern suchen können, die in Ihrem LDAP-Verzeichnis bereits existieren.

Relationale und andere Datenbanken

Ein hervorragender Ersatz für den Benutzerordner heißt *exUserFolder*, was für *extensible user folder* steht. Er ist sehr einfach zu installieren. Laden Sie ihn einfach unter *http://prdownloads.sourceforge.net/exuserfolder/exUserFolder-0_20_0.tgz* herunter, packen Sie ihn wie üblich aus, und kopieren Sie ihn in Ihr Products-Verzeichnis. Nach einem Plone-Neustart sollten Sie ACL_USERS anklicken können, SOURCES wählen und zur Option *Users source #1* scrollen. Dann wählen Sie *exUserFolder* und markieren *I'm sure*.

Tatsächlich führt *exUserFolder* eine Authentifizierung auf folgende Arten durch:

- Radius
- SMB
- LDAP
- Relationale Datenbanken

Um das machen zu können, müssen Sie die spezifischen Datenbankadapter für die relationale Datenbank installieren. Glücklicherweise sind Adapter für alle wichtigen Datenbanken vorhanden. Weitere Informationen zu fast allen Themen finden Sie in den `exUserFolder`-Verzeichnissen in den `ReadMe`-Dateien. Das Zope-Buch behandelt die Einrichtung des Zugriffs auf relationale Datenbanken unter *http://zope.org/Documentation/Books/ZopeBook/2_6Edition/RelationalDatabases.stx*.

10 Integration mit anderen Systemen

Integration ist ein großes Problem in allen Firmen, die bereits viele verschiedene Systeme verwenden. Da Plone ein Open Source-Projekt ist, existieren dafür eine Menge Produkte, Zusätze, Skins und Werkzeuge, die gratis zusätzliche Funktionalität bieten. Sie haben richtig gelesen, diese Produkte werden oftmals einfach so an alle abgegeben, die sie haben wollen. Dazu kommt noch, dass Python als Open Source-Sprache über eine Menge hervorragender Produkte verfügt (oft *Pakete* genannt). Die meisten dieser Produkte betreffen Plone allerdings nicht direkt. Mit anderen Worten, sie fügen keine Funktionalität von sich aus zu Plone hinzu. Das ist Aufgabe von Plone-Produkten. Allerdings fragen die Leute oft »Kann Plone dies oder jenes machen?« Die Antwort lautet oftmals: »Wenn Python es kann, ja!«

Hier ist eine Liste der beliebtesten Python-Produkte:

- **Python Imaging Library (PIL)**: Damit können Sie Bilder manipulieren, konvertieren und analysieren (*http://www.pythonware.com/products/pil/*).
- **ReportLab**: Hiermit können Sie dynamisch PDF-Dateien (Portable Document Format) erzeugen, die Bilder, Grafiken und andere nette Dinge enthalten (*http://www.reportlab.org/*).
- **Windows-Erweiterungen**: Damit haben Sie eine Schnittstelle zu allen Windows-APIs (Application Programming Interfaces), z.B. können Sie damit COM-Objekte (Component Object Model) benutzen (*http://sourceforge.net/projects/pywin32/*).
- **Pygame**: Das ist ein Framework, mit dem man Spiele in Python entwickeln kann. Einige Leute haben es in Plone benutzt, um an die Schnittstelle zu seinen Medienschichten für die Erzeugung von Bildern oder Sounds zu gelangen (*http://www.pygame.org/*).
- **OpenOffice.org-Anbindung**: Mit dieser Anbindung können Sie fast alles mit OpenOffice.org-Dokumenten machen, z.B. sogar Microsoft Office-Dokumente

parsen, wie Sie in Kapitel 13 sehen werden (*http://udk.openoffice.org/python/python-bridge.html*).

- **mxTidy**: Dieses Paket sucht und korrigiert Probleme in HTML-Dateien (Hypertext Markup Language), inklusive Page Templates (*http://www.egenix.com/files/python/mxTidy.html*).

Diese hervorragenden Zusatzprodukte verfügen normalerweise über grafische Installationsprogramme für Windows, mit denen Sie eine Installation schrittweise durchführen können. Wenn Sie kein Windows-Benutzer sind, dann bietet das Python-Modul *distutils* eine einfache Schnittstelle für die Installation dieser Produkte von der Kommandozeile. Wie immer gilt auch hier, dass man vor der Installation die Anweisungen im heruntergeladenen Paket lesen sollte, damit nichts schief gehen kann. Dieses Kapitel konzentriert sich auf die Installation von Produkten, die eine zusätzliche Funktionalität für Plone bieten. Ein Verzeichnis von Python-Paketen finden Sie unter *http://www.python.org/pypi*.

Exkurs: Open Source-Lizenzen

Die meisten Open Source-Pakete werden unter einer bestimmten Lizenz herausgebracht, die beschreibt, wie das Paket benutzt werden darf. Bevor Sie Code von Dritten benutzen, sollten Sie sich die Lizenz ansehen, um zu prüfen, ob sie zu der Art des geplanten Einsatzes bei Ihnen passt.

Plone wird unter der General Public License (GPL) lizenziert, die Sie unter *http://www.gnu.org/copyleft/gpl.html* und in der Datei `LICENSE.txt` in Ihrer Plone-Installation finden. Als Entwickler einer Plone-Website dürfen Sie Websites problemlos fröhlich drauflosentwickeln. Die Benutzer Ihrer Website werden sich nie Gedanken über die Lizenz des Codes machen müssen, sondern sie benutzen die Website ganz normal. Wie bei den meisten Lizenzen gilt auch hier, dass die Weiterverbreitung und der Verkauf von anderer Leute Code eingeschränkt werden soll.

Normalerweise sind Lizenzen leicht zu lesen und zu verstehen, aber wenn Sie sich dabei nicht wohlfühlen, sollten Sie die Lizenz lieber einem ausgebildeten Profi vorlegen. Ich beschränke mich hier darauf, die wichtigsten in der Zope-Welt gebräuchlichen Lizenzen zu beschreiben und Links zu weiteren Angaben dazu anzugeben:

- **Zope Public License:** Dies ist die Lizenz des Zope Application Servers sowie die Lizenz der Wahl für viele Zusatzprodukte, die für Zope geschrieben wurden. Eine Kopie dieser Lizenz finden Sie unter *http://zope.org/Resources/ZPL*.
- **Python License:** Das ist die Python-Lizenz, von der Sie eine Kopie unter *http://www.python.org/2.3/license.html* finden.
- **GPL:** Das ist die Lizenz von Plone, zu finden unter *http://www.gnu.org/copyleft/gpl.html*.
- **Lesser GPL:** Und schließlich die Lesser GPL, die unter *http://www.gnu.org/copyleft/lesser.html* zu finden ist.

10.1 Plone-Produkte installieren

Ein *Produkt* ist ein Modul, das in Plone installiert wird, um weitere Funktionalität für Plone bereitzustellen. Auch wenn der Name *Produkt* an Kosten denken lässt, ist das nicht richtig, denn die meisten Produkte sind frei verfügbar und Open Source. Der Begriff *Produkt* beschreibt tatsächlich etwas, was im Dateisystem vorhanden ist und an andere Plone-Sites weitergegeben und dort verwendet werden kann.

Die Installation eines Produkts erfolgt normalerweise in den folgenden zwei Schritten:

1. Installation zu seiner Registrierung innerhalb von Zope
2. Installation in jeder Plone-Instanz, die es benutzen soll

Wegen der großen Vielzahl an verfügbaren Zusätzen kann man nur sehr schwer allgemeinen Grundsätze angeben, was man genau zu deren Installation tun muss. Wie ich in diesem Kapitel mehrfach darstellen werde, sollten Sie immer die Installationsanweisungen der Produkte lesen, die normalerweise erklären, wie man das Produkt installiert. Wenn Sie weitere Hilfestellung brauchen, wenden Sie sich an eine Mailing-Liste oder an den Autor des Produkts, um weitere Informationen zu erhalten. Aber lesen Sie unbedingt zuerst die Anweisungen!

Denken Sie bei der Installation von Produkten daran, dass Sie Code installieren, der unvollständig sein kann und keinerlei Qualitätsgarantien macht. Es liegt in der Natur von Open Source, dass Leute Produkte schreiben und sie dann liegen lassen, wenn sie mit dem nächsten Projekt weitermachen. In einer idealen Welt würde eine Person Ihres Vertrauens sich den Code Zeile für Zeile anschauen, bevor Sie irgendetwas installieren. In der Realität ist das nicht machbar. Dennoch gilt, dass die meisten Produkte ziemlich gut sind. Aber auf jeden Fall sollten Sie Produkte testen, bevor Sie diese auf Ihrer Millionen Euro schweren Website installieren.

10.1.1 Produkte finden

Das richtige Produkt für Ihre Zwecke zu finden ist vermutlich der schwierigste Teil der Integration. Die Website Zope.org enthält viele Produkte, die von Benutzern erstellt und hochgeladen wurden. Diese Produkte finden Sie vor allem unter *http://www.zope.org/Products*, aber wenn Sie sich die Homepage von Zope.org anschauen, werden Sie Produktankündigungen im rechten Teil der Seite sehen. Manche dieser Produkte sind für Plone gedacht, andere für Zope, das CMF (Content Management Framework) oder Python.

Der andere wichtige Bereich zum Auffinden von Produkten ist das Collective-Projekt auf SourceForge (*http://sf.net/projects/collective*). Die dortigen Produkte befinden sich im CVS (Concurrent Versioning System) von SourceForge. Produkte werden zwar oftmals in Archivdateien verbreitet, aber CVS ist die beste Methode, auf diese Produkte hier zuzugreifen.

Zurzeit gibt es kein umfassendes Produktverzeichnis für Plone-Produkte oder deren Entwicklungszustand (aber ich hoffe, eines wird unter Plone.org online verfügbar sein, wenn dieses Buch erscheint). Bei der Veröffentlichung von Produkten stellen Leute ihre Pakete oftmals auf die DATEIEN-Seite, aber am sichersten ist es, ins CVS zu schauen. Eine visuelle Ansicht aller verfügbaren Dateien finden Sie unter *http://cvs.sourceforge.net/viewcvs.py/collective/*.

Ein letztes CVS-Repository mit nützlichem Code ist das von Zope Corporation. Fast jeder öffentlich gemachte Code wird in dieses CVS-Repository gestellt. Auch wenn Sie die Quellen zu Zope 2 suchen, ist das der beste Zielort für Sie. Das `Products`-Verzeichnis enthält alle Produkte (*http://cvs.zope.org/Products/*). Weitere Informationen darüber, wie Sie Code auschecken können, finden Sie unter *http://dev.zope.org/CVS/ReadOnlyAccess*.

Exkurs: Was ist CVS?

CVS ist ein System zur Versionskontrolle von Quellcode. Meistens findet die Entwicklung unter einem Versionskontrollsystem wie CVS oder einem seiner vielen ähnlichen Konkurenzprodukte statt, z.B. *Subversion*, *Perforce*, *BitKeeper* usw.

Das Auschecken von Dateien aus CVS ist einfach, und die meisten Unix- und Linux-Benutzer werden schon damit vertraut sein, CVS von der Kommandozeile aus zu benutzen. Machen Sie Folgendes, um alle Produkte aus dem Collective-Projekt auf Ihren Rechner auszuchecken (das kann eine Weile dauern):

```
cvs -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/collective
login
```

Geben Sie ein leeres Passwort ein, und fahren Sie mit folgendem Befehl fort:

```
cvs -z3 -d:pserver:anonymous@cvs.sourceforge.net:
/cvsroot/collective co *.*
```

Die meisten Windows-Benutzer im Plone-Entwicklerteam verwenden *TortoiseCVS*, das sich direkt in die Windows Explorer-Shell einklinkt, d.h., Sie können Code im Explorer mit einem Rechtsklick ein- und auschecken. Weitere Informationen zu TortoiseCVS finden Sie unter *http://www.tortoisecvs.org*.

10.1.2 Installation in Zope

Nachdem Sie ein passendes Produkt gefunden und heruntergeladen haben, müssen Sie es installieren. Zuerst müssen Sie es in Zope installieren, damit es von Zope als neues Produkt erkannt wird. Dazu müssen Sie den Bereich finden, in dem sich alle vorhandenen Produkte befinden. Um die Verzeichnisse zu finden, gehen Sie im ZMI (Zope Management Interface) zum Control Panel. Dort sehen Sie eine Liste der Verzeichnisse Ihrer Plone-Instanz. Wenn Sie einen Wert für `INSTANCE_HOME` haben, dann befindet sich Ihr `Products`-Verzeichnis in diesem Verzeichnis, sonst finden Sie das `Products`-Verzeichnis in `SOFTWARE_HOME`. Es sollte angemerkt werden, dass bei fast allen Installationsmethoden von Plone die Variable `INSTANCE_HOME` für Sie gesetzt wird. Wie Sie in Abbildung 10.1 sehen, ist der Wert für mein `INSTANCE_HOME` gleich `/var/book`, d.h., mein `Products`-Verzeichnis ist `/var/book/Products`.

Um den Zope-Anteil der Installation durchzuführen, packen Sie das heruntergeladene Produkt aus und kopieren es in das `Products`-Verzeichnis Ihres Servers. Tatsächlich ist das ein wenig trickreich und hängt sehr stark davon ab, wie das Produkt verpackt wurde. Um das ein wenig detaillierter zu zeigen, erklärt der folgende Abschnitt, wie man *CMFExternalFile* als Beispielprodukt installiert. CMFExternalFile wird seinerseits weiter unten im Abschnitt »Eine Datei in Plone verwalten« behandelt.

CMFExternalFile besitzt die nette Eigenschaft, dass es aus zwei Teilen besteht, die separat heruntergeladen werden. Zuerst haben Sie einen Zope-spezifischen Codeteil namens *ExternalFile*. Sollten Sie dieses Produkt jemals außerhalb von Plone, im normalen Zope, verwenden wollen, dann könnten Sie das tun. Und zweitens haben Sie einen Plone- und CMF-spezifischen Codeteil namens CMFExternalFile. Die meisten Produkte benötigen jedoch keine zwei Installationen, sondern bestehen aus einem einzigen Teil.

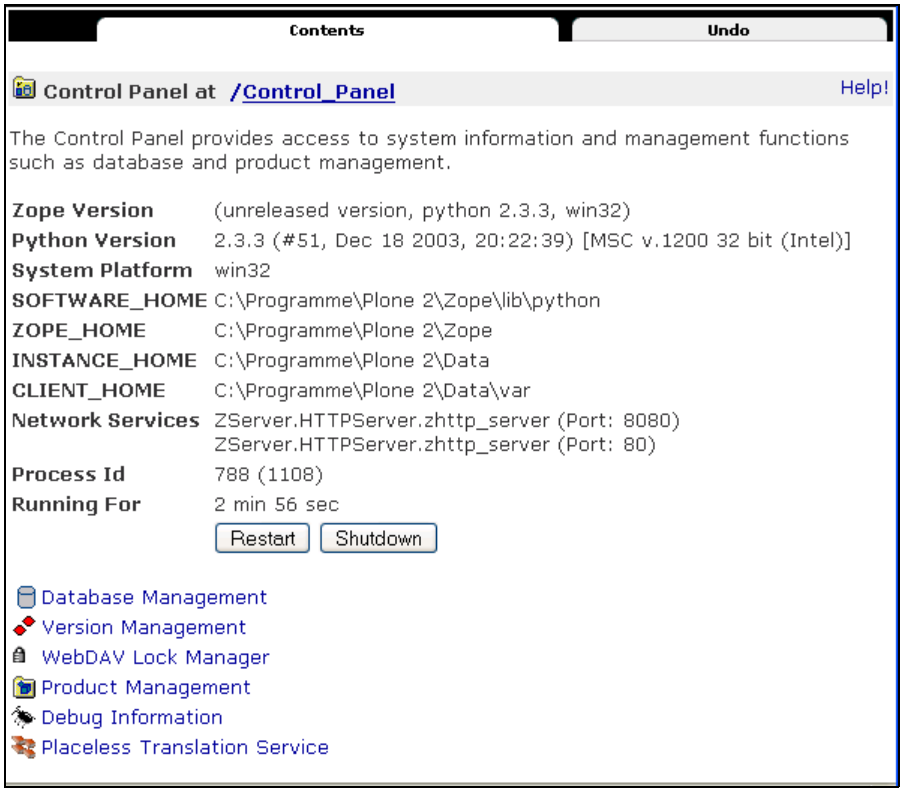


Abbildung 10.1: Suche nach Ihrem Products-Verzeichnis

Eine Beispielinstallation unter Windows durchführen

Zuerst müssen Sie das Produkt von Zope.org unter *http://zope.org/Members/ariel-partners/ExternalFile/1.2.0/ExternalFile-1-2-0.zip* herunterladen und auf Ihrem Rechner speichern.

Dann packen Sie die Datei aus. Dazu könnten Sie WinZip benutzen, das Sie heutzutage auf den meisten Windows-Rechnern finden (ich ziehe 7-Zip vor, das Sie unter *http://www.7-zip.org* finden).

Nach dem Auspacken erhalten Sie ein Verzeichnis namens `ExternalFile`. Darin befindet sich das Produktverzeichnis (siehe Abbildung 10.2). Sie erkennen es daran, dass sich in diesem Verzeichnis eine Menge Python- und Textdateien befinden, darunter auch `INSTALL.txt` und `README.txt` mit weiteren Informationen zur Installation.

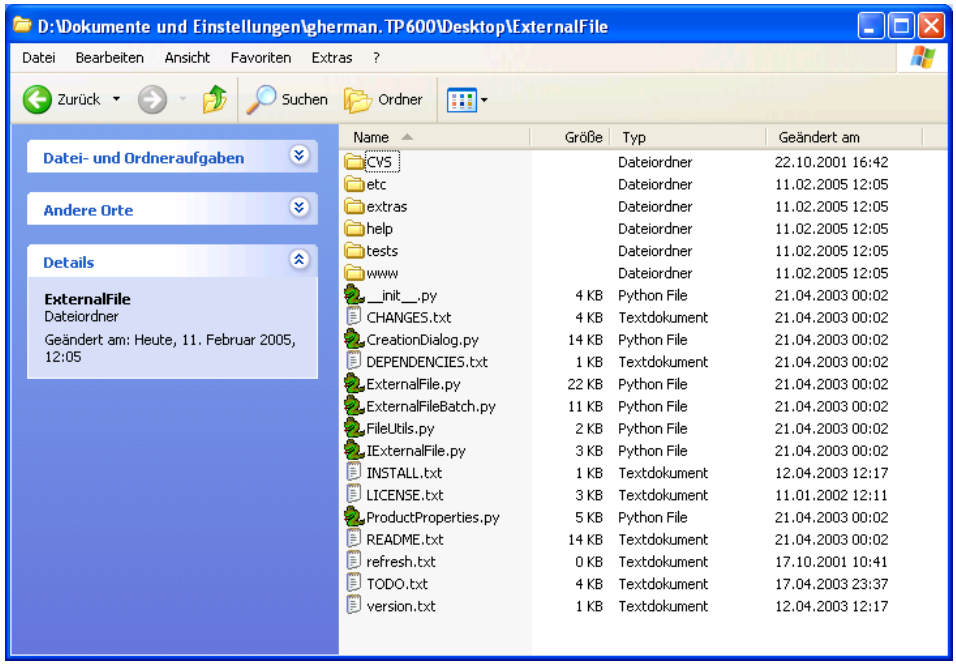


Abbildung 10.2: Der Inhalt des ExternalFile-Verzeichnisses

Als Nächstes verschieben Sie den `ExternalFile`-Ordner (nicht seinen Inhalt) in Ihr `Products`-Verzeichnis. Unter Windows befindet sich dieses Verzeichnis unter `C:\Programme\Plone 2\Data\Products`. In diesem Verzeichnis sehen Sie eine Reihe anderer Verzeichnisse, darunter `CMFPlone`, `CMFCore` usw. Das Verzeichnis `ExternalFile` sollte nun eines davon sein. Nun können Sie zum Abschnitt über das Testen der Installation auf dem Server springen.

Eine Beispielinstallation unter Unix durchführen

Zuerst müssen Sie das Produkt von Zope.org unter *http://zope.org/Members/arielpartners/ExternalFile/1.2.0/ExternalFile-1-2-0.zip* herunterladen und auf Ihrem Rechner speichern. Packen Sie dann die Datei aus. Auf den meisten Unix-Systemen ist solch ein unzip-Programm bereits installiert. Geben Sie dann die folgenden Befehle ein:

```
$ unzip ExternalFile-1-2-0.zip
Archive: ExternalFile-1-2-0.zip
   creating: ExternalFile/CVS/
...
```

Nach dem Auspacken erhalten Sie ein Verzeichnis namens `ExternalFile`. Darin befindet sich das Produktverzeichnis. Sie erkennen es daran, dass sich in diesem

Verzeichnis eine Menge Python- und Textdateien befinden, darunter auch `INSTALL.txt` und `README.txt` mit weiteren Informationen zur Installation.

Als Nächstes verschieben Sie den `ExternalFile`-Ordner (nicht seinen Inhalt) in Ihr `Products`-Verzeichnis. Dieser Befehl hängt von der Konfiguration Ihres Servers ab. In meinem Fall lautet er wie folgt:

```
$ mv ExternalFile /var/zope/Products
```

Die Installation auf dem Server testen

Nach der Installation eines Produkts müssen Sie Plone neu starten, damit das neue Produkt in Plone registriert wird. Nach dem Neustart Ihres Servers gehen Sie ins ZMI und zeigen den PRODUCT MANAGEMENT-Schirm des Zope-Control Panels an. Dieser Schirm listet alle auf dem Server installierten Produkte auf. Wenn Sie das Produkt erfolgreich installiert haben, wird es hier aufgeführt, wie Sie in Abbildung 10.3 sehen können.

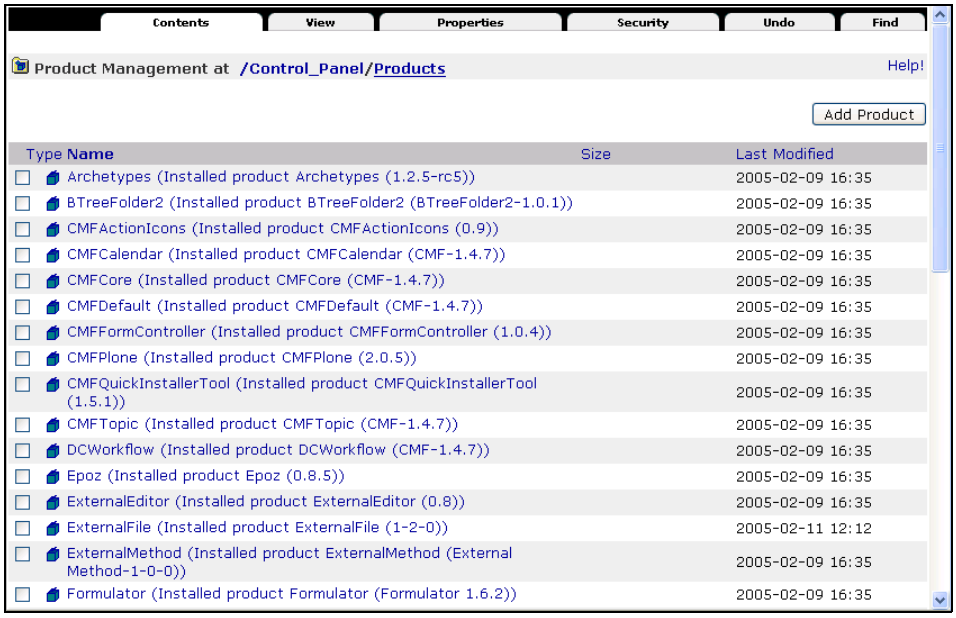


Abbildung 10.3: Korrekt installierte Produkte

Gelegentlich kann an dieser Stelle eines von drei Dingen schief gehen. Sollte zum einen im ZMI nichts auftauchen, dann haben Sie das Verzeichnis an den falschen Ort kopiert. Korrigieren Sie das, indem Sie die Installationsanweisungen und den Ort Ihres Products-Verzeichnisses noch einmal überprüfen, wie zuvor erklärt wurde.

Zweitens könnten Sie ein »defektes« Icon in der Produktliste sehen, was bedeutet, dass versucht wurde, das Produkt in Zope zu registrieren, wobei aber ein Fehler aufgetreten ist. Klicken Sie auf das defekte Icon, um einen Traceback zu erhalten, der Ihnen den Fehler nennen und eine Möglichkeit bieten sollte, ihn zu beheben.

Und sollten Sie schließlich nach dem Neustart nicht mehr auf das ZMI zugreifen können, dann haben Sie womöglich ein ernsteres Problem. Zope konnte dann nicht gestartet werden, weil Plone einen ernsten Fehler gefunden hat. Um herauszufinden, was das Problem ist, starten Sie Plone von der Kommandozeile im Debug-Modus. Sie erhalten auf dem Bildschirm einen Traceback.

10.1.3 Installation in Plone

Nach der korrekten Installation in Zope ist der nächste Schritt einfach. Um CMFExternalFile vollständig zu installieren, müssen Sie das Produkt CMFExternalFile (*http://prdownloads.sourceforge.net/collective/CMFExternalFile.0.5.zip?download*) nun auf dieselbe Weise installieren, wie Sie ExternalFile installiert haben. Dann müssen Sie Plone wieder neu starten.

Sie müssen CMFExternalFile in *allen* Plone-Instanzen installieren. Nicht alle, aber die meisten Plone-Produkte verlangen das. Wie man das macht, weiß man nur dann genau, wenn man die Installationsanweisungen anschaut. Wenn Sie etwas lesen wie »auf die normale CMF-Art installieren« oder »erstellen Sie in Ihrer Plone-Instanz eine externe Methode«, dann müssen Sie diesen Schritt ausführen.

Glücklicherweise können Sie die Anweisung, eine externe Methode zu erstellen, sogar ignorieren, da Plone über einen viel einfacheren Weg dafür verfügt. Klicken Sie in Plone auf PLONE KONFIGURATION und dann auf PRODUKTE HINZUFÜGEN/LÖSCHEN. Sie erhalten eine Liste von Produkten, die auf Ihrem Server installiert sind und in Plone konfiguriert werden müssen. Klicken Sie einfach auf das Kontrollkästchen neben dem Produkt (in diesem Fall CMFExternalFile), und klicken Sie dann auf INSTALLIEREN, wie in Abbildung 10.4 zu sehen ist.

Damit wird das Produkt installiert – jedenfalls, wenn es keinen Fehler dabei gibt. Sonst erscheint es nicht in dieser Liste der installierten Produkte. Dann können Sie das Problem eventuell dadurch lösen, dass Sie die Protokolldatei lesen. Klicken Sie also auf den Link neben dem Produktnamen, um das Protokoll zu erhalten. Diese Installation ist ein Dienst, der vom Werkzeug `portal_quickinstaller` in Zope bereitgestellt wird. Um zu sehen, was dieses Produkt tatsächlich tut, springen Sie zum Abschnitt »Integration von Plone mit dem Dateisystem«.

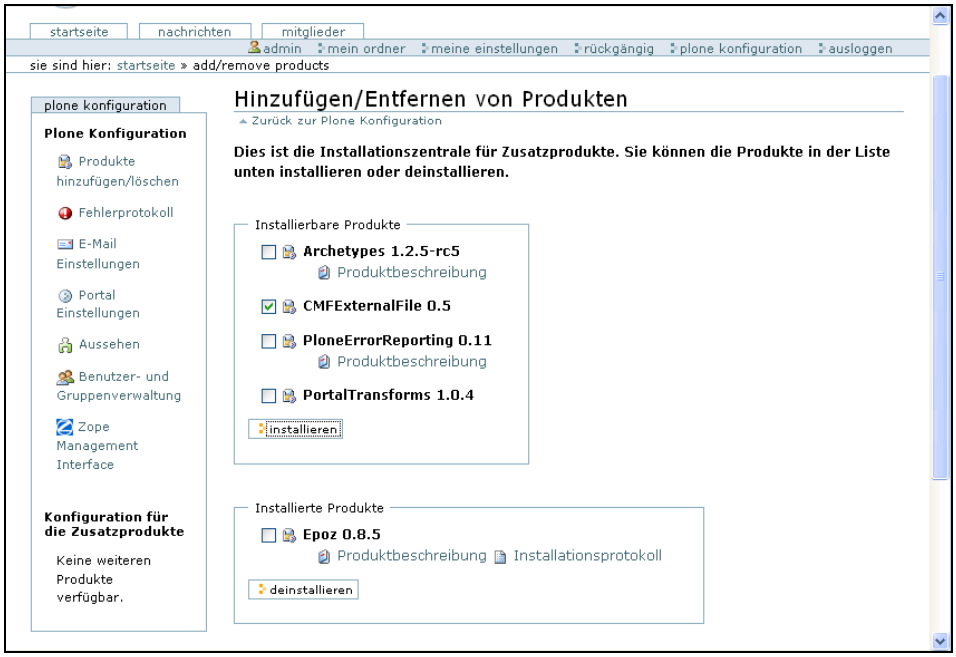


Abbildung 10.4: Eine Liste für den Benutzer verfügbarer Produkte

10.2 Einen anderen Webserver verwenden

Wenn Sie Teil einer Organisation sind, die bereits Websites betreibt, dann verwenden Sie sehr wahrscheinlich eine bestimmte Plattform für den Webserver. Unter *Virtual Hosting* versteht man die Möglichkeit, mehrere Websites auf einem Server zu betreiben, wobei die Sites über ihre IP- (Internet Protocol) Adressen oder Namen unterschieden werden. Damit kann ein erster Server, z.B. Apache, Anfragen an eine oder mehrere Plone-Instanzen weiterreichen.

Virtual Hosting wird normalerweise mit Hilfe von *Proxies* erreicht, auch wenn der Einsatz eines Proxy-Servers mit Plone ganz unabhängig von der Anzahl der gehosteten Sites ein wünschenswerter Ansatz ist. Ein Proxy-Server sitzt zwischen einem Client und einem Server und leitet Anfragen von Client und Server weiter. Ein Proxy-Server sollte für den Benutzer transparent sein. In Kapitel 14 werde ich Ihnen zeigen, wie Sie Proxy-Server benutzen können, um die Performance von Plone dramatisch zu steigern.

Obwohl Plone den Webserver benutzt, der Zope zugrunde liegt, funktioniert dieser ZServer sehr gut. Es ist aber kein vollständiger, industrietauglicher Webserver, auf den man die Welt loslassen sollte. Der Server hat einige Probleme, was mögliche Denial-of-Service-(DOS-)Attacken angeht, allerdings sind das in ZSer-

ver obskure und schwer zu findende Probleme. Es sind keine Attacken gegen ZServer bekannt, die diese Probleme ausnutzen, was aber vielleicht an deren relativ obskurem Charakter in der realen Welt liegt. ZServer ist nicht speziell als industrietauglicher Server entworfen worden, und da seine Eigenschaften ausreichend vollständig sind, wird er nicht weiterentwickelt. Durch die Aktualisierung eines Servers wie Apache können Sie garantieren, dass Ihre Besucher einen robusten, sicheren Server zu Gesicht bekommen. Wenn Sie natürlich eine Intranet- oder andere Anwendung für vertrauenswürdige Benutzer entwickeln, ist das möglicherweise kein Problem.

Abbildung 10.5 zeigt, wie eine solche Einstellung aussehen könnte. Sie zeigt keine echten Rechner, sondern nur Dienste. Eine Anfrage käme normalerweise aus dem Internet zur Firewall, um dann an Apache und Plone zu gelangen. All diese Dienste könnten auf verschiedenen Rechnern laufen. Der wichtige Punkt ist der, dass nicht vertrauenswürdige Benutzer keinen Zugriff auf Plone erhalten sollten, außer über einen Proxy.

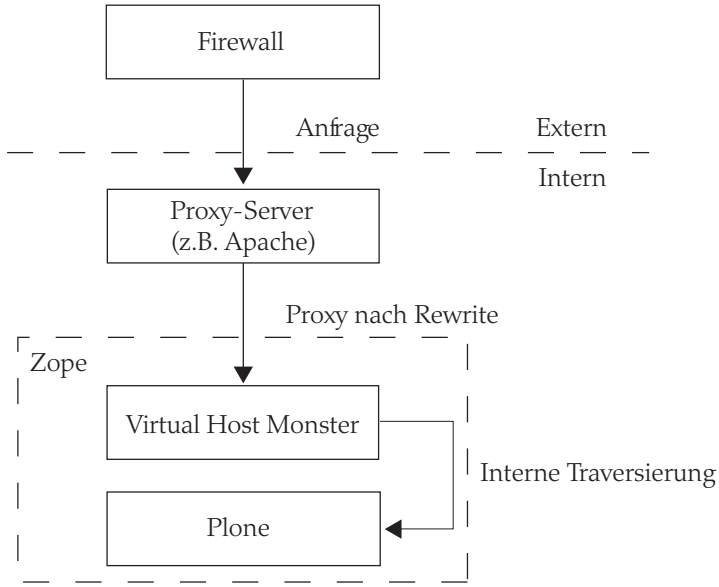


Abbildung 10.5: Funktionsweise des Virtual Hosting

Das Vorschalten eines Webservers wie Apache vor Plone bringt eine Reihe nützlicher Dienste mit sich, die ZServer nicht hat. Apache bietet z.B. Folgendes: Rewriting von URLs (Uniform Resource Locator), Unterstützung von SSL (Secure Sockets Layer), Caching, Inhaltsdekomprimierung, Virtual Hosting, Proxy-Dienste auf andere Webservices, Prüfung eingehender Anfragen usw. Die am häufigsten gestellte Frage ist die, wie man eine URL wie *http://localhost:8080/Plone*

in etwas Freundlicheres wie *http://ihresite.com* ändert. Das bezeichnet man als *URL-Rewriting*. Ein Proxy-Server ist dafür zwar nicht notwendig, aber mit einem solchen ist das viel einfacher.

Eine beliebte Methode für die Proxy-Weiterleitung zu Plone besteht im Einsatz eines HTTP-Proxys (Hypertext Transfer Protocol). Apache bewerkstelligt das mit Hilfe des Moduls `mod_proxy`. Wenn ein Proxy-Server eine Anfrage nach einer Seite erhält, führt er verschiedene Funktionen aus. Dann wird eine neue Anfrage erzeugt und an den ZServer geschickt. Dessen Antwort wird an den Server und dann an den Client zurückgegeben. Natürlich ist das für den Client alles transparent. Er stellt ganz normale Anfragen an einen Server.

Hinweis

Die alte Methode, Apache mit Plone zu verbinden, ist die über Fast CGI oder Persistent CGI. Diese sind schwieriger zu konfigurieren und sogar langsamer im Betrieb. Zwar existiert eine Menge alter Dokumentation zu diesen Themen, aber heute gibt es effizientere Lösungen. Daher kann ich diese alten Methoden *nicht* empfehlen.

10.2.1 Plone konfigurieren

Bevor Sie Ihren Proxy-Webserver konfigurieren, müssen Sie zuerst Plone konfigurieren. Da jeweils nur ein Server einen Port belegen kann, ändern Sie Plone so, dass es einen Port mit einer höheren Nummer belegt. Normalerweise wäre das ein Port, der von der Firewall blockiert wird und von außen nicht erreichbar ist. Beispiel-Ports sind 8080, 9090, 9673 usw. Kapitel 2 enthält Informationen darüber, wie Sie die Ports ändern können, auf denen Ihr Plone-Server läuft.

Als Nächstes möchten Sie wahrscheinlich mit Hilfe von URL-Rewriting die URL Ihrer Site ändern. Da das Plone-Objekt in der Zope-Objektdatenbank (ZODB) liegt und eine ID hat, greift man darauf zu, indem diese ID in die URL gesetzt wird, wie in *http://localhost:8080/Plone*. Um das etwas freundlicher zu machen, müssen Sie die Anfrage an den Webserver von *http://ihresite.com* in eine Anfrage an das richtige Objekt in Zope übersetzen. Dafür haben Sie je nach Bedarf zwei leicht unterschiedliche Möglichkeiten. Wenn Sie einen Proxy-Webserver benutzen oder Ihre Sites auf Domain-Namen basieren, können Sie ein Virtual Host Monster (VHM) benutzen. Dies ist ein benutzerfreundliches und mächtiges Objekt, das Ihnen das Leben sehr viel leichter machen wird, daher empfehle ich es wärmstens. Sie benötigen in der Wurzel einer Zope-Instanz nur ein VHM. Das VHM-Objekt sitzt in der Wurzel einer Zope-Site und fängt alle eingehenden Anfragen ab. Dann ändert es die Anfragen, damit sie an den gewünschten Teil von Zope gehen.

Um ein VHM zu erstellen, gehen Sie im ZMI zur Zope-Wurzel und wählen *Virtual Host Monster* im Dropdown-Menü aus. Im dann erscheinenden Formular geben Sie eine ID ein, z.B. **vhm** (wie die ID genau aussieht, spielt keine Rolle).

Wenn Sie einen Proxy-Webserver vor Plone benutzen, machen Sie an diesem Punkt mit der Konfiguration dieses Webservers im Abschnitt »Konfigurieren des Proxy-Servers« weiter.

Der nächste Schritt ist nur dann notwendig, wenn Sie *keinen* Proxy-Webserver benutzen. Klicken Sie auf das VHM-Objekt, das Sie im ZMI erstellt haben, und wählen Sie dann den MAPPINGS-Reiter, der eine Liste der verfügbaren Abbildungen von Hosts auf dieses Objekt anzeigt. Eine Abbildung nimmt eine ankommende Anfrage und bildet sie mit der folgenden Syntax auf Plone ab

```
host/pfad
```

wobei `host` der abgebildete Hostname und `pfad` der eigentliche Pfad zu dem Objekt in Zope ist. Beispiel:

```
www.einesite.com/Plone
```

Um zu garantieren, dass alle Namensvarianten auf den Pfad abgebildet werden, können Sie in der Abbildung Joker benutzen. Im folgenden Beispiel werden alle Unterdomains von `einesite.com` abgebildet:

```
*.einesite.com/Plone
```

Um diese Abbildung hinzuzufügen, gehen Sie zum MAPPING-Reiter, geben eine Abbildung pro Zeile ein und klicken auf SAVE. Das bedeutet, Sie können mit den abgebildeten Adressen nicht mehr auf die Wurzel Ihrer Zope-Site zugreifen. Glücklicherweise können Sie aber mit einer IP-Adresse weiterhin auf die Wurzel Ihres Zope-Servers zugreifen. Das funktioniert weiterhin, weil numerische Adressen von der Abbildung nicht betroffen sind. Abbildung 10.6 zeigt, wie sich Abbildung 10.5 ändert, wenn Sie das Rewriting umgehen und direkt über die IP auf den Server zugreifen.

Nun haben Sie eine benannte Domain wie `einesite.com` auf eine bestimmte Plone-Instanz abgebildet. Wenn eine Anfrage nach diesem Site-Namen ankommt, wird sie zu der Plone-Instanz weitergeleitet.

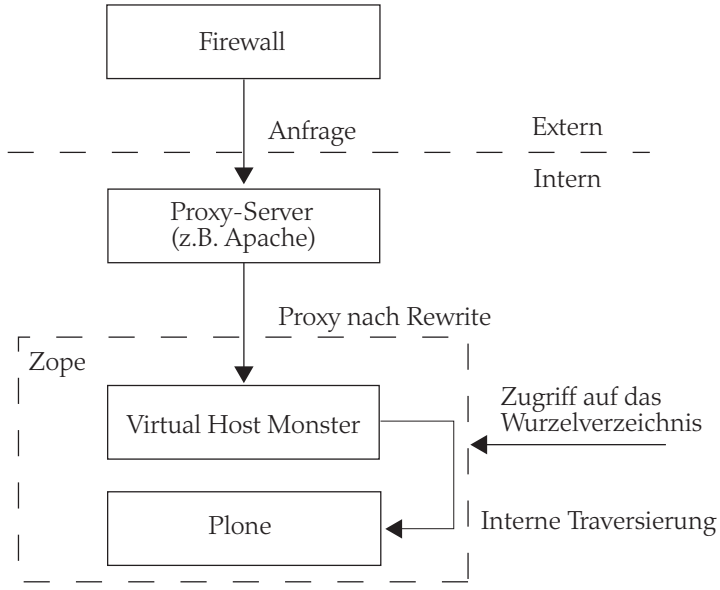


Abbildung 10.6: Virtual Hosting mit Root-Zugriff

10.2.2 Konfigurieren des Proxy-Servers

Nachdem Sie Ihr VHM in Plone hinzugefügt haben, ist es Zeit, den Proxy-Server zu konfigurieren. Aber die Konfiguration von Proxy-Servern ist vom tatsächlich eingesetzten Server abhängig. Die folgenden Abschnitte behandeln die Eigenarten jedes Servers. Um das Virtual Hosting zum Laufen zu bekommen, müssen Sie allerdings eine URL an Plone weiterleiten, die das VHM-Objekt versteht.

Es gibt einen weiteren Vorteil des Virtual Hosting mit einem Proxy-Server, der erwähnt werden sollte. Die gesamte Konfiguration der Domains nehmen Sie im Proxy-Server, also außerhalb von Plone, vor. Das bedeutet, dass Ihr Systemadministrator ein vertrautes Werkzeug verwalten und benutzen kann, ohne sich um Plone kümmern zu müssen.

Die Weiterleitung funktioniert so, dass eine ankommende Anfrage so manipuliert wird, dass eine Anfrage mit einer speziellen URL an Plone gesendet wird. Diese Anfrage ist manipuliert und enthält alle Informationen, die Plone benötigt, um eine Antwort zu produzieren. Wenn die Antwort produziert und an die fragende Person zurückgeschickt wird, müssen alle URLs korrekt auf Ihre Site zeigen. Das garantiert, dass alle Links innerhalb Ihrer Seite korrekt sind.

Eine URL besteht aus den folgenden drei Hauptkomponenten:

- Aus einer IP-Adresse oder einem Hostnamen und Port für den Server, auf dem Plone läuft.
- Aus einer IP-Adresse oder einem Hostname dafür, wo Plone laufen soll, damit alle Links in den entstehenden Dokumenten die korrekte URL haben.
- Aus dem eigentlichen Objekt in Zope, das erreicht werden soll, und aus der ihm übergebenen URL.

Diese Information wird an Plone übergeben, indem die URL in eine lange, komplizierte URL in folgendem Format übersetzt wird (zur Verdeutlichung wurden hier Zeilenenden hinzugefügt):

```
http://[URL to server]:[port]
/VirtualHostBase/[protocol]/[URL]:[port]
/[path to virtual host root]
/VirtualHostRoot/[actual URL]
```

Betrachten Sie folgendes Beispiel:

- Plone läuft auf einem Rechner mit der IP-Adresse 192.168.2.1 auf Port 8080. Beachten Sie, dass die IP-Adresse eine ist, auf die der Proxy-Server zugreifen kann. Es ist nicht die IP-Adresse für den Rest der Welt – um diese kümmert sich der Proxy-Server.
- Plone sollte unter `www.meinesite.com` auf Port 80 erscheinen.
- Das eigentliche Plone-Objekt findet man unter `/Plone`.
- Die ankommende Anfrage lautet `/Members/andym`.

Daraus wird die folgende lange URL erzeugt:

```
http://192.168.2.1:8080 ~↵
/VirtualHostBase/http/www.meinesite.com:80/Plone ~↵
/VirtualHostRoot/Members/andym
```

Das macht man deswegen, weil das VHM-Objekt genau weiß, was es mit dieser URL machen soll, wenn es sie sieht. Es verändert sie und sendet die Anfrage an das Plone-Objekt. Ganz offensichtlich wird das Seitenfragment (`/Members/andym`) für jede Anfrage verschieden sein und muss berechnet werden. Aber wenn Sie wissen, was Sie vorhaben, können Sie Ihren Server nun konfigurieren.

Apache konfigurieren

Apache ist wahrscheinlich der beliebteste Server, der vor Plone gesetzt wird, und ist für alle Linux-, Unix- und Windows-Plattformen verfügbar (*http://httpd.apa-*

che.org/). Nach der Installation von Apache müssen Sie Anfragen mit HTTP-Proxies an Plone weitergeben.

Um Apache zu konfigurieren, benötigen Sie den Zugriff auf Apaches Konfigurationsdateien, deren Ort von Ihrer Apache-Installation abhängt. Wenden Sie sich dazu an die Apache-Dokumentation. Unter Windows ist die Konfiguration vom START-Menü aus erreichbar. Und unter Linux finden Sie die Apache-Konfiguration im Verzeichnis /etc unter `/etc/apache/httpd.conf` oder `/etc/apache2/httpd.conf`. Um diese Dateien verändern zu dürfen, müssen Sie normalerweise Root oder ein privilegierter Benutzer sein.

Hinweis

In diesem Beispiel wird Apache 2 verwendet, aber all diese Befehle sind mit früheren Versionen wie Apache 1.3.2 rückwärtskompatibel. Einige ältere Apache-Versionen (vor 1.3.2) sind allerdings für Probleme mit Cookies bekannt.

Die einfachste Art eines URL-Rewrite in Apache ist die Verwendung der eingebauten Rewrite- und Proxy-Module. Das bedeutet, dass Apaches Module `mod_rewrite` und `mod_proxy` aktiviert werden müssen. In Apache ist jede Site normalerweise in einem Virtual Host-Verzeichnis enthalten, dessen Beschreibung mit dem folgenden Code anfängt:

```
<VirtualHost *:80>
    ServerName ihresite.com
    # other configuration options
```

Sie müssen lediglich Rewrites aktivieren und die Rewrite-Regel hinzufügen, etwa so:

```
    RewriteEngine On
    RewriteRule ^/(.*)  http://192.168.2.1:8080 ~↵
/VirtualHostBase/http/www.meinesite.com:80/Plone ~↵
/VirtualHostRoot/$1 [L,P] ~↵
</VirtualHost>
```

Die Rewrite-Regel, um die es hier geht, nimmt einen beliebigen ihr übergebenen Anfrage-String und fügt ihn ans Ende Ihrer festcodierten Rewrite-Regel an. Das `[L,P]` sagt Apache, dass das die letzte Rewrite-Regel ist und sie zum angegebenen Proxy weiterleiten soll. Danach müssen Sie Apache neu starten, um die Konfiguration zu aktualisieren. Weitere Informationen über das Rewriting finden Sie in der Dokumentation zu `mod_rewrite` unter *http://httpd.apache.org/docs-2.0/misc/*

rewriteguide.html. Beachten Sie, dass Sie in diesem Fall die Information zur Rewrite-Regel in einer Virtual Host-Direktive gesetzt haben. Sie könnten in Apache auch mehrere solche Virtual Hosts haben, so dass PHP-, Perl- und Java-Sites alle nebeneinander auf einem Server liegen könnten.

Squid

Squid ist wegen seiner mächtigen Caching- und Konfigurationsmöglichkeiten bei Benutzern sehr beliebt. Es ist für Unix verfügbar, und es gibt mit Cygwin übersetzte Pakete für Windows. Ich habe die Windows-Version nicht speziell getestet, aber es heißt, sie funktioniere gut. Sie finden die Dateien zum Herunterladen unter *http://www.squid-cache.org*. Die folgenden Bemerkungen gelten für die letzte beim Schreiben dieses Buchs aktuelle stabile Version 2.5.

Die Installation von Squid aus seinem Quellcode ist recht einfach. Geben Sie nach dem Herunterladen folgende Befehle ein:

```
$ tar -xvf squid-2.5.STABLE3.tar.gz
$ cd  squid-2.5.STABLE3
$ ./configure --prefix=/usr/local/squid
...
$ make all
...
$ make install
...
```

Leider kennt Squid keine Rewrite-Regeln, mit denen Sie ankommende Anfragen vor ihrer Weiterleitung ändern können. Aber *Squid Guard* (*http://www.squidguard.org*) kann das. Ich habe die Version 1.2.0 getestet. Nach dem Herunterladen geben Sie folgende Befehle zur Installation ein:

```
$ tar -zxvf squidGuard-1.2.0.tar.gz
$ cd squidGuard-1.2.0
$ ./configure
...
$ make
...
$ make install
...
```

Nun sind sowohl Squid als auch SquidGuard bereit, aber es müssen noch beide Konfigurationsdateien eingerichtet werden. Die Konfigurationsdatei für Squid finden Sie unter `/etc/squid.conf`. Es ist eine lange Datei, in der zum Glück alle Optionen detailliert erklärt werden. Das Folgende sind die wesentlichen Optionen, die man einstellen muss:

```
http_port 80
httpd_accel_host virtual
httpd_accel_port 0

http_access allow all
http_access allow localhost
```

Diese letzten beiden Zeilen sind Sicherheitsregeln, die den Zugriff über einen Browser erlauben. Das sind lasche Regeln, weil sie hinter einer Firewall getestet wurden. Wenn Sie Squid extern betreiben, sollten Sie sich über Zugriffsregeln im Detail informieren. Am leichtesten sichert man diese Regeln ab, indem man `http_access allow all` in `http_access deny all` ändert. Fügen Sie abschließend folgende Zeile zur Konfigurationsdatei hinzu:

```
redirect_program /usr/bin/squidGuard -c /etc/squid/squidGuard.conf
```

Das richtet einen Redirect über Squid Guard mit der Konfigurationsdatei unter `/etc/squid/squidGuard.conf` ein. SquidGuard enthält von sich aus keine Konfigurationsdatei, aber eine Standarddatei, die die Virtual Host-Konfiguration enthält, sieht wie folgt aus:

```
dbhome /var/lib/squidguard/db
logdir /var/log/squid
acl {
    default {
            redirect http://192.168.2.1:8080 ~↵
            /VirtualHostBase/http/www.agmweb.ca:80 ~↵
            /Plone/VirtualHostRoot/%p ~↵
  }
 }
```

Schließlich hat Squid die benötigte Konfiguration, um den Datenverkehr so umzuleiten, dass das Host-Monster ihn versteht. Ankommende Anfragen werden dann von Squid behandelt und an Plone weitergegeben.

Microsoft Internet Information Services

Der MS-IIS (Internet Information Services) ist nicht gerade mein bevorzugter Server, aber weil viele Firmen ihn verwenden, habe ich hier einen Abschnitt über IIS aufgenommen. Leider kann IIS keine Weiterleitung auf die gleiche Weise wie Squid und Apache vornehmen, sondern Sie brauchen ein separates Plug-In dafür. Kurz vor der Veröffentlichung dieses Buches wurde ein freier Proxy namens IIS2Zope geschrieben, der genau diese Funktionalität bietet. Allerdings konnte ich ihn noch nicht auf einer Site testen, die performant genug wäre. Weitere Informationen finden Sie unter *http://zope.org/Members/freshlogic/index_html*.

Stattdessen möchte ich eine freie und einfach einzurichtende Lösung vorstellen. Früher haben Zope-Benutzer PCGI empfohlen, aber mit der Zeit wurde es langsam und schwer zu installieren. Durch die Verwendung der Sprache ASP von Microsoft und einiger Eigenschaften von IIS bekommen Sie eine schnellere Lösung. Sie heißt ASP 404 und führt die Weiterleitung mit Hilfe der Programmiersprache ASP aus.

Laden Sie unter *http://www.zope.org/Members/hiperlogica/ASP404* die letzte Version herunter. Ich habe `ASP404_1-0-b2.zip` getestet. Wenn Sie die geladene Datei auspacken, werden Sie darin eine Datei namens `default.asp` finden. Nehmen Sie diese Datei, und kopieren Sie sie an die Wurzel der Site, die Sie weiterleiten möchten. Auf meinem Server ist das `C:\inetpub\wwwroot`.

Als Nächstes müssen Sie das Script mit der passenden Angabe zum Ort Ihrer Plone-Site konfigurieren. Sie müssen das Script in einem einfachen Texteditor öffnen und zwei Zeilen ändern, die die Variablen für die Site-Konfiguration enthalten. Konkret: Ändern Sie Zeile 18 von

```
zopeAddress = "http://127.0.0.1:8080"
```

in die Adresse des Zielservers. In diesem Beispiel ist das:

```
zopeAddress = "http://192.168.2.1:8080"
```

Ändern Sie dann Zeile 27 von

```
zopePath = "/"
```

wie folgt in die ID des Plone-Objekts:

```
zopePath = "/Plone"
```

Speichern Sie die Datei, und schließen Sie den Editor. Und schließlich müssen Sie IIS noch beibringen, dass er mit Plone sprechen soll. An dieser Stelle müssen Sie ein bisschen tricksen. Öffnen Sie den Internet Services Manager, den Sie normalerweise irgendwo in der Windows-Systemsteuerung finden. Suchen Sie die weiterzuleitende Site, und greifen Sie auf die Site-Eigenschaften zu, wie in Abbildung 10.7 zu sehen ist.

Wählen Sie in den Eigenschaften den Reiter BENUTZERDEFINIERTE FEHLER, und scrollen Sie bis zum Fehler für 404 nach unten. Doppelklicken Sie auf den Fehler 404, und ändern Sie ihn wie folgt:

- NACHRICHTENTYP: URL
- URL: /default.asp

Abbildung 10.8 zeigt die Einstellungen.

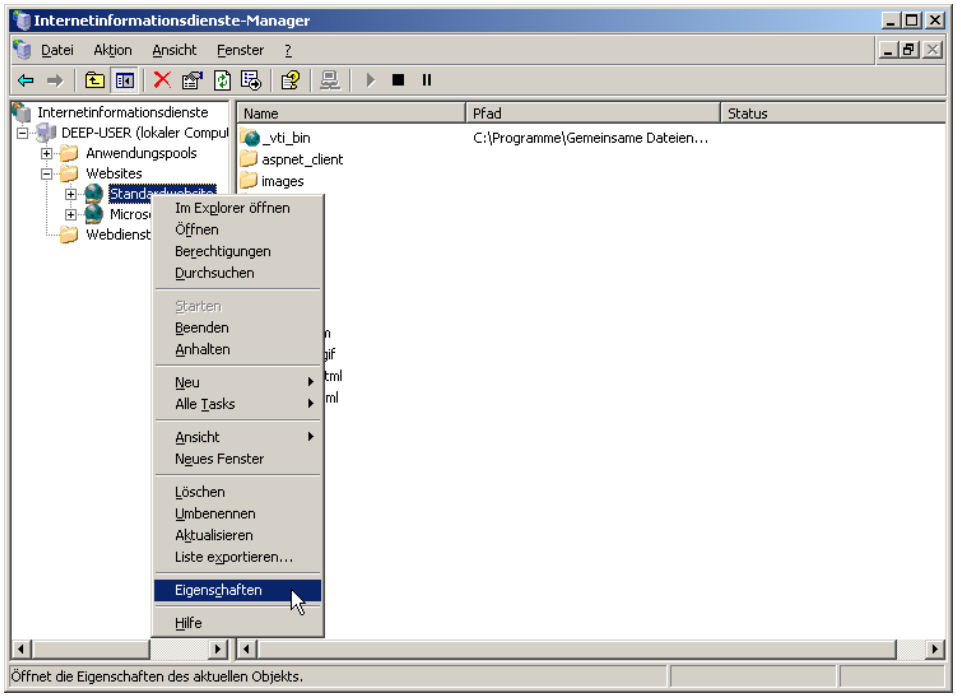


Abbildung 10.7: Zugriff auf die Eigenschaften der Site

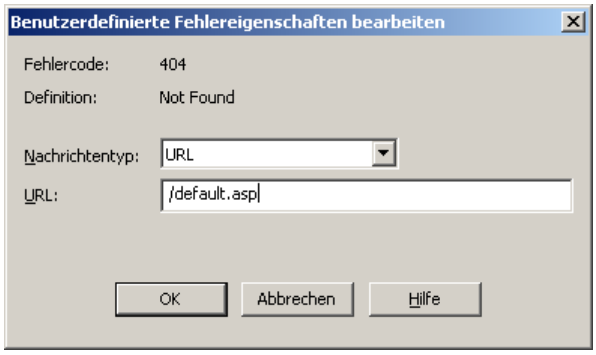


Abbildung 10.8: Die Weiterleitung für 404 einstellen

Speichern Sie die Änderungen mit einem Klick auf OK. Nun sollte Ihre Fehlerliste wie die in Abbildung 10.9 aussehen. Wenn das der Fall ist, sollte es richtig eingestellt sein. Greifen Sie über den Browser auf IIS zu, und Bingo – Sie sehen Plone!

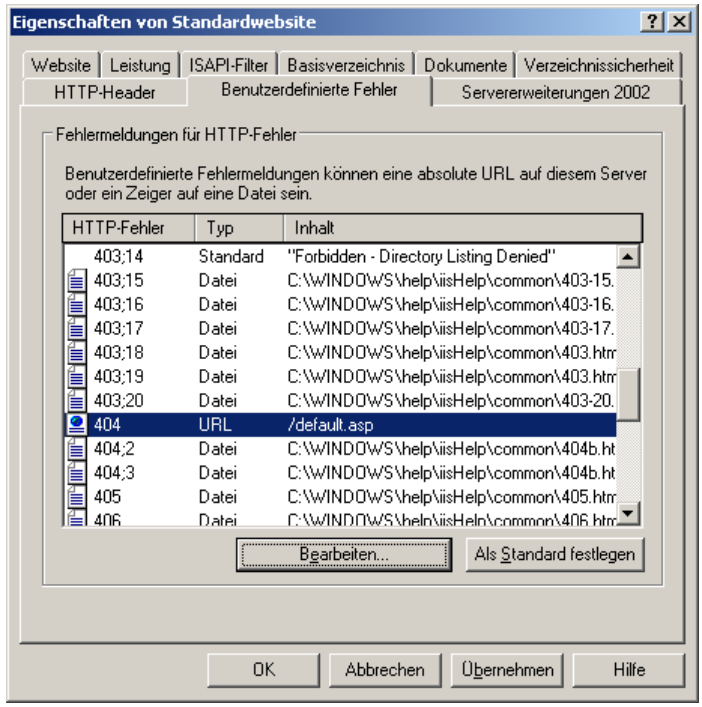


Abbildung 10.9: Die Fehlerliste

Hier ist Folgendes passiert: Sie fangen den Fehler dafür ab, dass ein Element in IIS nicht gefunden werden kann. Das ASP-Script, das Sie installiert haben, liest dann die Anfrage und leitet sie an Plone weiter. Dann nimmt es die Antwort entgegen und reicht Sie an IIS und schließlich an den Browser zurück. Das heißt, Sie haben ein einfaches Proxy-Programm zu IIS hinzugefügt.

Hierbei müssen Sie allerdings einige Punkte beachten. Der erste ist der, dass eine Seite nicht gefunden werden darf, wenn die Weiterleitung erfolgen soll, sonst wird das Script nicht ausgeführt. Das hat Vor- und Nachteile. Sie können zu IIS Ordner und Bilder hinzufügen, die dem Benutzer anstelle derer aus Plone angezeigt werden, falls die Namen auf die Anfrage vom Browser passen. Zweitens wird die ankommende Anfrage geparst und weitergereicht, was in manchen Situationen mit all den möglichen HTTP-Anfragekonfigurationen verwirrend wird. Außerdem werden Sie feststellen, dass all Ihre Plone-Anfragen von IIS als 404-Fehler protokolliert werden, was Analysewerkzeuge für solche Dateien verwirren kann.

Im Großen und Ganzen hat diese Einstellung für die meisten Leute funktioniert, die sie verwendet haben, aber dass es eine Enterprise-Lösung sein kann, die mit jeder Situation fertig wird, ist eher unwahrscheinlich. Auf jeden Fall ist es eine gute Ausgangsbasis für diejenigen, die damit arbeiten und entwickeln möchten.

Fehlersuche bei Proxy-Servern

Nachdem Sie den Server eingerichtet und alles neu gestartet haben, werden Sie den Server testen wollen, indem Sie mit Ihrem Browser die Site besuchen. Nachdem Sie das ein paarmal gemacht haben, können Sie folgende Tipps verwenden, falls etwas nicht ganz so funktioniert, wie es sollte:

- **Testen der Site**: Die goldene Regel bei der Fehlersuche in Proxy-Servern ist die, die Site *immer* zu testen, indem Sie sich anmelden und Ihren Proxy-Server benutzen. Das können Sie tun, indem Sie direkt auf die IP-Adresse und den Port Ihres Plone-Servers zugreifen. Im vorigen Beispiel können Sie auf die Site zugreifen, indem Sie auf *http://192.168.2.1:8080/Plone* gehen, und schon haben Sie den Proxy-Server vollständig umgangen. Wenn Sie keine Probleme haben, auf diese Weise auf Plone zuzugreifen und sich darin anzumelden, aber sehr wohl Probleme bekommen, wenn Sie es über den Proxy-Server versuchen, ist es wahrscheinlich, dass etwaige Fehler auf Seiten des Proxy-Servers liegen. Einige ältere Versionen von Apache 1.3 machen bei der Anmeldung Probleme mit Cookies, d.h., Sie sollten auf die neueste Version von 1.3 aktualisieren.
- **Prüfen der URL**: Überprüfen Sie doppelt, ob Ihr Proxy-Server die richtige URL benutzt, die ziemlich lang und kompliziert sein kann. Unterteilen Sie sie an den Schrägstrichen, um jeden Teil zu untersuchen. Bedenken Sie, dass Sie Plone die richtigen Werte übergeben müssen, damit es die korrekte URL zurückgeben kann. Das heißt, Sie müssen sicherstellen, dass der Abschnitt `/[protocol]/[URL]:[port]` korrekt ist. Falls Ihre Site z.B. SSL verwendet, stellen Sie sicher, dass der Protokollabschnitt `https` und nicht `http` lautet.

10.3 Integration von Plone mit dem Dateisystem

Eine Integration von Plone mit dem Dateisystem mag etwas merkwürdig klingen, aber hiermit meine ich, dass man in Plone Inhalte benutzen kann, die im Dateisystem liegen. Natürlich besteht Plone schon aus einer Reihe von Dateien, die im Dateisystem installiert sind und von dort ausgeführt werden. Allerdings werden in Plone alle Inhalte in der ZODB gespeichert, wenngleich viele Leute mir sagen, sie möchten ihre Inhalte im Dateisystem speichern und von dort ausgeben.

Tatsächlich ist es so, dass viele Leute sich Zope und Plone anschauen, kleine Icons für Ordner sehen und annehmen, dass diese direkt den Ordnern und Ele-

menten im Dateisystem entsprechen. Allerdings ist das überhaupt nicht der Fall. Angenommen, Sie würden eine relationale Datenbank benutzen, wie es die meisten Content-Management-Systeme (CMS) tun. Würden Sie das dann immer noch so sehen? Viele Leute denken automatisch, dass das ein Problem sei, aber hier ist eine Liste von Gründen, warum Sie das vielleicht haben möchten:

- **Sie haben viele sehr große Inhaltselemente**: Plone kann mit sehr großen Dateien ohne große Probleme umgehen. Datenbanken mit mehr als 10 Gigabyte sind nicht so selten und funktionieren einwandfrei. Wenn Sie es mit wirklich großen Inhalten zu tun haben (ein Kunde von mir benutzt Plone, um seine DVDs zu verwalten, d.h. die eigentlichen Inhalte der DVDs), sollten Sie sich CMFExternalFile und Apache anschauen. Bei wirklich großen Sachen ist die Verwendung von Apache oder eines anderen Dienstes zur Ausgabe Ihres Inhalts ein guter Ansatz.
- **Sie möchten Inhalte mit Programmen verwalten, die vom Dateisystem lesen, z.B. Microsoft Word**: Mit External Editor können Sie Inhalte mit Ihren lokalen Programmen bearbeiten, die in einer Plone-Site gespeichert sind. Wenn Sie Microsoft Word installiert haben, können Sie ein Microsoft Word-Dokument hochladen und es auf Ihrem Rechner mit Microsoft Word bearbeiten.
- **Sie sind es leid, Code über das Web in kleinen Textbereichen zu bearbeiten**: Schauen Sie sich wieder zuerst External Editor an. Zweitens, warum arbeiten Sie überhaupt über das Web? Wie ich in Kapitel 7 demonstriert habe, können Sie alle Skins und CSS-Templates im Dateisystem schreiben.
- **Sie können meine Dateisystemwerkzeuge auf den Inhalt anwenden**: Nun, Sie können Plone über FTP (File Transfer Protocol) und WebDAV mounten. Beide bieten dateisystemähnliche Schnittstellen, die mit Plone zusammenarbeiten.
- **Sie möchten leicht Backups von Ihrem Inhalt anfertigen**: In Kapitel 14 zeige ich, wie man Plone administriert und Backups davon macht und wie man einfache, inkrementelle Backups anlegt. Ein alternativer Speicher namens Directory Storage kann das ebenfalls bewerkstelligen (siehe *http://dirstorage.sf.net*).
- **Sie möchten CVS/Subversion/BitKeeper oder ein anderes System zur Versionsverwaltung der Inhalte benutzen**: Ja, das macht Sinn, ist aber leider noch nicht vollständig integriert. In zukünftigen Versionen, von denen eine provisorisch Plone 3 heißt, wird das wahrscheinlich integriert sein.

Mit diesen Punkten im Hinterkopf werden Sie nun verschiedene Methoden kennen lernen, um Inhalte aus Plone auszugeben, die im Dateisystem existieren.

10.3.1 Den Proxy-Webserver benutzen

Nun haben Sie also den Webserver so eingerichtet, wie es zuvor in diesem Kapitel beschrieben wurde. Auf die Gefahr hin, dass ich mich wiederhole: Dieser Webserver kann einfache Inhalte weit besser ausgeben, als Plone das jemals können wird. Wenn Sie eine hohe Anzahl von herunterzuladenden Dateien haben, sollten Sie diese in ein Verzeichnis auf Ihrem Server platzieren, das nicht zu Plone weitergeleitet wird, und von Plone aus Links darauf setzen. Die Benutzer werden einfach auf einen Link klicken und die Dateien ganz normal herunterladen.

In IIS kann man das leicht machen, da IIS automatisch zuerst prüft, ob die Datei existiert, bevor es einen Fehler 404 ausgibt. Apache benötigt dafür nur zwei zusätzliche Zeilen in der Konfiguration, die im folgenden Code fett hervorgehoben sind:

```
<VirtualHost *:80>
    ServerName ihresite.com
    # other configuration options
    DocumentRoot /var/downloads
    RewriteEngine On
    RewriteRule ^/download(.*) - [L]
    RewriteRule ^/(.*)  http://192.168.2.1:8080 ~↵
    /VirtualHostBase/http/www.meinesite.com:80 ~↵
    /Plone/VirtualHostRoot/$1 [L,P]
</VirtualHost>
```

In diesem Beispiel platzieren Sie den Inhalt in `/var/downloads`, und die URLs zu den über Apache herunterladbaren Inhalten fangen alle mit `/download` an.

Der Rewrite-Mechanismus sieht dann, dass die URL mit `/download` anfängt, und wird daher keinerlei Änderungen daran vornehmen, was mit dem Bindestrich (`-`) ausgedrückt wird. Durch die Angabe des `[L]` am Zeilenende werden keine weiteren Rewrite-Regeln angewendet, d.h., es findet keine Weiterleitung statt, und Apache macht ganz normal weiter, indem es die Datei ausgibt.

Dieser Trick ist hilfreich, wenn Sie andere Dienste auf dem gleichen Virtual Host anbieten möchten. Auf einer Site habe ich Mailman laufen, einen Mailinglisten-Manager. Alle Mailman-URLs fangen mit `/mailman` oder `/pipermail` an. Nachdem Mailman korrekt eingerichtet und konfiguriert ist, habe ich folgende zwei Zeilen zur Konfiguration hinzugefügt, damit es schön funktioniert:

```
RewriteRule ^/mailman(.*) - [L]
RewriteRule ^/pipermail(.*) - [L]
```

Der einzige Haken hierbei ist, dass Sie keine Objekte zu Plone hinzufügen können, deren Namen mit Ihren Regeln kollidieren, z.B. Ordner mit ähnlichen Namen. In diesem Beispiel wären etwa `mailman`, `pipermail` oder `download` verboten, weil die Benutzer diese Objekte nie sehen würden. Mit dieser Methode könnten Sie den Zugriff auf bestimmte Teile Ihrer Site einschränken, aber ich rate Ihnen, dafür lieber die Sicherheitsmechanismen von Plone zu verwenden. Sonst verwaltet Plone den Inhalt nicht wirklich, d.h., es gibt keine Sicherheit, keinen Workflow und keine Metadaten dazu. Der Inhalt liegt völlig außerhalb von Plone. Eventuell kann das aber auch eine gute Lösung sein.

10.3.2 Eine Datei in Plone verwalten

CMFExternalFile ist ein Produkt, mit dem Sie Inhalte aus Plone heraus verwalten können, obwohl die Kerninhalte im Dateisystem liegen. Falls Sie ExternalFile und CMFExternal im oberen Teil dieses Kapitel schon installiert haben, dann sind Sie bereit. Wenn nicht, gehen Sie noch einmal zum ersten Abschnitt »Plone-Produkte installieren« zurück.

Nach deren Installation gehen Sie zur Plone-Schnittstelle zurück. Wenn Sie zu Plone gehen, werden Sie bemerken, dass Sie nun einen neuen Inhaltstyp namens *External File* hinzufügen können. Den Typ External File fügen Sie genauso hinzu wie eine normale Datei, was ich in Kapitel 3 beschrieben habe. Wenn Sie in den Programmcode schauen, werden Sie sogar festellen, dass die gleichen Templates verwendet werden.

Es gibt hierbei allerdings einen kleinen Unterschied. Die Datei wurde tatsächlich im Dateisystem hinzugefügt. Sie wird in ein neues Verzeichnis im `var`-Verzeichnis Ihrer Plone-Installation platziert. Wenn Sie nicht genau wissen, wo das ist, gehen Sie ins Control Panel, und suchen Sie nach dem Verzeichnis, das von der Instanzwurzel aufgelistet wird. Mein `var`-Verzeichnis befindet sich in `/var/zope/var`. Darin ist ein Verzeichnis namens `externalfiles`. In diesem werden alle Dateien erzeugt, die Sie in Plone hochladen. Wenn Sie sich das Verzeichnis anschauen, sollten Sie die Datei finden, die Sie hochgeladen haben.

Was Sie nun haben, ist eine hybride Speicherlösung, die die Datei im Dateisystem speichert und die Metadaten zu dem Objekt (Beschreibung, Stichwörter etc.) in Plone. Das ist besser als eine Lösung mit nur einem Webserver, weil die Inhalte über Sicherheit, Metadaten usw. verfügen. Wenn Sie wirklich wollten, könnten Sie durch die korrekte Konfiguration Ihres Webservers erreichen, dass der Inhalt von Apache ausgegeben wird, indem Sie das Verzeichnis aus dem `externalfiles`-Verzeichnis lesen.

10.3.3 FTP-Zugriff auf Plone

FTP ist eine gute Möglichkeit, Inhalte hoch- und herunterzuladen, um sie ohne einen Browser zu bearbeiten. Um FTP in Plone zu aktivieren, müssen Sie sicherstellen, dass es auf dem Server aktiviert ist. Gehen Sie noch einmal zu Kapitel 2, um zu sehen, wie Sie solche Dienste hinzufügen und bearbeiten können. Kurz gesagt: Vergewissern Sie sich, dass Ihre Zope-Konfigurationsdatei, `zope.conf`, Folgendes enthält:

```
<ftp-server>
    address 21
</ftp-server>
```

Hinweis

Wenn Ihr Server den Port 21 benutzt, müssen Sie sicherstellen, dass kein anderes Programm diesen Port belegt. Außerdem gilt für die meisten Unix-Systeme, dass Sie Ihren Dienst unter Root starten müssen, damit er das Recht hat, einen Port mit einer kleinen Nummer zu belegen. Dazu müssen Sie den effektiven Benutzer in Ihrer Zope-Konfigurationsdatei einstellen. Lesen Sie dazu den Abschnitt »Sicherheit auf Ihrem Server« in Kapitel 9.

Als Nächstes benötigen Sie einen FTP-Client, um auf den Server zuzugreifen. Unter Windows können Sie einfach den Internet Explorer benutzen, indem Sie die Adresse des Servers in der Adresszeile eingeben. Setzen Sie die Adresse auf den Pfad Ihres Zope-Servers (z.B. *ftp://localhost:8021/*), und Sie bekommen Zugriff auf die Objekte in Ihrer Site, wie in Abbildung 10.10 zu sehen ist.

Wenn Sie einen Benutzernamen und ein Passwort benötigen, um auf den Server zuzugreifen, dann müssen Sie diese in folgendem Format zur URL hinzufügen: *ftp://benutzer:passwort@localhost:8021/*. Es gibt viele andere FTP-Clients, bei denen Sie eine ausgefeiltere Schnittstelle haben können, wenn Sie möchten. Unter Linux sind ebenfalls viele FTP-Clients verfügbar, ob auf der Kommandozeile oder mit einer GUI, z.B. *gFTP* und *Konqueror*.

10.3.4 WebDAV-Zugriff auf Plone

WebDAV ist ein System zur Erstellung von Inhalten mit Systemen wie Plone, die HTTP benutzen. Damit können Sie einen Plone-Server auf ein Dateisystem abbilden. Um das zu aktivieren, müssen Sie die Zope-Konfigurationsdatei so bearbeiten, wie ich in Kapitel 2 beschrieben habe, damit die Datei `zope.conf` Folgendes enthält:

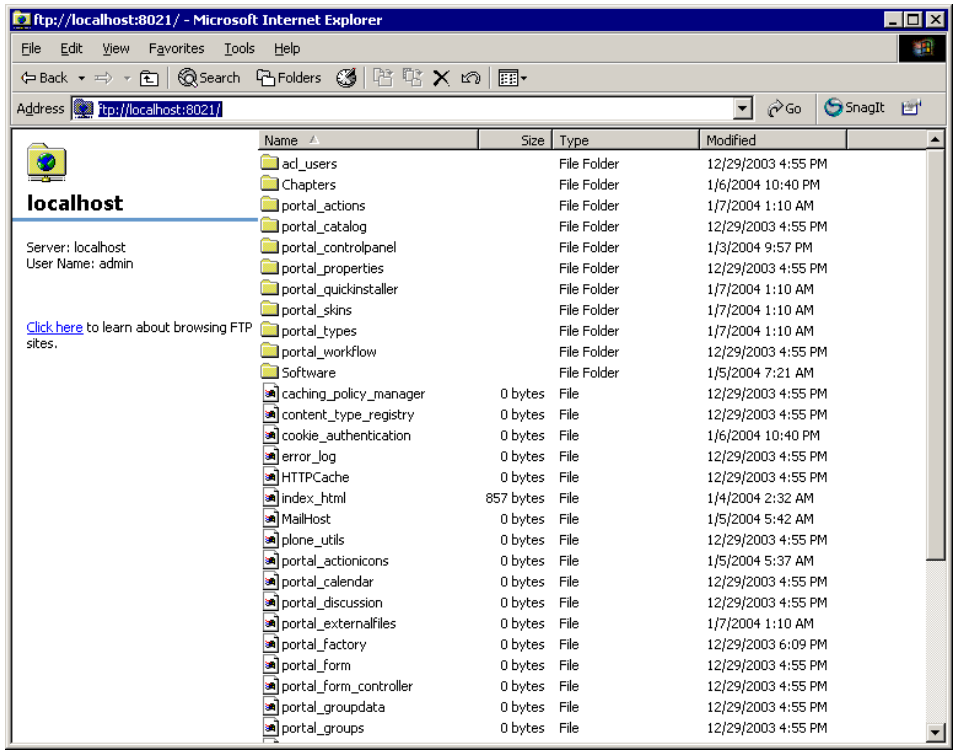


Abbildung 10.10: FTP-Zugriff im Internet Explorer

```
<webdav-source-server>
    address 1980
</webdav-source-server>
```

Für Windows gibt es das Programm *WebDrive* unter *http://www.webdrive.com/*, samt einer freien Testversion, die Sie ausprobieren können. Nach der Installation von WebDrive fügen Sie eine Verbindung zum Plone-Server hinzu, und dann müssen Sie nur noch aus dem Dateisystem direkt auf Ihr Plone zugreifen, indem Sie zum Windows Explorer gehen, wie in Abbildung 10.11 gezeigt wird.

Unter Unix können Sie *Cadaver* (*http://www.webdav.org/cadaver*) verwenden, einen Client auf der Kommandozeile mit vielen Funktionen. Nach der Installation von Cadaver können Sie eine Verbindung zur Plone-Site von der Kommandozeile aus herstellen. Beispiel:

```
cadaver http://192.168.2.1:8080/Members/Plone
```

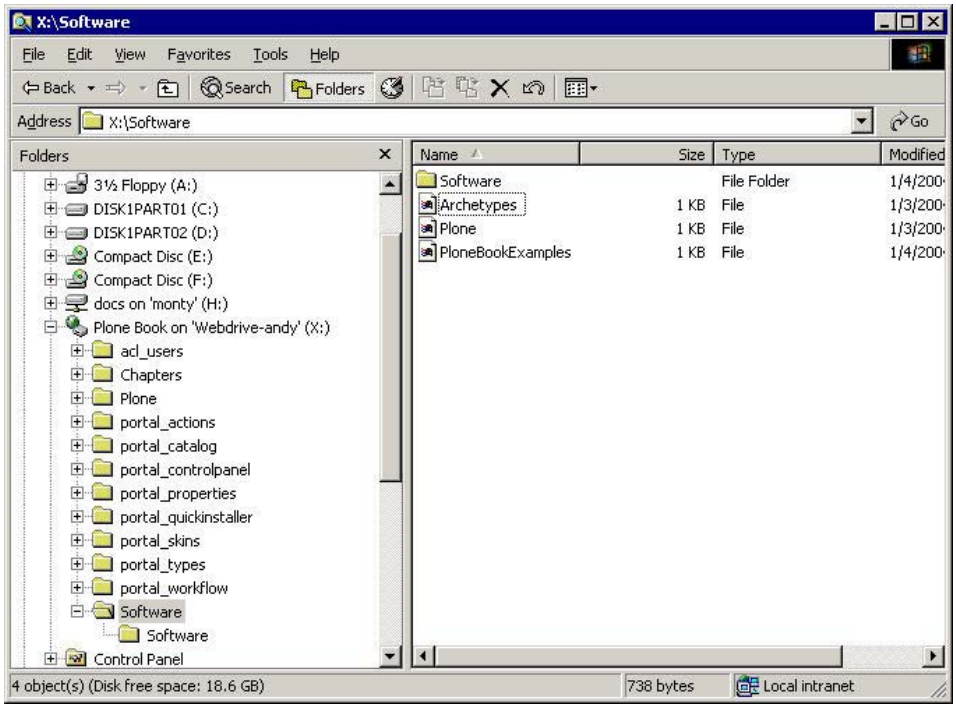


Abbildung 10.11: Zugriff auf Ihre Plone-Inhalte mit WebDrive

10.3.5 Inhalte mit erweiterten Editoren bearbeiten

Wie ich schon mehrfach betont habe, ist es keine gute Idee, Ihre Benutzer dazu zu zwingen, Inhalte in einem HTML-Textbereich schreiben und bearbeiten zu müssen. Manche Editoren haben dafür eine Lösung parat.

Einer davon ist *Epoz*, mit dem die Benutzer Dokumente direkt im Browser bearbeiten und ändern können, ohne etwas über HTML wissen zu müssen. Wenn Sie es mit vielen Benutzern zu tun haben, die Inhalte eingeben, können diese nach der Installation von Epoz HTML-Inhalte ändern, ohne dass sie HTML verstehen müssen. Für eine noch ausgefeiltere Bearbeitung können Sie *External Editor* verwenden, mit dem Sie Inhalte in einem lokalen Programm wie Microsoft Word bearbeiten können.

WYSIWYG-Editor im Browser

Die getestete Version von Epoz 0.7.4 finden Sie unter *http://zope.org/Members/mjablonski/Epoz/0.7.4*. Epoz verlangt einen modernen Browser, den die meisten Plone-Benutzer aber sowieso benötigen. Die erforderlichen Browser sind Internet Explorer 5.5+, Mozilla 1.3.1+ und Netscape 7.1+.

Laden Sie Epoz herunter, und installieren Sie es wie üblich. Danach müssen Sie Ihre persönlichen Plone-Voreinstellungen so ändern, dass Sie Epoz benutzen können. Melden Sie sich bei Plone an, klicken Sie auf MEINE EINSTELLUNGEN, und dann auf PERSONAL PREFERENCES. Öffnen Sie auf der MEINE EINSTELLUNGEN-Seite das TEXTEDITOR-Dropdown-Menü, wählen Sie die Option EPOZ, und klicken Sie dann auf SPEICHERN, um Ihre Änderungen zu bestätigen.

Nun haben Sie Ihren Editor gewählt und können zu einem beliebigen Dokument gehen, um dann auf den BEARBEITEN-Reiter zu klicken. Sie werden bemerken, dass das Feld *Haupttext* sich erheblich zu einem erweiterten Editor verändert hat. Der Editor sollte ziemlich selbsterklärend für Sie sein und vertraute Buttons wie B für fett (bold), I für kursiv (italics) usw. enthalten (siehe Abbildung 10.12).

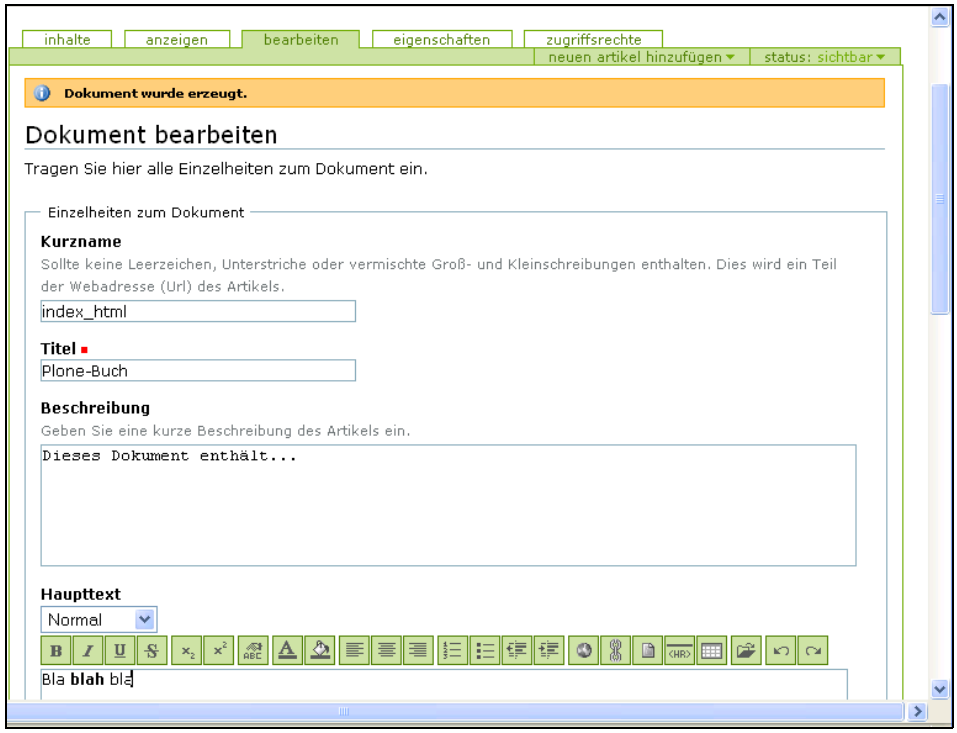


Abbildung 10.12: Bearbeiten eines Dokuments in Epoz

External Editor

External Editor ist ein Werkzeug, das Sie bei allen möglichen Plone-Inhalten, Templates und Code benutzen können. Damit können Sie in einer Plone-Site gespeicherte Objekte mit lokalen Programmen Ihrer Wahl bearbeiten. Sie können etwa ein in Ihrer Plone-Site gespeichertes Microsoft Word-Dokument lokal mit

Microsoft Word bearbeiten. Sobald Sie das Dokument speichern, wird es automatisch an Plone gesendet.

External Editor ist in den Plone-Installationspaketen enthalten und wird auf dem Server automatisch eingerichtet. Die Anwendung ist insofern ungewöhnlich, als sie aus zwei Komponenten besteht: einer für den Server und einer für *jeden* Client, der das Produkt benutzen möchte.

Installation des Server-Produkts

Es ist nicht notwendig, das Server-Produkt zu installieren, falls Sie bei der Installation Ihrer Plone-Site ein Installationsprogramm verwendet haben. Wenn das nicht der Fall ist, dann finden Sie das serverseitige Produkt unter *http://zope.org/Members/Caseman/ExternalEditor*. Installieren Sie das Produkt auf die übliche Weise, die ich am Anfang dieses Kapitels beschrieben habe, und starten Sie dann Ihr Zope neu.

Melden Sie sich dann in Plone als Administrator an, klicken Sie auf PLONE KONFIGURATION, und wählen Sie dann PORTAL EINSTELLUNGEN. Wählen Sie die Option EXTERNE EDITOREN ERMÖGLICHEN, um sicherzugehen, dass Sie Objekte mit diesem Werkzeug bearbeiten können.

Installation des Client-Produkts

Dieses Produkt müssen Sie auf jedem Client-Rechner installieren, der auf die Plone-Site zugreift. Genauso wie Sie in Ihrem Browser Flash oder QuickTime installieren würden, installieren Sie den clientseitigen Code von External Editor. In einem Intranet oder auf dem eigenen Rechner kann man das leicht machen, aber auf öffentlichen Rechnern kann das schon schwieriger sein.

Für Windows 2000 und XP laden Sie das ausführbare Windows-Installationsprogramm namens `zopeedit-win32-0.7.1.exe` herunter. Wenn Sie auf die Datei doppelklicken, geht es mit der grafischen Oberfläche weiter. Sie müssen nur alle Voreinstellungen auswählen. Dabei werden die Optionen für Internet Explorer ausgewählt. Um zu überprüfen, ob das auch funktioniert hat, machen Sie Folgendes:

1. Im Fenster ARBEITSPLATZ wählen Sie EXTRAS – ORDNEROPTIONEN.
2. Im Fenster DATEITYPEN scrollen Sie bis nach unten, wo Sie die Zope-Erweiterung sehen sollten (siehe Abbildung 10.13).

Unter Unix laden Sie das Archiv namens `zopeedit-0.7-src.tar.gz` herunter. Dieses müssen Sie dann auspacken und einrichten, wie in den Unix-Installationsanweisungen unter *http://zope.org/Members/Caseman/ExternalEditor/install-unix* beschrieben ist. Folgendes ist ein Beispiel mit der Version 0.7:

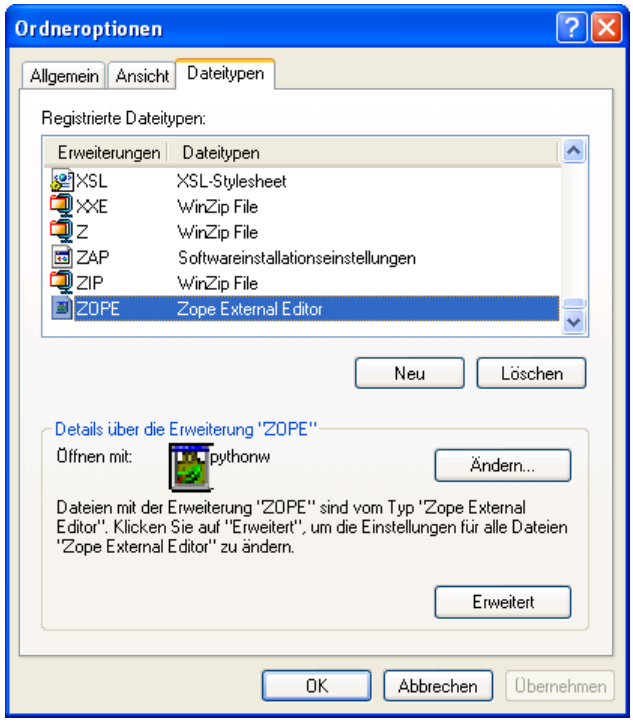


Abbildung 10.13: Dateityp-Konfiguration unter Windows

```
$ tar -zxf zopeedit-0.7.1-src.tgz
$ cd zopeedit-0.7.1-src
$ python setup.py install
...
```

Nachdem Sie den Client installiert haben, müssen Sie alle Browser konfigurieren, die Sie benutzen möchten. Anweisungen für Konqueror, Galeon und andere Browser sind online unter Zope.org verfügbar. Die folgende schrittweise Konfiguration gilt für Mozilla:

1. Wählen Sie BEARBEITEN – EINSTELLUNGEN.
2. Unter NAVIGATOR wählen Sie HILFSANWENDUNGEN.
3. Klicken Sie auf den NEUER TYP-Button.
4. Geben Sie eine Beschreibung ein, z.B. **Zope Editor**.
5. Als MIME-TYP geben Sie **application/x-zope-edit** ein.
6. Unter ÖFFNEN – DURCHSUCHEN wählen Sie die Helfer-Applikation PYTHON-DATEI.
7. Klicken Sie auf OK, und schließen Sie dann die Voreinstellungen.

External Editor öffnet einen Editor. Welcher Editor erscheint, hängt vom Inhalt einer Konfigurationsdatei ab. Um den Editor Ihrer Wahl aufzurufen, ändern Sie diese Datei. Sie können sie unter verschiedenen Namen – je nach Ihren Einstellungen – an folgenden Orten finden:

- Wenn Sie Plone unter Windows mit einem Installationsprogramm installiert haben, finden Sie die Datei unter `C:\Programme\Plone\Zope\pwi\zopeedit.ini`.
- Wenn Sie unter Windows das separate Installationsprogramm für External Editor verwendet haben, finden Sie die Datei in dem Verzeichnis, wo Sie External Editor installiert haben. Standardmäßig ist das `C:\Programme\ZopeExternalEditor\zopeedit.ini`.
- Unter Unix heißt die Datei `.zope-external-edit` und befindet sich im Home-Verzeichnis des Benutzers, der das Programm ausführt, z.B. `/home/andy/.zope-external-edit`. Es ist deswegen im Home-Verzeichnis des Benutzers, weil jeder Benutzer eventuell eigene Einstellungen hat.

Diese Datei enthält eine Abbildung von Erweiterungen auf den gestarteten Editor. Um z.B. den Editor für Page Templates zu ändern, finden Sie die folgenden Zeilen mit `meta-type:Page-Template`:

```
[meta-type:Page Template]
extension=.pt
```

Sie könnten z.B. *Scite* benutzen, einen freien Texteditor. Um ihn für Page Templates zu verwenden, müssen Sie die Datei wie folgt ändern:

```
[meta-type:Page Template]
extension=.pt
editor=scite
```

Damit External Editor funktioniert, muss jeder Aufruf des Editors einen eigenen Prozess starten. Das heißt, dass das External Editor-Client-Programm diesen Prozess überwachen kann, um zu sehen, wann er beendet wird. Für manche Editoren, die versuchen, mehrere Dateien im gleichen Prozess zu öffnen, ist das ein Problem. Um z.B. VIM in KDE zu laden, müssen Sie eine separate Shell wie folgt ausführen:

```
editor=konsole -e vim
```

Ein Word-Dokument bearbeiten

Die Bearbeitung eines Microsoft Word-Dokuments kann man tatsächlich ziemlich einfach einrichten. Sie brauchen auf Ihrem lokalen Rechner nur ein installiertes Microsoft Word. Laden Sie Ihr Microsoft Word-Dokument in Plone als Standarddatei hoch, und zeigen Sie die Datei dann in Plone an. Klicken Sie auf das

Icon mit dem kleinen Bleistift in der oberen rechten Ecke Ihrer Seite. Dann wird auf Ihrem Rechner Microsoft Word gestartet, und das Dokument vom Server wird angezeigt. Nun können Sie den Inhalt nach Belieben bearbeiten, und bei einem Klick auf SPEICHERN wird die Datei automatisch in Plone gespeichert.

Page Templates über External Editor bearbeiten

Um ein Page Template zu erstellen, verwenden Sie das ZMI. Wenn Sie den Ordner mit dem Inhalt des Page Templates anzeigen, sehen Sie rechts vom Objekt bestimmt ein zusätzliches Icon mit einem Bleistift. Ein Klick auf diesen Bleistift aktiviert External Editor und öffnet das Page Template im gewählten Editor. Sie müssen nur einen guten Editor finden, mit dem Sie in Page Templates schreiben können. Da Page Templates lediglich aus XHTML (Extensible HTML) bestehen, benutze ich einen einfachen Editor, der XML (Extensible Markup Language) unterstützt. Die folgenden Abschnitte behandeln zwei Beispiel-Editoren: *Dreamweaver* und *HTML-Kit*.

Dreamweaver MX

Ändern Sie den Teil `[meta-type:Page Template]` der Konfigurationsdatei so, dass er auf Dreamweaver zeigt. In meiner Installation sieht das wie folgt aus:

```
[meta-type:Page Template]
extension=.pt
editor=C:\Programme\Macromedia\Dreamweaver MX\Dreamweaver.exe
```

Ein Klick auf das Bleistift-Icon zur Bearbeitung in External Editor öffnet es nun direkt in Dreamweaver, wie in Abbildung 10.14 zu sehen ist. Leider öffnet Dreamweaver nicht jede Datei in einer separaten Instanz, d.h., Sie können nicht mehr als eine Datei gleichzeitig bearbeiten.

HTML-Kit

HTML-Kit ist ein freier und mächtiger HTML-Editor und die bevorzugte Wahl vieler Plone-Entwickler. Um HTML-Kit mit External Editor zu benutzen, ändern Sie Ihre Konfigurationsdatei so, dass sie auf HTML-Kit zeigt. In meiner Installation sieht das wie folgt aus:

```
[meta-type:Page Template]
extension=.pt
editor=C:\Programme\Chami\HTML-Kit\Bin\HTMLKit.exe
```

Beim Klick auf das Bleistift-Icon zur Bearbeitung in External Editor wird nun direkt HTML-Kit gestartet. Sie können auch eine Einstellung ändern, mit der alle Dateien in einem separaten Prozess geöffnet werden. Wählen Sie EDIT – PREFERENCES – STARTUP, und aktivieren Sie LIMIT TO A SINGLE HTML-KIT INSTANCE. Nun werden alle Dateien in einem neuen Prozess geöffnet.

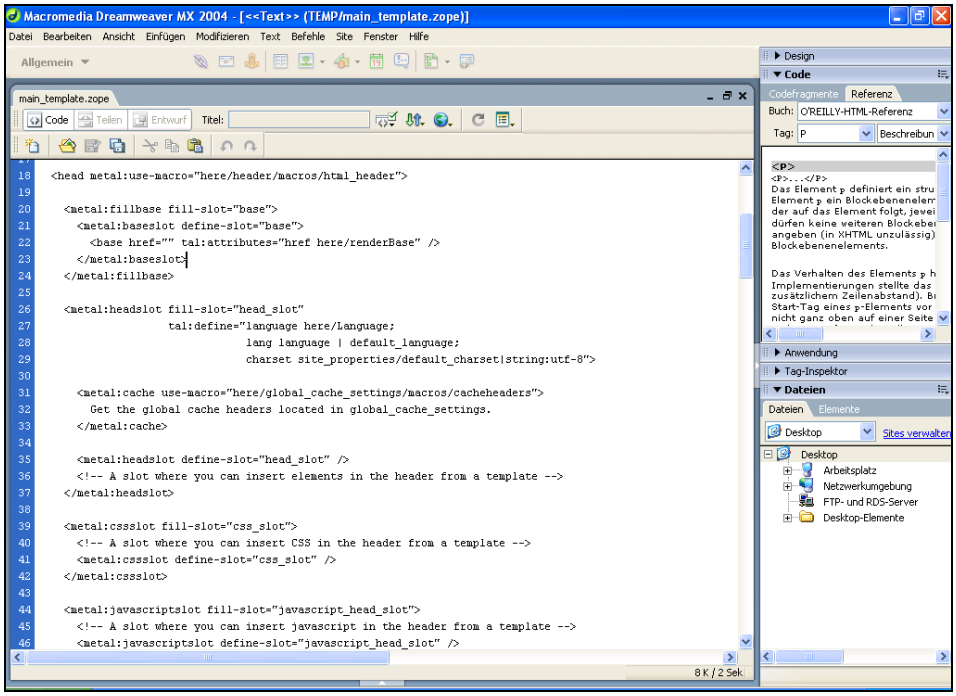


Abbildung 10.14: Page Templates mit Dreamweaver bearbeiten

11 Inhaltstypen manipulieren und kategorisieren

In diesem Buch habe ich Ihnen gezeigt, wie Sie Inhalte zu Ihrer Site hinzufügen können, und ich habe die in Plone enthaltenen Inhaltstypen wie Dokumente, Bilder usw. beschrieben. Bisher waren Sie allerdings auf genau diese Inhaltstypen eingeschränkt und auf solche, die Sie in Produkten aus dem Internet gefunden haben. Der mächtigste Teil von Plone, die Manipulation dieser Inhaltstypen, bildet das Hauptthema dieses Kapitels.

In diesem Kapitel vergleiche ich die verschiedenen Objekttypen in Plone miteinander. Dadurch erhalten Sie einen Einblick in manche Entwicklungstaktiken für Ihre eigenen Projekte. Dann behandle ich Inhaltstypen und zeige, wie sie in Plone registriert werden. Diese Registrierung bildet die Basis für die Anpassung der Typen an das von Ihnen gewünschte Format. Anschließend fahre ich mit der Kategorisierung von bzw. der Suche in Inhalten fort – Aufgaben, von denen Sie sicher wissen möchten, wie man sie ausführt. Mit diesem Wissen werden Sie wichtige Entscheidungen darüber treffen können, wie Sie Ihre Site entwickeln und neue Inhaltstypen erstellen können.

Lassen Sie mich jetzt also Ihren Appetit auf die Manipulation von Inhaltstypen wecken. Sobald Sie einen völlig neuen Inhaltstyp anpassen können, können Ihre Benutzer fast alles erstellen und bearbeiten, was Sie nur möchten! Zum Beispiel:

- Benutzer können ein Bild einer Zellkultur hochladen, das mit gewissen Bildbearbeitungsbibliotheken zerschnitten und manipuliert wird und dann dem Benutzer in einem bestimmten Format präsentiert wird.
- Benutzer können eine MP3-Audiodatei hochladen, daraus den Titel und Künstlernamen extrahieren und sie in Plone platzieren.
- Sie können einen vollständigen E-Commerce-Shop erstellen, in dem Plone-Benutzer Elemente wie z.B. Kleidungsstücke zum Verkauf anbieten, mit Informationen zu Versandkosten, Größe und Garantie.
- Benutzer können ein Microsoft Word-Dokument hochladen und dann so manipulieren, dass gewisse Teile davon ausradiert werden. Benutzer mit einer niedrigen Sicherheitsstufe können dann nur die Dokumente mit ausradierten Passagen sehen.

All diese Möglichkeiten und noch mehr stehen Ihnen in Plone zur Verfügung! Es gibt wirklich nur wenige Einschränkungen. Aus diesem Grund ist Plone wahrscheinlich eines der erweiterbarsten und flexibelsten Frameworks, die verfügbar sind. Die einzige wirkliche Einschränkung liegt in Ihrer Fähigkeit, in Python zu programmieren (oder darin, sich jemanden zu leisten, der es in Ihrem Auftrag macht).

In diesem Kapitel behandle ich daher Inhaltstypen im Detail, und dazu zählt auch, wie man sie über das Web registriert und manipuliert. Für die folgenden Abschnitte wird zwar kein Wissen über Python explizit verausgesetzt, aber ich empfehle Ihnen dennoch, sich wenigstens damit vertraut zu machen. Dieses Kapitel enthält auch Informationen über Formulare und darüber, wie man sie validiert.

Wenn Sie eigene Inhaltstypen entwickeln möchten, sollten Sie dieses Kapitel unbedingt lesen, unabhängig davon, ob Sie über das Web oder in Python entwickeln, denn dieses Kapitel behandelt die Bereiche, die man kennen muss, um die Registrierung von Inhaltstypen zu verstehen.

Das nächste Kapitel setzt diese Reise dann fort und bringt Ihnen die wirklich blutigen Details bei der Entwicklung von Inhaltstypen mit Python näher. Danach werden Sie *Archetypes* verwenden, um das Gleiche mit einem Zehntel an Aufwand zu machen. Mit Archetypes werden Sie dann einige wirklich coole und abgefahrene Sachen machen. Aber jetzt geht es zuerst einmal direkt zu Inhaltstypen!

11.1 Übersicht zu Inhaltstypen

Ich muss gestehen, ich habe in diesem Buch gewisse Begriffe verwendet, die ich nur oberflächlich erklärt habe. Daher wird es jetzt Zeit, das nachzuholen, damit Sie diese besser verstehen können. Folgende Konzepte sind von Bedeutung:

- **Inhaltstyp (Content Type)**: Das ist eine Art von Inhalt, der in einer Plone-Site registriert ist. Normalerweise, wenn auch nicht immer, ist ein Inhaltstyp etwas, das mit Hilfe der Plone-Schnittstelle von Benutzern einer bestimmten Sicherheitsstufe hinzugefügt und bearbeitet werden kann. Es wird empfohlen, in Plone Inhaltstypen wie Dokumente, Dateien und Bilder voneinander zu trennen. Diese Trennung von Inhalten in verschiedene Typen ist ein grundlegendes Konzept in Plone.
- **Element und Objekt**: Diese Begriffe beziehen sich auf die eigentliche Instanz von irgendetwas. Es sind sehr überladene Begriffe, d.h., ihre Definition ist normalerweise stark kontextabhängig. Bisher habe ich in diesem Buch diese Begriffe bei einer bestimmten Instanz eines Inhaltstyps benutzt, z.B. bei einem

bestimmten Dokument oder Bild. Von nun an werde ich den spezifischen Begriff, nämlich *Inhaltstyp* verwenden, wenn es um Inhaltstypen geht.

- **Werkzeug (Tool)**: Ein Werkzeug ist ein Dienst, der in einer Plone-Site vorhanden ist. Auf jeder Plone-Site kann es nur eine Instanz eines Werkzeugs geben. Ein Werkzeug macht von sich aus gar nichts, und externe Benutzer der Anwendung werden nie mitbekommen, wie viele Werkzeuge vorhanden sind oder was sie machen. Aber Inhaltstypen oder Anfragen von Benutzern interagieren mit den Werkzeugen. Einige wichtige Werkzeuge haben Sie bereits gesehen, z.B. `portal_workflow`, `portal_skins` und `portal_actions`.
- **Zope-Objekt**: Das ist ein Objekt, das in Zope lebt. Es bietet den Benutzern eine bestimmte Funktionalität, und man kann über das ZMI (Zope Management Interface) darauf zugreifen. Aber es ist nichts, worauf Plone selbst zugreifen oder was es steuern würde. Wenn Sie zu einer Plone-Site gehen und auf das ZMI zugreifen, werden Sie eine große Anzahl von Objekten im ZMI sehen. Ein Werkzeug ist ein solches Objekt, die Plone-Site ist ein anderes, und die CacheManager sind weitere. Zwischen diesen Objekten gibt es eine Menge an Überlappungen. Plone enthält z.B. einen Inhaltstyp für Bilder und Zope verfügt über ein Bildobjekt. Beide haben ähnliche Aufgaben und funktionieren auf ähnliche Weise, aber Plone kann nur auf eines zugreifen.

Hinweis

Obwohl alles in einer Plone-Site in der Zope-Objektdatenbank (ZODB) ein *Zope-Objekt* ist, benutze ich diesen Begriff, um Objekte zu beschreiben, die keine Werkzeuge oder Instanzen eines Inhaltstyps sind.

11.1.1 Wann man Inhaltstypen erstellen sollte

Also gut, Sie bauen in Plone Ihre Superanwendung, die Ihnen Ruhm und Ehre und noch dazu ein sicheres Einkommen einbringen wird. Wie werden Sie sie aber strukturieren, und was bauen Sie überhaupt? Nun, das hängt davon ab, was Sie machen wollen, und davon, wie Sie ihre Entwicklung bisher eingeteilt haben. Die folgenden Fragen helfen Ihnen möglicherweise bei der Entscheidung:

- **Ändern Sie einfach nur Skins und einfache Verhalten, z.B. Portlets?** Sie können in einer Skin fast alles machen, was Sie wollen, außer ein Werkzeug oder einen Inhaltstyp schreiben. Wenn Sie wirklich wollen, können Sie alle Cascading Stylesheets (CSS), Templates und Scripten ändern, die in einer Plone-Site enthalten sind.

- **Werden Mitglieder Ihrer Site mehrere Kopien eines Elements hinzufügen?** Wenn ja, dann möchten Sie wahrscheinlich einen Inhaltstyp dafür schreiben.
- **Ist das ein Dienst, den andere Inhaltstypen benutzen könnten?** Wenn ja, dann möchten Sie wahrscheinlich ein Werkzeug schreiben.
- **Möchten Sie mehrere Kopien von etwas, wollen aber nicht, dass es Mitglieder Ihrer Site erstellen und bearbeiten können?** Wenn ja, dann möchten Sie wahrscheinlich ein Zope-Objekt. Aber Sie sollten noch einmal genau darüber nachdenken, was Sie machen.

Normalerweise wird eine Anwendung in mehrere Teile aufgeteilt: ein bis zwei Werkzeuge und ein oder zwei Inhaltstypen. Kapitel 12 behandelt die Erstellung eines Inhaltstyps, der einen Source Code, z.B. einen Python-Schnipsel, nimmt und die Syntax des Codes hervorhebt. Falls Sie diese Syntax-Hervorhebung an anderen Stellen brauchen können, dann könnten Sie sie in ein Werkzeug verwandeln, das von mehreren Inhaltstypen verwendet werden kann. Werkzeuge sind, kurz gesagt, die beste Art, um Funktionalität zu einer Site statt zu einem bestimmten Inhaltstyp hinzuzufügen.

Die Definition der Erstellung eines Inhaltstyps wird normalerweise von den Anforderungen der Benutzer diktiert, die diese Objekte hinzufügen, bearbeiten und steuern müssen. Es kann verlockend sein, mit der Erstellung eines Inhaltstyps für jede Art von Objekt zu beginnen, aber wie bei jeder Art von Entwicklung müssen Sie vorsichtig sein. Wäre es möglich, einen statt zwei Inhaltstypen mit minimalen Unterschieden zu benutzen? Das Wissen darüber, wie man so etwas konfiguriert, kommt mit zunehmender Erfahrung, aber die folgenden paar Kapitel werden bestimmt eine Hilfe sein.

11.1.2 Inhaltstypen konfigurieren

Nun enthält Ihre Plone-Site also Inhaltstypen, aber woher weiß die Plone-Site, wie diese konfiguriert werden? Die Antwort lautet, dass Attribute, Methoden, Sicherheit und Skins für alle Inhaltstypen im Dateisystem in Python und im entsprechenden Code definiert werden. Diese Information genügt Plone, um zu wissen, wie das Produkt benutzt wird. Wie Sie gesehen haben, ist die einzige Ausnahme hiervon der Workflow, der normalerweise außerhalb des Inhaltstyps definiert wird. Manche Produkte haben ihren eigenen Workflow, der als Verhalten zum Inhaltstyp hinzugefügt wird.

In Kapitel 10 habe ich gezeigt, wie Inhaltstypen in zwei Schritten in Plone installiert werden: Zuerst wird das Produkt in Zope installiert, dann wird der Inhaltstyp in *jeder* Plone-Instanz installiert. Beim zweiten Schritt werden Angaben über den Inhaltstyp installiert, die aus dem Dateisystem stammen und dann in Ihrer Plone-Site installiert werden.

Warum besteht dieser Vorgang aus zwei Phasen? In der zweiten Phase wird eine lokale Kopie des Produkts in Ihrer Plone-Site angelegt, und nun können Sie ändern, wie sich der Inhalt bei Ihnen verhält. Möchten Sie, dass ein Dokumentobjekt andere Reiter oben hat? Oder soll ein Dokumentobjekt anders manipuliert werden, anders aussehen oder sogar völlig anders benannt werden? Kein Problem! Nun können Sie Ihre Plone-Instanz über das Web ändern.

Dieser Ansatz ist der gleiche wie bei `portal_skins`, in dem Sie eine Skin in Ihrer lokalen Instanz anpassen können. Wenn es im Produkt zu Änderungen kommt und Sie eine neue Version von Plone installieren, betreffen diese Änderungen das Dateisystem. Aber nun können Sie diese Änderungen herunterladen und installieren. Weil Sie die Angaben in Ihrer Datenbank gemacht haben, behalten Sie diese angepasste Version.

Jeder Inhaltstyp in Plone verfügt über eine Einstellung im Werkzeug `portal_types`. Auch wenn jeder Inhaltstyp nur eine Einstellung im Werkzeug `portal_types` hat, kann es zu diesem Typ eine unbeschränkte Anzahl von Objekten in Ihrer Datenbank geben. Die Konfiguration wird bei Bedarf zu Rate gezogen, d.h., wenn Sie die Konfiguration ändern, aktualisieren Sie alle Objekte dieses Typs in der Datenbank.

11.1.3 Registrierung von Inhaltstypen im Werkzeug portal_types

Gehen Sie im ZMI zum Werkzeug `portal_types`, um auf die Registrierungsinformation zuzugreifen. Dort erhalten Sie eine Liste aller in dieser Plone-Site registrierten Inhaltstypen. Die meisten davon sind als etwas zu erkennen, das Sie mit wenigen Ausnahmen über die Plone-Schnittstelle hinzufügen könnten, z.B. Plone Site, TempFolder usw.

Jedes dieser Objekte ist eine Instanz mit Angaben zum Factory-Typ, was der Name eines bestimmten Konfigurationstyps ist. Klicken Sie auf ein beliebiges dieser Objekte, um auf die Typinformation zuzugreifen. Wenn Sie z.B. ein Ereignis anklicken, erhalten Sie eine lokale Kopie der Information zu dem Inhaltstyp. Über das Web können Sie Ihre Konfiguration ändern. In diesem Formular gibt es folgende Einträge:

- **Title**: Der Titel des Inhaltstyps.
- **Description**: Die Beschreibung, die zu diesem Inhaltstyp erscheint. Diese wird benutzt, wenn Sie zu den Ordnerinhalten gehen und NEUEN ARTIKEL HINZUFÜGEN anklicken, ohne einen Inhaltstyp anzugeben. Dann erscheint eine Liste aller Inhaltstypen und ihrer Beschreibungen.
- **Icon**: Die ID des Icons, das für diesen Inhaltstyp benutzt wird.

- **Product metatype**: Der Metatyp für diesen Inhaltstyp. Dieser bildet den Plone-Inhaltstyp auf den Zope-Metatyp ab.
- **Product name**: Der Name des Produkts, in dem dieser Metatyp definiert ist.
- **Product factory method**: Die Methode, die von der Produkt-Factory aufgerufen wird, um diesen Inhalt zu erstellen.
- **Initial view name**: Wird in Plone nicht benutzt.
- **Implicitly addable**: Gibt an, ob dieser Inhalt zu Plone hinzugefügt werden kann. Wenn dieser Eintrag ausgewählt ist, kann der Inhalt hinzugefügt werden, außer es ist explizit etwas anderes angegeben.
- **Filter content types**: Falls dieser Inhaltstyp ein Ordner ist, aktivieren Sie das, um die Inhaltstypen zu filtern, die von Benutzern zu diesem Objekt hinzugefügt werden können.
- **Allowed content types**: Falls dieser Inhaltstyp andere Elemente enthalten kann und *Filter content types* aktiviert ist, sind nur die in dieser Liste angegebenen Inhaltstypen erlaubt.
- **Allow discussion**: Setzt den voreingestellten Status für Diskussionen bei allen Inhaltstypen. Falls dieser Eintrag aktiviert ist, können Benutzer den Inhalt diskutieren. Welche Benutzer das tun können, hängt von dem Recht *Discuss content* ab.

Nun werden Sie sich einige Aspekte dieser Registrierungsangaben detaillierter anschauen, wobei ich auch einige Beispiele zeige.

Wie ändert man das Icon eines Inhaltsyps?

Wenn Sie z.B. das Icon nicht mögen, das bei einem Inhaltstyp erscheint, müssen Sie lediglich ein neues Bild hochladen und sicherstellen, dass der Wert für das Icon im zuvor beschriebenen Formular gesetzt ist. Icons funktionieren am besten, wenn sie einen transparenten Hintergrund haben und 16 mal 16 Pixel groß sind.

Klicken Sie auf PORTAL_SKINS und CUSTOM, und fügen Sie dann ein neues Bild hinzu. Im Werkzeug `portal_types` setzen Sie dann den Wert für das Icon auf den Wert der ID des hochgeladenen Objekts. Um zu testen, ob das Icon sich verändert hat, gehen Sie zur Plone-Schnittstelle und schauen sich dort um, wo das Objekt erscheinen könnte. Führen Sie z.B. eine Suche durch, oder sehen Sie im Formular zur Erstellung von Inhalten nach.

Aktionen

Wenn Sie sich die Konfiguration von Inhaltstypen in `portal_types` anschauen, sehen Sie einen ACTIONS-Reiter. Diese Aktionen können auf den Inhaltstyp angewendet werden. Aktionen haben Sie schon kurz in Kapitel 4 gesehen, das eine detaillierte Liste dessen enthält, was unter diesem ACTIONS-Reiter vorkommt.

Erinnern Sie sich daran, dass eine *Aktion* die Möglichkeit bietet, eine Liste von Angaben zu speichern, die leicht bearbeitet und auf die dann unter verschiedenen Bedingungen zugegriffen werden kann. Im Plone-Portal werden Aktionen oben auf den Seiten mit blauen Reitern dargestellt. Zu jedem Inhaltstyp erscheinen die Aktionen als grüne Reiter in der Seitenmitte.

Wie Sie gesehen haben, werden Aktionen in Werkzeugobjekten gespeichert. Viele Werkzeuge enthalten Aktionen, aber es gibt keine gute Möglichkeit, den Ort einer Aktion in Erfahrung zu bringen. Wenn Sie auf Ihrer Plone-Site eine bestimmte Aktion ändern möchten, müssen Sie zuerst das entsprechende Werkzeug finden, in dem sie gespeichert ist.

Sobald Sie diese Aktion gefunden haben, können Sie sie nach Belieben anpassen. Wenn Sie z.B. eine neue Aktion als grünen Reiter zu einem Dokument hinzufügen möchten, müssen Sie zunächst den richtigen Ort finden. Zu Glück sind die folgenden Tipps recht hilfreich beim Finden von Aktionen:

- Wenn Sie eine Aktion zu einem Stück Inhalt suchen, z.B. Anzeigen oder Bearbeiten, dann befindet sie sich im entsprechenden Inhaltstyp im Werkzeug `portal_types`.
- Wenn Sie nach einer Site-Aktion suchen, finden Sie sie im Werkzeug `portal_action`.
- Sollten Sie die Aktion immer noch nicht finden, sehen Sie in einem verwandten Werkzeug nach. So befinden sich z.B. Registrierung und Anmeldung in `portal_membership`.
- Wenn Sie nach allen vorherigen Tipps die Aktion weiterhin nicht finden können, gehen Sie zu `portal_actions`, um die Liste von Werkzeugen zu sehen, und sehen Sie dort bei allen Anbietern von Aktionen nach.

Plone sucht auf folgende Art und Weise nach Aktionen:

- Bei einem Objekt werden alle Aktionen abgefragt.
- Bei jeder Aktion werden die Eigenschaften `conditions`, `permissions` und `visible` geprüft. Wenn sie durchgehen, wird die Aktion zurückgegeben.
- Jede Aktion wird in der Benutzerschnittstelle angezeigt, normalerweise in Form von Reitern am Anfang des Inhalts oder oben auf der Seite.
- Die URL dieser Aktion ist die URL des Objekts, bei der die eigentliche *Aktion* am Ende angefügt wird.

Bei einem Dokument unter *http://localhost.com/Plone/Document123* wäre z.B. die URL zum Bearbeiten *http://localhost.com/Plone/Document123/document_edit_form*. Hierbei sollte Ihnen ein wichtiges Sicherheitsproblem auffallen: Die Werte der Eigenschaften `conditions`, `permissions` und `visible` beziehen sich auf die Anzeige

der Aktion in der Liste der Aktionen. Mit anderen Worten: Wenn ein Benutzer wirklich will, kann er die URL ändern und *http://localhost.com/Plone/Document123/document_edit_form* eingeben, selbst dann, wenn die Rechte an der Aktion das gar nicht erlauben. Aus diesem Grund sollten Sie immer mit Rechten an den Aktionen arbeiten, die ausgeführt werden. Als Benutzer, der ein Objekt anzeigen, aber nicht bearbeiten kann, können Sie trotzdem die URL ändern, um zum BEARBEITEN-Formular zu gelangen. Bis hierher ist noch kein echter Schaden entstanden, da nach dem Abschicken des Formulars die Sicherheit neu überprüft und Ihnen dann die Erlaubnis verweigert wird.

Normalerweise werden Aktionen in Plone als Reiter angezeigt. Da sie aber durch ein Programm aufgerufen werden können, können sie auf beliebige Weise genutzt werden. Um eine Aktion durch ein Programm zu benutzen, rufen Sie die Methode `listFilteredActionsFor` im Werkzeug `portal_actions` auf. Bei einem gegebenen Objekt erhalten Sie damit für alle Aktionen eines Objekts ein Python-Dictionary mit Kategorien als Schlüssel:

```
actions = context.portal_actions.listFilteredActionsFor(object)
```

Damit erhalten Sie Folgendes:

```
{'site_actions': [
  {'category': 'site_actions', 'name': 'Small Text',
  'url': "javascript:setActiveStyleSheet('Small Text', 1);",
  'visible': 1, 'id': 'small_text',
  'permissions': ('View',)
   },
... und so weiter
```

Die grünen Reiter am oberen Rand sind eine Kombination zweier Kategorien: `object` und `object_tabs`. Die von der Methode zurückgegebenen Aktionen sind ein Python-Dictionary, dessen Schlüssel die Gruppe der Kategorie für diese Aktion sind. Um also nur an das Aktionsobjekt für eine Kategorie zu kommen, z.B. alle Aktionen in der Kategorie `object`, können Sie auf nur diesen Schlüssel des Dictionarys zugreifen. So erhalten Sie z.B. mit `actions["object"]` eine Liste aller dieser Aktionen:

```
{'category': 'object',
'name': 'Contents',
'url': ' http://localhost:8080/Plone/folder_contents',
'visible': 1,
'id': 'folderContents',
'permissions': ('List folder contents',)},
... und so weiter
```

Sie werden bemerken, dass, solange Sie das zu untersuchende Objekt angeben, das Werkzeug `portal_types` verwendet wird, um alle Aktionen für Ihren bestimmten `portal_type` ebenso wie andere relevante Aktionen zu finden.

Wenn Sie einen neuen Reiter zu einem Inhaltstyp hinzufügen möchten, müssen Sie lediglich zu `portal_types` gehen, dort den Inhaltstyp anklicken und den ACTIONS-Reiter wählen. Dann fügen Sie Ihre Aktion hinzu. Wenn die Aktion für den Inhaltstyp als grüner Reiter erscheinen soll, dann müssen Sie sicherstellen, dass die Kategorie `object_tabs` lautet.

Andere Objekte im Werkzeug portal_types

Wenn Sie sich das Werkzeug `portal_types` anschauen, werden Sie wahrscheinlich bemerken, dass Sie zu dem Ordner andere Objekttypen hinzufügen können, z.B. *DTML Method*, *External Method*, *Script (Python)* und *Scriptable Type Information*. Die ersten drei dieser Optionen sollen Unterstützung für die letzte Option in der Liste, *Scriptable Type Information*, bieten.

Mit *Scriptable Type Information* können Sie einen Typ mit eigenen Erzeugungsrechten und einem eigenen Erzeugungsscript über das Web definieren, anstatt dass diese für Sie definiert werden. Das kommt eventuell dann in Frage, wenn die vorgegebenen Rechte bei einem Inhaltstyp nicht ausreichen. Diese Option scheint zwar sehr nützlich, aber dennoch habe ich noch nie einen guten Anwendungsfall für *Scriptable Type Information* gegenüber der normalen Factory-basierten Typinformation gesehen, denken Sie also nicht zu viel darüber nach.

11.1.4 Speichern von Inhaltstypinformationen im Dateisystem

Nun haben Sie gesehen, wie diese Information in Zope gespeichert wird, aber natürlich kommt sie irgendwoher aus dem Dateisystem. Diese Information wird normalerweise im Produkt in einem Dictionary gespeichert, das üblicherweise `factory_type_information` heißt. Listing 11.1 zeigt diese Factory-Information zu dem Produkt *Folder*, einem Produkt, das Ordner in Plone anzeigt. Sie stammt aus der Datei `PloneFolder.py` im Verzeichnis `CMFPlone`.

Listing 11.1: Factory-basierte Typinformation

```
factory_type_information = (
    'id':'Folder',    'meta_type':'Plone Folder',
    'description':"""\
Plone folders can define custom 'view' actions,\
 or will behave like directory listings without one defined.""",
    'icon':'folder_icon.gif',
    'product':'CMFPlone',    'factory':'addPloneFolder',
    'filter_content_types':0,
```

```
    'immediate_view':'folder_listing',    'actions':
      ( {
           'id':'view',           'name':'View',
           'action':'string:${folder_url}/',           'permissions':
(CMFCorePermissions.View,),           'category':'folder',         }
    ...
      )
    }
```

Das Python-Dictionary bildet die Formulare ab, die Sie in der Plone-Schnittstelle gesehen haben. So ist z.B. `'meta_type': 'Plone Folder'` der `meta_type` des Produkts und erscheint in diesem Feld. Die Aktionen erscheinen als Liste von Dictionaries zu jeder Aktion, und es sind wiederum einfache Schlüssel/Wert-Paare für alle Eigenschaften einer Aktion. Hier habe ich nur die erste Aktion, *View*, angegeben, aber mittlerweile sollte Ihnen diese Art von Angaben vertraut vorkommen.

11.1.5 Einen neuen Inhaltstyp aus einem vorhandenen Typ erstellen

Unter *Umwidmen* versteht man die Erstellung mehrerer, leicht verschiedener Kopien des gleichen Typs, ausgehend von der Information eines vorhandenen Inhaltstyps. Eine Umwidmung kann dann eine schnelle und einfache Lösung sein, wenn Sie z.B. einen Typ erstellen möchten, der fast, aber nicht ganz, mit einem Nachrichtenelement identisch ist.

Der große Nachteil dieses Ansatzes ist der, dass Sie außer den Aktionen, Skins und einigen Inhaltstyp-Einstellungen nicht wirklich viel ändern können. Bevor Sie also auf diesem Weg weitermachen, machen Sie sich bitte klar, dass Sie auf diese Punkte beschränkt sind, d.h., Sie können z.B. keine neuen Felder oder Attribute hinzufügen. Auf der Mailing-Liste habe ich viele E-Mails gesehen, in denen ungefähr Folgendes steht: »Ich habe schon so viel gemacht, aber nun will ich die Attribute meiner Pressemitteilung ändern.« Nehmen Sie das als Warnung: Das geht nicht! Wenn Sie mehr machen möchten, sehen Sie sich die nächsten beiden Kapitel an, in denen steht, wie Sie eigene Inhaltstypen schreiben können.

Angenommen, Sie möchten einen Typ Pressemitteilung haben, der einer Nachricht ähnelt, aber auch noch Folgendes macht:

- Er hat den Namen *Pressemitteilung* in der Dropdown-Liste.
- Er hat ein anderes Icon.
- Er hat einen anderen Workflow als Nachrichtenelemente.
- Er wird anders angezeigt.

- Er hat die gleichen Datenstrukturen wie ein Nachrichtenelement.
- Er behält den Typ eines Nachrichtenelements.

Nun, in diesem Fall ist die Umwidmung eines Inhaltstyps ideal. Nehmen Sie in diesem Beispiel die Factory-basierte Typinformation eines Nachrichtenelements, laden Sie sie in das Werkzeug `portal_types`, und nennen Sie sie dann *Pressemitteilung*. Dadurch können Sie den gesamten vorhandenen Code und die Information wiederbenutzen, während Sie auch neue Möglichkeiten haben. Gehen Sie im ZMI zu `portal_types`, und führen Sie folgende Schritte aus:

1. Wählen Sie *Factory-based Type Information*.
2. Geben Sie als ID **Pressemitteilung** ein, und wählen Sie CMF DEFAULT: NEWS ITEM im Feld *Use default type information*.
3. Klicken Sie auf ADD, um dieses Formular zu beenden.

Das ist nun eine Instanz der Konfiguration eines Nachrichtenelements, heißt aber *Pressemitteilung*. Welchen Vorteil haben Sie davon? Nun, Sie haben jetzt einen weiteren Objekttyp, den Benutzer über das Web hinzufügen können. Damit haben die Benutzer Ihrer Site eine wirklich einfache Möglichkeit, zwischen einer Nachricht und einer Pressemitteilung zu unterscheiden, ohne sich mit Stichwörtern oder Metadaten herumschlagen zu müssen. Diese Objekte erscheinen nun auch bei Suchvorgängen und an allen anderen Stellen als Pressemitteilungen. Nun können Sie die Konfiguration der Pressemitteilung ändern, ohne dass es einen Einfluss auf die Konfiguration von Nachrichten hätte.

Änderungen am Icon wurden in diesem Kapitel bereits angesprochen: Laden Sie das Bild einfach in Ihr Verzeichnis, und ändern Sie dann für eine Pressemitteilung die Icon-Eigenschaft auf der Seite `portal_types`. Wenn Sie zum `portal_workflow` gehen, sehen Sie, dass alle Inhaltstypen einen eigenen Workflow haben. Da Sie nun einen neuen Inhaltstyp haben, können Sie gezielt den Workflow von Pressemitteilungen ändern. Vielleicht benötigen Pressemitteilungen eine zusätzliche Überprüfung, oder sie senden nach ihrer Veröffentlichung E-Mails an bestimmte Benutzer. Nun können Sie einen neuen Workflow erstellen, wie ich es in Kapitel 8 gezeigt habe, und ihn an Ihre Pressemitteilungen zuweisen.

Eine neue Anzeige hinzuzufügen bedeutet, das Page Template `newsitem_view` anzupassen und in etwas Sinnvolles umzubenennen, z.B. in `pressrelease_view`. Vielleicht möchten Sie diese Datei ändern, um am Seitenende einige Angaben über die Firma zu machen. Beispiel:

```
<h2>About ACME Widget Company</h2>
<p>Our company is the prime maker of widgets in the world. Founded
in 1980 we've been providing excellent widgets to all parts of the
```

```
globe. For more marketing information, please contact: Joe Bloggs,
marketing director.</p>
```

Nachdem Sie Ihre Änderungen an Ihrem neuen Page Template gespeichert haben, gehen Sie zu den Einstellungen der Pressemitteilung in `portal_types` zurück und gehen dort auf die ACTIONS-Seite. Ändern Sie die Aktion für die Anzeige einer Pressemitteilung von `newsitem_view` in `pressrelease_view`. Immer, wenn Sie nun eine Pressemitteilung anzeigen, wird diese Anzeigeseite angezeigt, wie auch in Abbildung 11.1 zu sehen ist.

Abbildung 11.1: Ein in Plone geladenes Beispiel-Python-Script

In diesem Fall habe ich ein Pressemitteilungsobjekt hinzugefügt, und die Fußzeile über ACME Company befindet sich im Template, d.h., die Benutzer müssen sie nicht immer eingeben.

11.1.6 Ein Scripting-Objekt erstellen

Sobald ein Objekt im Werkzeug `portal_types` registriert ist, können Sie dann Objektinstanzen davon in Ihrer Plone-Site erzeugen. Die Erzeugung solcher Objekte können Sie auch programmgesteuert mit Scripten durchführen. Das ist beim Erstellen von Objekten nützlich, die auf bestimmten anderen Faktoren basieren oder wenn Objekte en masse erzeugt werden. Plone hat zu diesem Zweck zwei nützliche Script (Python)-Objekte:

- **generateUniqueId**: Dies erzeugt eine neue eindeutige ID für diesen Objekttyp, z.B. `Folder.2003-12-19.7359814582`. Sie ist nur in dem Ordner eindeutig, in dem sie erzeugt wird. Wenn Sie schnell viele Objekte erzeugen, kann es sein, dass diese nicht eindeutig sind. Aber im Normalfall ist das gut genug.
- **invokeFactory**: Dies erwartet eine ID und einen Typnamen. Es erzeugt ein Objekt des angegebenen Typs und weist ihm die angegebene ID zu.

Nun werden Sie ein Beispielscript erstellen, das einen Ordner und eine Standardseite darin erzeugt, und in dieser Standardseite setzen Sie einen bestimmten Inhalt. Wenn Ihnen das bekannt vorkommt, dann vielleicht deswegen, weil das immer dann passiert, wenn Sie Mitglied auf einer Site werden und für Sie ein Home-Verzeichnis angelegt wird. Die Typnamen entsprechen der Registrierung im Werkzeug `portal_types`. Das heißt, wenn Sie einen Ordner erzeugen und ein Dokument darin platzieren möchten, dann müssen Sie die Parameter `Folder` und `Document` an das Script `invokeFactory` übergeben.

Listing 11.2 zeigt ein Script, das sich eine eindeutige ID holt und einen Ordner auf Basis dieser ID erstellt. Dann erstellt es in diesem Ordner ein neues Dokument.

Listing 11.2: Holen einer ID und Erstellen eines Ordners

```
##title=Create
##parameters=
# create with a random id
newId = context.generateUniqueId('Folder')

# create a object of type Folder
context.invokeFactory(id=newId, type_name='Folder')
newFolder = getattr(context, newId)

# create a new Document type
newFolder.invokeFactory(id='index.html', type_name='Document')

# get the new page
newPage = getattr(newFolder, 'index.html')
newPage.edit('html', '<p>This is the default page.</p>')

# return something back to the calling script
return "Done"
```

Wenn Sie das als Script(Python)-Objekt hinzufügen und es im TEST-Reiter testen, wird ein Ordner für Sie erzeugt. Interessanterweise werden der Ordner und das Dokument im aktuellen Kontext erzeugt, was auch immer das Kontextobjekt sein mag.

11.1.7 Das Inhaltstyp-Register

Ich habe Ihnen eine Vielzahl verschiedener Zugriffsarten auf Plone gezeigt, z.B. FTP und WebDAV. Wenn Plone Inhalte über einen dieser Zugangswege empfängt, muss es mit diesen Inhalten in der passenden Weise umgehen. Dafür sorgt das Inhaltstyp-Register, das Sie im ZMI unter dem Werkzeug `content_type_`

`registry` sehen können. Wenn Sie dieses Werkzeug in Zope aufsuchen, werden Sie wahrscheinlich von einem weiteren schlecht entworfenen Formular im ZMI geblendet, aber lassen Sie sich davon nicht entmutigen!

Wenn ein Inhalt über FTP oder WebDAV in Plone hinzugefügt wird, werden die Regeln im Register von oben nach unten ausgeführt, bis eine Übereinstimmung erfolgt. Diese basiert auf den Kriterien dieser Regel, und bei einer Übereinstimmung wird der entsprechende Inhaltstyp für diese Regel erzeugt. Folgendes sind die vier verschiedenen Arten von Kriterien:

major_minor: Nimmt die zwei Teile (links und rechts vom Schrägstrich) des MIME-Typs (Multipurpose Internet Mail Extensions) einer empfangenen Datei und führt darauf den Vergleich aus. Wenn Sie einen Teil leer lassen, passt dieser auf alles. Ein `major_minor` mit dem Wert `image` (beachten Sie das eine Leerzeichen rechts) passt z.B. auf `image/jpeg`, `image/gif`, `image/png` usw.

- **extension**: Passt zur Dateinamenserweiterung. Alle Erweiterungen werden mit einem Leerzeichen voneinander getrennt. Das heißt, `doc pdf` passt auf `rechnungen.doc` und `bericht.pdf`.
- **mimetype_regex**: Führt einen Vergleich mit einem regulären Ausdruck auf dem MIME-Typ durch. Das heißt, `*,^j` passt auf `image/jpeg`, `image/jpg`, `application/java` usw.
- **name_regex**: Führt einen Vergleich mit einem regulären Ausdruck auf dem Dateinamen aus. Zum Beispiel passt `^Rechnung` auf `Rechnung-123.pdf`, aber nicht auf `Keine_Rechnung-123.pdf`.

Um einen Typ hinzuzufügen, geben Sie den Namen der Regel und den Typ aus dem Dropdown-Menü im Formular am Seitenende ein und klicken auf ADD. Dadurch wird am Seitenende eine Regel erstellt. In dieser Regel können Sie ein Muster eingeben, das mit der von Ihnen erstellten Art von Regel übereinstimmt, und den gewünschten Inhaltstyp in der Dropdown-Liste wählen. Dann können Sie UP und DOWN anklicken, um Ihren Eintrag jeweils nach oben und unten zu verschieben, um dessen Priorität zu verändern.

Ein Beispiel: Neulich habe ich eine digitale Kamera gekauft, und da das Installationsprogramm von Plone unter Windows auch CMFPhoto und PIL einrichtet, dachte ich, ich könnte aus meinen Bildern mit geringem Aufwand ein Online-Fotoalbum erstellen. Zuerst habe ich den FTP-Server aktiviert und bin dann zur Inhaltstyp-Registry gegangen, wo ich eine neue Regel erstellt habe, die von der Erweiterung abhängt, die `image/jpeg` auf den Inhaltstyp `photo` abbildet. Dann habe ich diese Regel nach oben über die vorhandene Regel für Bilder verschoben. Anschließend musste ich nur noch meine Fotos in meinen FTP-Client ziehen, und schon wurden sie automatisch in Plone geladen, mit Miniaturbildchen versehen und angezeigt.

11.2 Inhalte suchen und kategorisieren

Wie Sie in Plone nach Inhalten suchen können, haben Sie bereits gesehen, aber nun werde ich ins Detail gehen und zeigen, wie die dahinter liegende Kategorisierung und Suche von Inhalten erfolgt. Das wesentliche Werkzeug, in dem all diese Informationen gespeichert sind, heißt `portal_catalog`, eine leicht andere und erweiterte Version des zugrunde liegenden Werkzeugs ZCatalog. Eine sehr gute Online-Referenz zum ZCatalog finden Sie unter *http://zope.org/Documentation/Books/ZopeBook/2_6Edition/SearchingZCatalog.stx*.

Der Katalog bietet für eine Plone-Site drei Schlüsselelemente: Er erstellt Indizes von Inhalten, enthält Metadaten über diese Inhalte im Index und bietet eine Suchschnittstelle zum schnellen Untersuchen von Inhalten in Ihrer Plone-Site. Von all den verschiedenen Objekten in Ihrer Zope-Site werden nur die eigentlichen Instanzen Ihrer Inhaltstypen katalogisiert. Zope-Objekte, -Werkzeuge und andere Objekte kommen nicht in den Katalog. Aus diesem Grund ist das Katalog-Werkzeug eng an die Inhaltstypen und ihre Benutzung gebunden. Auf den Katalog können Sie mit dem Werkzeug `portal_catalog` im ZMI zugreifen.

11.2.1 Inhalte indizieren

Als Erstes muss der Katalog Indizes vom Inhalt erstellen. Ein Index bietet vor allem eine Methode, um schnell und effizient im Inhalt zu suchen. Daher muss der Index-Inhalt für den Benutzer nicht klar sein oder Sinn machen. Er muss nur die schnelle und effiziente Suche ermöglichen. Wenn Sie in einer Plone-Site suchen, dann suchen Sie in den Indizes, und der Katalog gibt die zur Anfrage passende Ergebnisliste zurück.

Ein Index fragt ein Plone-Objekt nach einem bestimmten Wert ab, z.B. nach einer Methode oder einem Attribut, und indiziert dann, was immer dieses Objekt bei dieser Anfrage zurückgibt. Wie es den Inhalt tatsächlich indiziert, hängt von der Art des Indexes ab. Tabelle 11.1 führt alle in Plone verfügbaren Indizes auf.

Name	Beschreibung
`DateIndex`	Das dient zur Indizierung von Datumsangaben. Damit können Sie in Datums- und Zeitangaben suchen.
`DateIndexRange`	Eine effizientere Implementation von `DateIndex` für Fälle mit zwei Datumsangaben, z.B. Anfangs- und Enddatum, bei denen Sie viel darin suchen müssen.

Tabelle 11.1: Verfügbare Index-Typen

Name	Beschreibung
FieldIndex	Behandelt jedes Ergebnis automatisch und ermöglicht es Ihnen, nach allem zu suchen, was der Index enthalten mag. Er passt bei allen Suchen im Index.
KeywordIndex	Nimmt eine Folge von Stichwörtern und teilt sie in separate Wörter auf. Gibt ein Ergebnis zurück, falls eines der Stichwörter im Index auf die Anfrage passt. Ideal für die Suche nach Themen oder Stichwörtern von Objekten.
PathIndex	Indiziert den Pfad eines Objekts, z.B. /Members/jane/myDocument, als Liste von Objekten. Ermöglicht die Suche im Katalog nach allen Inhalten von Members, ohne den Ordner abfragen zu müssen. Ein Pfadindex gibt alles unter dem Members-Ordner zurück.
TextIndex	Ein alter Textindex, der Text nimmt, ihn aufteilt und dann indiziert. Siehe ZCTextIndex.
TopicIndex	Erstellt während der Katalogisierung vordefinierte Ergebnismengen. Nützlich bei oft wiederholten Abfragen.
ZCTextIndex	Ein neuer Index, der effiziente Volltext-Suchmöglichkeiten auf Textteilen bietet. Verfügt über viele verschiedene Eigenschaften, die später im Detail erörtert werden.

Tabelle 11.1: Verfügbare Index-Typen (Forts.)

Welche Indizes in einem Katalog definiert sind, können Sie sehen, wenn Sie auf PORTAL_CATALOG klicken und den INDEXES-Reiter wählen. Dann erhalten Sie eine Liste aller in Ihrer Plone-Site definierten Indizes. Die Spalten enthalten den Indexnamen, den Typ, die Anzahl der Treffer und den Zeitpunkt der letzten Änderung im Index. Die verschiedenen Indextypen wurden zuvor kurz behandelt, aber in Tabelle 11.2 finden Sie eine Beschreibung aller Standardindizes einer Plone-Site.

Name	Typ	Beschreibung
Creator	FieldIndex	Der Benutzername der Person, die das Objekt erzeugt hat.
Date	FieldIndex	Die Sperrfrist; falls nicht vorhanden, ist es das Datum der letzten Änderung.
Description	TextIndex	Das Beschreibungsfeld.
SearchableText	ZCTextIndex	Beschreibung, Titel und Rumpf des Objekts als ein suchbarer Text.
Subject	KeywordIndex	Die Stichwörter zu einem Element.
Title	TextIndex	Der Titel des Elements.

Tabelle 11.2: In Plone eingerichtete Standardindizes

Name	Typ	Beschreibung
Type	FieldIndex	Der Portal-Typ, wie er im Werkzeug portal_types definiert ist.
allowedRolesAndUsers	KeywordIndex	Gibt an, wer diesen Inhalt anzeigen kann; eine effiziente Art, das zu untersuchen, damit Sie die Suchergebnisse filtern können.
created	FieldIndex	Gibt an, wann das Element erzeugt wurde.
effective	FieldIndex	Gibt an, wann das Element freigegeben wird.
End	FieldIndex	Nur bei Ereignissen; gibt an, wann das Element beendet wird.
expires	FieldIndex	Gibt an, wann das Element abläuft und nicht mehr sichtbar ist.
GetId	FieldIndex	Die ID eines Elements.
Id	FieldIndex	Identisch mit getId.
in_reply_to	FieldIndex	Bei Diskussionen, ergibt das Element, dem dieser Kommentar gilt.
meta_type	FieldIndex	Der dem Element zugrunde liegende Metatyp.
modified	FieldIndex	Gibt an, wann das Element zuletzt verändert wurde.
Path	PathIndex	Der Pfad zum Element.
Portal_type	FieldIndex	Identisch mit Type.
Review_state	FieldIndex	Der Zustand des Objekts im Workflow.
Start	FieldIndex	Nur bei Ereignissen; gibt an, wann das Ereignis beginnt.

Tabelle 11.2: In Plone eingerichtete Standardindizes (Forts.)

Wenn Sie einmal nicht genau wissen sollten, was in einem Index enthalten ist, können Sie sich im ZMI dessen Inhalt anschauen. Klicken Sie auf PORTAL_CATALOG, und wählen Sie CATALOG, womit Sie eine Liste aller gerade katalogisierten Objekte erhalten. Klicken Sie auf ein Objekt, und es erscheint ein Fenster mit dem Inhalt des Indexes und der Metadaten. Die Metadaten kommen zuerst, d.h., Sie müssen nach unten scrollen, um die Indizes zu sehen.

Um Indizes zu ändern, zu löschen oder hinzuzufügen, gehen Sie zum INDEXES-Reiter zurück. Benutzen Sie das normale Dropdown-Menü namens ADD, um einen neuen Index hinzuzufügen oder einen zu löschen. Wenn Sie einen bestimmten Index neu erstellen möchten, wählen Sie links die gewünschten Indizes und klicken auf den Button REINDEX. Wenn Sie einen Index zum Katalog hin-

zufügen, hat er zunächst keinen Inhalt, d.h., Sie müssen dann auf den Button REINDEX klicken, um sicherzugehen, dass Ihr Index auch einen Inhalt hat.

Hinweis

Wenn Ihre Site sehr groß ist, kann diese Indizierung ziemlich viel Zeit und Prozessorleistung in Anspruch nehmen. Das heißt, Sie werden diesen Vorgang vermutlich nicht dann ausführen wollen, wenn das System gerade unter Spitzenlast fährt.

11.2.2 Metadaten

Die vom Katalog zurückgegebenen Ergebnisse sind nicht die gefundenen Objekte, sondern die dazu im Katalog gefundenen Metadaten. Diese Metadaten bestehen aus einer Reihe von Feldern oder Spalten für alle Werte im Objekt. Entsprechend wird für eine Plone-Site eine Liste von Spalten erzeugt, wie es in Tabelle 11.3 beschrieben ist.

Name	Beschreibung
`CreationDate`	Das Datum der Erzeugung des Objekts.
`Creator`	Der Benutzername der Person, die das Objekt erzeugt hat.
`Date`	Die Sperrfrist; falls nicht vorhanden, dann das Datum der letzten Änderung.
`Description`	Das Beschreibungsfeld.
`EffectiveDate`	Die Sperrfrist.
`ExpiresDate`	Das Ablaufdatum.
`ModificationDate`	Das Änderungsdatum.
`Subject`	Die Stichwörter des Objekts.
`Title`	Der Objekttitel.
`Type`	Der `portal_type` des Objekts.
`created`	Identisch mit `CreationDate`.
`effective`	Identisch mit `EffectiveDate`.
`End`	Nur bei Ereignissen, das Ende des Ereignisses.
`expires`	Gibt an, wann das Objekt abläuft.
`getIcon`	Das Icon des Objekts.
`GetId`	Die ID des Objekts.

Tabelle 11.3: In Plone verfügbare Standard-Metadaten

Name	Beschreibung
`getRemoteUrl`	Nur bei Links; die URL, auf die der Link zeigt.
`Id`	Identisch mit `getId`.
`location`	Nur bei Ereignissen; gibt an, wo das Ereignis eintritt.
`meta_type`	Der `meta_type` des Objekts.
`modified`	Gibt an, wann das Objekt verändert wurde.
`Portal_type`	Der `portal_type` des Objekts.
`Review_state`	Der Zustand des Objekts im Workflow.
`Start`	Nur bei Ereignissen; gibt an, wann das Ereignis eintritt.

Tabelle 11.3: In Plone verfügbare Standard-Metadaten (Forts.)

11.2.3 Wie Objekte indiziert werden

Inhaltstypen werden automatisch indiziert, weil sie von einer Klasse namens `PortalContent` erben, die ihrerseits von einer Klasse namens `CMFCatalogAware` erbt. Die Klasse `CMFCatalogAware` enthält den gesamten Code, der sicherstellt, dass beim Hinzufügen, Bearbeiten, Ausschneiden, Kopieren, Löschen oder Umbenennen eines Objekts der Katalog (und ebenso der Workflow) aktuell sind. Das Objekt wird im Wesentlichen an den Katalog übergeben, wo die passende Anweisung aufgerufen wird (indizieren, aus dem Index entfernen usw.).

Der Katalog geht dann alle Indizes durch und fragt alle aus dem Objekt mit Hilfe der Attribute oder Methoden des Objekts ab. Bei den meisten Indizes ist der Name des gesuchten Attributs bzw. der gesuchten Methode identisch mit dem des Indexes. Beim Indexnamen `Title` würde er nach einem Attribut oder einer Methode namens `Title` suchen und im Index das Ergebnis ablegen. Dann setzt er diesen Vorgang mit allen anderen Metadatenspalten fort.

Zwei Ausnahmen bei diesem Vorgang sind die Typen `FieldIndex` und `TopicIndex`. Wenn Sie einen `FieldIndex` hinzufügen, können Sie angeben, dass der Index einen anderen Wert als den Namen des Indexes untersucht. Sie könnten z.B. einen Index mit der ID `getVersion` anlegen, der Versionswerte untersucht. Wie Sie später noch sehen werden, haben manche Indizes Vorteile gegenüber anderen, d.h., es kann sinnvoll sein, verschiedene Indizes für den gleichen Wert zu haben.

`TopicIndex` ist insofern ein anderer Indextyp, als er bei der Erstellung des Inhaltsindexes eine Reihe von Mengen bildet. Wenn Sie viele Suchvorgänge auf allen Bildern durchführen möchten, könnten Sie eine Suche nach `o.portal_type == 'Image'` hinzufügen. Dazu müssen Sie einen `TopicIndex` erstellen und dann im INDEXES-Reiter auf den Index klicken. Sie können einen Index sogar mit mehreren

Ausdrücken erstellen. In Plone wird momentan nirgendwo ein `TopicIndex` verwendet.

Wie indizieren Sie alle Inhalte Ihrer Plone-Site neu?

Wenn Sie eine große Anzahl von Änderungen auf Codeebene vorgenommen, ein neues Produkt installiert, Ihr Plone-Wurzelobjekt umbenannt oder verschoben haben, müssen Sie eventuell den gesamten Inhalt auf Ihrer Site neu indizieren. Klicken Sie im ZMI hintereinander auf PORTAL_CATALOG, ADVANCED und UPDATE CATALOG. Dann wird Ihr Katalog aktualisiert.

Achtung

Diese Aufgabe ist noch aufwendiger als die Neuindizierung nur eines Indexes, und sie kann lange Zeit in Anspruch nehmen und viel Speicher und Rechenzeit benötigen, wenn Ihre Datenbank entsprechend groß ist.

Exkurs: Relationale Datenbanken vs. Plone

Die Entwicklung von Inhaltstypen in Plone unterscheidet sich leicht von der Entwicklung mit einer relationalen Datenbank. Heute ist LAMP, d.h. eine Kombination von Linux, Apache, MySQL und PHP oder Perl, ein weit verbreitetes Entwicklungsparadigma. Dabei werden Daten in einer Tabelle in der Datenbank gespeichert, und eine Scriptsprache bietet die Anwendungsebene, um den Inhalt aus der Datenbank zu holen und ihn in Templates zu setzen. Inhalte werden dabei gesucht, indem Sie Anfragen an die Datenbank in SQL abschicken, also `SELECT`-Anweisungen verwenden.

Plone macht das anders, da es eine Objektdatenbank verwendet. Alle Inhaltselemente können beliebige Attribute irgendeines Typs enthalten, und die zugrunde liegende Objektdatenbank kümmert sich um die persistente Speicherung dieser Attribute in der Datenbank. Um suchen zu können, werden dann alle Objekte im Werkzeug `portal_catalog` indiziert. Sie müssen dem Katalog explizit mitteilen, welche Attribute Sie genau indizieren lassen möchten. Anstatt SQL-Aufrufe zu verwenden, benutzen Sie dann den Katalog, um die Indizes zu untersuchen.

Im Entwicklungsstadium kann dieser Unterschied verwirrend sein, besonders deswegen, weil Beziehungen zwischen Objekten nicht so erzeugt und verwaltet werden, wie es in einer Anwendung der Fall wäre, die auf einer relationalen Datenbank basiert. Stattdessen gibt es zwei übliche Arten, Referenzen zu verwalten: die Verwendung eines Katalogs zur Verwaltung der Beziehung über Stichwörter oder andere Werte, oder die Verwendung eines Ordners zur Gruppierung von Inhalten. Mit Archetypes, die ich in Kapitel 13 behandeln werde, können Sie Beziehungen sehr leicht verwalten. Auch diese verwenden dabei den Katalog.

11.2.4 Suchen im Katalog

Die wichtigste Frage ist natürlich die, wie man im Katalog sucht und dessen Ergebnisse nutzt. Der erste Teil dieser Aufgabe hängt von den Indizes ab, daher behandle ich alle Indizes und zeige Ihnen, wie man darin sucht. Der zweite Teil beinhaltet die Manipulation der Ergebnisse, die ich Ihnen anschließend auch zeigen werde.

Alle folgenden Beispiele sind in Python geschrieben, weil dies die beste Art ist, in einem Katalog zu suchen. Ich zeige Ihnen auch schnell ein Beispiel dafür, wie man das in ein Page Template einbaut. Ich möchte Ihnen nahe legen, Python für die Manipulation des Katalogs einzusetzen, weil es wirklich der beste Platz mit der höchsten Flexibilität ist, um etwas zu machen, ohne sich um die Syntax sorgen zu müssen.

Ganz allgemein bewerkstelligen Sie eine Suche dadurch, dass Sie die Methode `searchResults` auf dem Objekt `portal_catalog` aufrufen und dabei eine Reihe von Schlüsselwortparametern übergeben. Zwar gibt es ein paar reservierte Schlüsselwörter, aber der Rest wird direkt auf die Indizes gleichen Namens abgebildet. Wenn Sie also z.B. im Index `SearchableText` suchen möchten, würden Sie der Suchmethode den Schlüsselwortparameter für `SearchableText` übergeben. Folgendes sind die reservierten Schlüsselwörter:

- **sort_on**: Dies ist der Index, mit dem die Ergebnisse sortiert werden sollen, vorausgesetzt, dass der Index eine Sortierung erlaubt (Volltext-Indizes tun das nicht).
- **sort_order**: Hat einen der Werte `reverse` oder `descending`. Wenn nichts angegeben ist, lautet die Voreinstellung `ascending`.
- **sort_limit**: Dies ist ein Optimierungshinweis, um die Sortierung etwas zu beschleunigen.

Eine allgemeine Suche nach allen Elementen, in denen Plone erwähnt wird, und die in der Reihenfolge von `Date` veröffentlicht sind, sieht ungefähr wie folgt aus:

```
context.portal_catalog.searchResults(
    review_state = "published",
    SearchableText = "Plone",
    sort_order = "Date"
)
```

Die Suche gibt die Schnittmenge der Indexergebnisse zurück, d.h., es werden alle Elemente gefunden, die Plone erwähnen *und* die veröffentlicht sind. Sie können keine Suche mit der Vereinigungsmenge von Ergebnissen durchführen, aber Sie könnten mehrere Ergebnisse bekommen und diese zusammensetzen, was allerdings ein eher unüblicher Fall ist.

Tipp

Wenn Sie eine Suche ohne Werte durchführen, wird der gesamte Inhalt des Katalogs zurückgegeben. Standardmäßig werden bei allen Suchvorgängen Werte für Sperrfrist und Ablaufdatum angegeben, um sicherzustellen, dass Sie nur Inhalte zwischen diesen Zeitpunkten sehen, es sei denn, der Benutzer, der die Suche ausführt, verfügt über das Recht *Access inactive portal content*.

Suche in einem Feld- oder Datumsindex

Bei der Suche in einem `FieldIndex` übergeben Sie den Wert des Feldes. Alle passenden Treffer werden zurückgegeben. Hiermit z.B. können Sie alle Bilder in einer Site suchen:

```
results = context.portal_catalog.searchResults(
    Type = "Image"
)
```

Ein Feldindex kann auch einen Bereich von Objekten annehmen, wobei dann der Index versucht, alle Werte dazwischen zu finden, indem er einen Vergleich der Werte vornimmt. Dieser Bereich könnte zwischen zwei Datumsangaben, zwei Zahlen oder zwei Strings liegen, es hängt wirklich nur vom Wert von `FieldIndex` ab. Dabei übergeben Sie ein Dictionary an den Index statt nur eines Strings. Das Dictionary sollte zwei Werte enthalten: eine Liste namens `query` mit den zu testenden Werten und einen Wertebereich. Dieser Bereich ist einer der folgenden Strings:

- **min**: Alles, was größer ist als das kleinste Element
- **max**: Alles, was kleiner ist als das größte Element
- **minmax**: Alles zwischen dem kleinsten und dem größten Element

Verwenden Sie z.B. Folgendes, um alle Ereignisse mit einer Startzeit größer als jetzt (mit anderen Worten alles, was in der Zukunft liegt) zu finden:

```
from DateTime import DateTime
now = DateTime()
results = context.portal_catalog.searchResults(
       Type = "Event"
       end = { "query": [now,],
                "range": "min" }
)
```

Um Elemente in einem Bereich zu suchen, z.B. alle Nachrichten im Dezember, müssen Sie die Start- und Endzeiten des Monats berechnen. Mit diesen Daten können Sie dann die folgende Abfrage erstellen:

```
from DateTime import DateTime
start = DateTime('2004/12/01')
end = DateTime('2004/12/31')
results = context.portal_catalog.searchResults(
        Type = "News Item",
        created = { "query": [start, end],
                        "range": "minmax" }
)
```

Datumsindizes funktionieren auf die gleiche Weise wie Feldindizes, und oftmals werden Sie Datumsangaben innerhalb von Feldindizes sehen, was prima funktioniert.

Suche in einem KeywordIndex

Ein `KeywordIndex` gibt standardmäßig alle Werte zurück, die im Stichwortindex gefunden werden. Der einzige `KeywordIndex` ist `Subject`. Das ist das Stichwort, das ein Benutzer einem Objekt mit Hilfe des EIGENSCHAFTEN-Reiters in der Plone-Schnittstelle zugewiesen hat. Benutzen Sie Folgendes, um nach allen Elementen mit dem Stichwort `Afrika` zu suchen:

```
results = context.portal_catalog.searchResults(
        Subject = "Afrika"
)
```

Ähnlich wie beim `FieldIndex` kann auch an `KeywordIndex` eine kompliziertere Abfrage mit mehreren Objekten und einem `and`/`or`-Operator übergeben werden (`or` ist die Vorgabe). Damit können Sie alle Objekte mit einer fast beliebigen Kombination von Stichwörtern finden. Benutzen Sie z.B. Folgendes, um alle Objekte zu finden, die die Stichwörter `Afrika` und `Sonne` enthalten:

```
results = context.portal_catalog.searchResults(
        Subject = { "query": ["Afrika", "Sonne"],
                    "operator": "and" }
)
```

Suchen in einem Pfadindex

Mit einem Pfadindex können Sie in allen Objekten eines bestimmten Pfades suchen. Es werden alle Objekte unterhalb eines bestimmten Ortes angegeben. Wenn Sie also nach allen Objekten in `Members` fragen, erhalten Sie alles, was in den Home-Verzeichnissen aller Mitglieder enthalten ist. Eine solche Suche nach allem, was `Members` im Pfad hat, sieht wie folgt aus:

```
results = context.portal_catalog.searchResults(
        path = "/Plone/Members"
)
```

Wenn Sie das weiter einschränken möchten, können Sie das tun, indem Sie einen Ebenenparameter übergeben, der angibt, wo Sie das Element erwarten. Die Ebene ist eine Zahl, die für seine Position im Pfad steht, und zwar von links aus gesehen, wenn man ihn an den Schrägstrichen aufteilt. Im vorherigen Code z.B. liegt `Plone` auf der Ebene 0, `Members` auf der Ebene 1 usw. Ähnlich wie beim `KeywordIndex` können Sie auch hier einen `and`/`or`-Operator übergeben. Um z.B. alle Objekte in den Ordnern `/Plone/Members/danae` und `/Plone/testing/danae` zu erhalten, benutzen Sie Folgendes:

```
results = context.portal_catalog.searchResults(
        path = { "query": ["danae"],
                 "level" : 2 }
)
```

Suchen im ZCText-Index

`ZCTextIndex` ist der komplizierteste aller Indizes und verfügt über eine große Zahl von Optionen. Jeder `ZCTextIndex` benötigt ein Lexikon, aber Plone erzeugt und konfiguriert glücklicherweise all das von sich aus. Wenn Sie auf PORTAL_CATALOG klicken, den CONTENTS-Reiter wählen und dann auf PLONE_LEXICON klicken, können Sie die Standardkonfiguration des Lexikons sehen. Mit einem Klick auf den QUERY-Reiter sehen Sie alle Wörter, die im Lexikon enthalten sind, das aus den Inhalten Ihrer Plone-Site erstellt wurde.

Der `ZCTextIndex` wird mit dem Format durchsucht, das ich in Kapitel 3 beschrieben habe. Es arbeitet mit ähnlichen Suchbegriffen, wie Sie sie auch bei Google oder anderen Suchmaschinen benutzen können. In der einfachsten Form können Sie wie folgt nach einem beliebigen Begriff suchen (beachten Sie, dass hierbei die Schreibweise irrelevant ist):

```
results = context.portal_catalog.searchResults(
        SearchableText = "space"
)
```

Aber Sie können auch alles Folgende suchen:

- **Globbing**: Verwenden Sie einen Stern für beliebig viele Buchstaben. So passt z.B. `Dien*` auf `Dienstag` und `Dienstage`. Am Wortanfang können Sie allerdings keinen Stern verwenden.
- **Einzelne Fragezeichen**: Verwenden Sie ein Fragezeichen für einen Buchstaben. Beispiel: `Ro?e` passt auf `Rose`, `Robe`, `Rote` usw. Am Wortanfang können Sie jedoch kein Fragezeichen verwenden.
- **And**: Mit `and` geben Sie an, dass die beiden Begriffe rechts und links davon vorhanden sein müssen. Beispiel: `Rom and Dienstag` gibt nur Ergebnisse zurück, in denen beide Wörter im Inhalt vorkommen.
- **Or**: Mit `or` geben Sie an, dass mindestens einer der Begriffe vorkommen muss. Beispiel: `Rom or Dienstag` gibt Ergebnisse zurück, falls eines der Wörter im Inhalt vorkommt.
- **Not**: Mit `not` erhalten Sie Ergebnisse, in denen der Begriff nicht vorkommt (es wird ein `and` als Präfix benötigt). Beispiel: `Willkommen and not Seite` würde passende Seiten zurückgeben, in denen `Willkommen`, aber nicht `Seite` vorkommt.
- **Phrases**: Sätze können Sie in doppelten Anführungszeichen (") setzen, um damit mehrere Wörter nacheinander anzugeben. Beispiel: `"Diese Seite"` passt auf `Diese Seite führt Sie ins Plone Content-Management-System ein.`, aber nicht auf `Diese Startseite von...`.
- **Not phrase**: Sie können einem Satz ein Minuszeichen (`-`) als Präfix voranstellen. Beispiel: `Start -"Diese Seite"` passt auf alle Seiten, in denen `Start` vorkommt, aber nicht `Diese Seite`.

Tipp

Wenn Sie eine Suche ohne Text durchführen, werden keine Ergebnisse zurückgegeben.

Die Ergebnisse benutzen

Nun haben Sie also einige Ergebnisse, aber was fangen Sie damit an? Die meisten Leute sehen sich als Erstes die Ergebnisse an und nehmen dabei an, dass es sich

um eine Liste der katalogisierten Objekte handelt. Das ist aber nicht der Fall, sondern es handelt sich um eine Folge von »Kataloghirnen« (engl. *catalog brains*). Diese Hirne sind tatsächlich »Lazy«-Objekte, die die zuvor definierten Metadatenspalten enthalten. Auf all diese Spalten können Sie so zugreifen, als ob es sich um Attribute handeln würde. Verwenden Sie Folgendes, um z.B. alle IDs der Ergebnisobjekte auszugeben:

```
results = context.portal_catalog.searchResults()
for result in results:
    print result.getId
return printed
```

In diesem Beispiel ist `getId` der Name einer Metadatenspalte, d.h., sie wird den Wert von `getId` anzeigen, der im Katalog für dieses Objekt enthalten ist. Wenn Sie versuchen, auf einen Wert zuzugreifen, der nicht als Metadatenspalte existiert, erhalten Sie einen `AttributeError`. Folgendes sind ein paar nützliche Methoden solcher Hirne:

- **getPath**: Gibt den physischen Pfad dieses Objekts in Zope zurück.
- **getURL**: Gibt die URL für dieses Objekt zurück, bei der Virtual Hosting angewendet wurde.
- **getObject**: Gibt das eigentliche Objekt zurück.
- **getRID**: Eine eindeutige ID für das Objekt im Katalog, die sich jedes Mal ändert, wenn das Objekt nicht mehr katalogisiert ist. Ist nur für interne Aufgaben gedacht.

Wenn Sie also für jedes Ergebnis das Objekt haben möchten, können Sie das haben, wie Sie im folgenden Beispiel gleich sehen werden. Allerdings gibt es einen Grund dafür, warum der Katalog das nicht selbst macht: Diese Operation ist aufwendig in puncto Rechenzeit, weil damit ein Weckvorgang des Objekts (und aller Objekte dazwischen) verbunden ist, bei dem es aus der Datenbank geholt wird, und außerdem werden viele Sicherheitsprüfungen vorgenommen. Wenn Sie es einrichten können, dass Ihre Metadaten die passende Information enthalten, werden Sie eine wesentlich schnellere Anwendung haben. Natürlich können Metadaten manchmal nicht alles enthalten, aber eine Überlegung ist es beim Entwurf wert. Um an alle Objekte zu kommen, können Sie Folgendes benutzen:

```
results = context.portal_catalog.searchResults()
for result in results:
    object = result.getObject()
    print object
return printed
```

Da Sie über eine Python-Liste dieser Hirne verfügen, ist es nun sehr leicht, die Ergebnisse in der geeigneten Weise zu manipulieren. Um herauszufinden, wie viele Ergebnisse gefunden wurden, können Sie einfach wie folgt die Funktion *len* auf der Liste aufrufen:

```
results = context.portal_catalog.searchResults()
print "Anzahl der Ergebnisse", len(results)
return printed
```

Hinweis

`len` ist eine Python-Funktion, die Ihnen die Länge eines Objekts mitteilt.

Um nur die ersten zehn Elemente zu bekommen, verwenden Sie einen Python-Teilbereich wie folgt:

```
results = context.portal_catalog.searchResults()
return results[:10]
```

Um eine weitere Filterung vorzunehmen, können Sie die gesamte Liste wie folgt manuell filtern:

```
results = context.portal_catalog.searchResults()
for result in results[:10]:
    # Title returns a string so we can use the find method of
    # a string to look for occurence of a word
    if result.Title.find("Plone") > -1:
        print result.Title
return printed
```

Um ein zufällig gewähltes Objekt aus dem Katalog zu erhalten, können Sie das Modul random wie folgt benutzen:

```
import random
results = context.portal_catalog.searchResults()
r = random.choice(results)
object = r.getObject()
return object
```

11.2.5 Alles zusammen: Erstellen eines Suchformulars

In den vorangegangenen Betrachtungen habe ich Ihnen gezeigt, wie Sie einige Ergebnisse aus dem Katalog bekommen, und ich habe Script (Python)-Objekte benutzt, um das zu demonstrieren. Aber vermutlich fragen Sie sich, wie Sie das aus einem Page Template tun können?

Ich fange am anderen Ende an und nehme zuerst an, dass Sie die Ergebnisse aus einer Kataloganfrage schon haben und in einem Page Template mit `tal:repeat` darüber iterieren. Auf diese Weise funktionieren eine Menge Portlets, auch die für zuletzt veröffentlichte Elemente und Ereignisse, die beide lediglich den Katalog abfragen und dann die Ergebnisse anzeigen. Diese Portlets betten die Abfrage in einem Page Template ein, entweder, indem sie dieses direkt aufrufen:

```
<div tal:define="results python:
here.portal_catalog.searchResults(Type="Event")">
```

oder indem sie ein separates Script (Python)-Objekt aufrufen, das die Ergebnisse zurückgibt. Im folgenden Beispiel heißt das Script `getCatalogResults`:

```
##parameters=
kw = {}
# enter your query into the kw dictionary
return context.portal_catalog(**kw)
```

In einem Page Template würden Sie die Ergebnisse auf folgende Weise erhalten:

```
<div tal:define="results here/getCatalogResults">
```

Danach müssen Sie mit der normalen `tal:repeat`-Syntax über die Ergebnisse iterieren. Auf alle Metadatenspalten können Sie direkt in TAL (Template Attribute Language) mit einem Pfadausdruck auf die Spalte zugreifen. Mit einem gegebenen Hirn könnten Sie mit einem Aufruf von `result/Title` den Titel aus den Metadaten ermitteln. Listing 11.3 zeigt eine Beispielseite, die über den Inhalt von `getCatalogResults` iteriert und jedes Element in einer einfachen ungeordneten Liste anzeigt.

Listing 11.3: Iterieren über getCatalogResults

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-US"
      lang="en-US"
      metal:use-macro="here/main_template/macros/master"
      i18n:domain="plone">
<body>
<div metal:fill-slot="main">
<ul tal:define="results here/getCatalogResults">
```

```
    <li tal:repeat="result results">
        <a href=""
           tal:attributes="href result/getURL"
           tal:content="result/Title" />
        <span tal:replace="result/Description" />
    </li>
</ul>
</div>
</body>
</html>
```

Ein Merkmal der Methode `searchResults` ist, dass sie bei einem Aufruf ohne Parameter in der ankommenden Abfrage nach diesen sucht. Wenn Sie also möchten, dass ein Formular Parameter zu Ihren Ergebnissen hinzufügt, müssen Sie nur die vorherige Ergebniszeile wie folgt ändern:

```
<ul tal:define="

  results python: here.portal_catalog.searchResults(REQUEST=request)
  ">
```

Nun können Sie Ihre Abfrage erneut ausführen und einen beliebigen Index zur URL hinzufügen. Wenn Sie dieses Page Template z.B. `testResults` nennen und in Ihrem Browser ans Ende der URL `?Type=Document` hinzufügen, würden nur die Dokumente in Ihrer Site erscheinen. Da Sie fast alle Abfragewerte übergeben können, können Sie ein Suchformular so einrichten, dass diese Information zum Suchformular durchgereicht wird. Genau das machen die Seiten zur Suche und zur erweiterten Suche. Wenn Sie auf eine Plone-Site gehen und in dem Suchkasten nach `Wein` suchen, werden Sie bemerken, dass Ihre URL nun `?SearchableText=Wein` enthält.

Listing 11.4 zeigt ein Formular zum Aufrufen Ihrer Page Templates.

Listing 11.4: Ein Formular zum Aufrufen Ihres Templates

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-US"
      lang="en-US"
      metal:use-macro="here/main_template/macros/master"
      i18n:domain="plone">
<body>
<div metal:fill-slot="main">
  <p>Select a content type to search for</p>
  <form method="post" action="testResults">
    <select name="Type">
      <option
       tal:repeat="value python:here.portal_catalog.uniqueValuesFor('Type')"
```

```
        tal:content="value" />
         </select>
         <br />
         <input type="submit" class="context">
    </form>
</div>
</body>
</html>
```

Dieses Script benutzt eine Methode namens `uniqueValuesFor` im Katalog, die alle eindeutigen Werte zurückgibt, die in einem Index existieren. Damit können Sie eine Aufgabe wie das Ausfüllen eines kleinen Dropdown-Menüs in einem Formular erledigen, was sehr nützlich ist.

An diesem Punkt wird es zu einer Übung in HTML und Page Templates, Ihre Seiten so komplex zu machen, wie Sie es wünschen. Der beste Platz, um all das nachzuschauen, ist natürlich in den echten Templates von Plone, die viele Zeilen mit tollen Beispielen enthalten. Alle Portlets, die Sie in Plone kennen (z.B. der Kalender, die Ereignisse, Dazu passend usw.) sind so aufgebaut, dass Sie Katalogabfragen benutzen, um zu bestimmen, was angezeigt werden soll.

In diesem Kapitel habe ich Ihnen einen Überblick über Methoden gezeigt, um eine Plone-Site zu entwickeln, und ich habe gezeigt, wie Inhaltstypen in Ihrer Site funktionieren. Außerdem habe ich vorgeführt, wie ein Inhaltstyp gebaut und dann über den Katalog referenziert wird. In Plone ist das eine sehr wichtige Entwicklungsstrategie.

Im nächsten Kapitel werde ich zeigen, wie man einen neuen Inhaltstyp mehr oder weniger bei null beginnend erstellt. Und Sie werden sehen, wie Sie diesen neuen Inhaltstyp mit dem Katalogregister im Werkzeug `portal_types` integrieren können.

12 Ein Produkt in Python schreiben

Wenn Sie ein Produkt für Plone schreiben, dürfen Sie fast alles tun, was Sie jemals mit Plone machen wollten. Inhaltstypen oder Werkzeuge in Python zu schreiben ist der beste Weg, höchste Flexibilität zu erreichen. Falls Sie dringend etwas Bestimmtes in Plone benötigen, was noch nicht anderswo vorhanden ist, dann haben Sie die Gelegenheit, dieses Feature hinzuzufügen, indem Sie ein Produkt schreiben. Das könnte z.B. die Speicherung einer firmenspezifischen Art von Inhalt oder irgendeine Manipulation sein, die nur Sie benötigen. Im vorigen Kapitel habe ich gezeigt, wie Sie einen Inhaltstyp anpassen können. Diese Anpassung bringt Sie jedoch nur bis zu einem gewissen Punkt. Sie können z.B. keine neuen Attribute zu Ihrem Inhaltstyp hinzufügen. Daher werden Sie lieber Ihren eigenen Inhaltstyp schreiben wollen.

In diesem Kapitel werde ich zwei Beispiele beschreiben: einen Inhaltstyp und ein Werkzeug. Beide Beispiele sind relativ einfach gehalten, bereiten Sie aber auf das nächste Kapitel vor, in dem ich Ihnen zeigen werde, wie Sie *Archetypes* benutzen, ein Framework für Plone, mit dem Sie Inhaltstypen schnell und einfach mit nur einer minimalen Menge von Quellcode erzeugen können.

Insbesondere erzeuge ich einen eigenen Inhaltstyp in Plone und gehe durch den gesamten Code durch, der diesen Inhaltstyp erzeugt. Es handelt sich um einen recht interessanten Inhaltstyp, denn er benutzt mehrere Bausteine, die er aus einem C-Modul von dritter Seite herauszieht, und baut sie in Ihre Plone-Site ein. Ich zeige Ihnen dabei, wie Sie zuerst den Inhaltstyp erstellen und dann Rechte, eine Suchintegration, neue Elemente für die Benutzerschnittstelle sowie Installationscripts hinzufügen. Und schließlich behandle ich, wie man ein Plone-Werkzeug erstellen kann, d.h. eine Art, neue Werkzeuge zu einer Site hinzuzufügen. Beide Beispiele in diesem Kapitel können Sie online herunterladen, installieren und studieren. Außerdem finden Sie in Anhang B den gesamten Code.

12.1 Einen eigenen Inhaltstyp schreiben

Auf der Website zum Plone-Buch (*http://plone-book.agmweb.ca*) wollte ich den Code dieses Buchs online anzeigen. Dazu hätte ich den Code nehmen und ihn einfach in Dokumente setzen können, aber dann würde er ohne Syntax-Hervorhebung angezeigt werden. Außerdem würden die Leerzeichen für die Einrückung in Python verschwinden. Für ein so tolles Produkt wie Plone wollte ich etwas, was gut aussieht. Also brauchte ich einen Inhaltstyp, der die Syntax im Code hübsch hervorhebt und es den Benutzern ermöglicht, ihn online anzuschauen. Abbildung 12.1 zeigt das fertige Beispielprodukt.

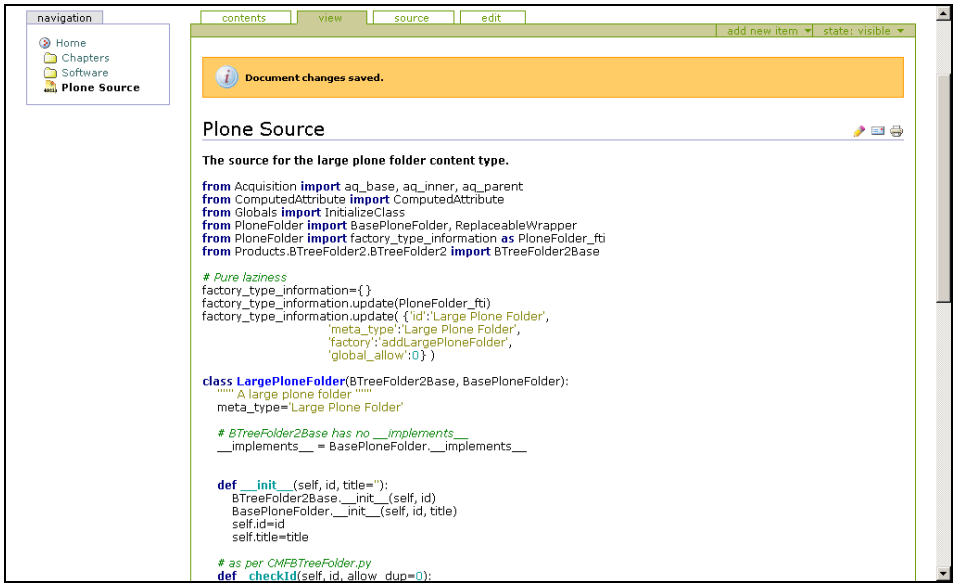


Abbildung 12.1: Ein in Plone geladenes Python-Beispielscript

Aus diesem Design können Sie einige Anforderungen für dieses Produkt ableiten. Insbesondere wird dieses Produkt die folgenden Attribute haben:

- **ID**: Jedes Stück Code hat eine eindeutige ID. Dieses Attribut ist notwendig.
- **Title**: Jedes Stück Code sollte einen Titel haben. Dieses Attribut ist notwendig.
- **Description**: Jedes Stück Code sollte eine Beschreibung dessen haben, was es tun sollte. Dieses Attribut ist optional.
- **Source code**: Jedes Stück Code wird ein Quellcode-Attribut haben, das die Quelle zu diesem Inhaltstyp enthält. Das wird optional sein, aber es ist vernünftig, es notwendig zu machen.

- **Language**: Dies ist die Sprache für den Quellcode, z.B. Perl, Python, HTML usw.

Der Inhaltstyp sollte natürlich mit Plone interagieren, damit Sie dessen Möglichkeiten nutzen können. Sie müssen sicherstellen, dass im Produkt gesucht werden kann, dass es mit den Sicherheitsmechanismen und dem Workflow zusammenarbeitet und korrekt persistent gespeichert wird. Außerdem wäre es nett, wenn die Benutzer Scripten direkt von ihrer Festplatte hochladen könnten, anstatt Code in irgendwelche Textbereiche einfügen zu müssen.

Beim Untersuchen dieses Codes musste ich einen einfachen Weg finden, Code in HTML umzuwandeln. Bei einer Sprache mit einer einfachen Syntax wie Python ist das ziemlich einfach (Python kann sogar seinen eigenen Code »lexen«), aber am liebsten hätte man das natürlich für mehrere Sprachen, z.B. HTML (Page Templates), JavaScript, Cascading Style Sheets (CSS) usw. Das *SilverCity*-Modul macht bereits genau das. Es ist auf SourceForge verfügbar (*http://silvercity.sf.net/*). Es verwendet C-Bibliotheken aus dem Scintilla-Texteditor, um den Code zu lexen. Ohne besonders auf die Implementierung einzugehen: Der Vorteil besteht darin, dass es zu fast einem Dutzend Programmiersprachen fröhlich HTML mit Syntax-Hervorhebung ausspuckt.

Bei einem Blick auf die Anforderungen werden Sie sehen, dass diese ziemlich leicht nachvollziehbar sind. Tatsächlich werden die ID, der Titel und die Beschreibung alle in der Dublin Core-Implementierung von Plone definiert. Sie brauchen sich also nur um den Quellcode und die Sprache zu kümmern. Plone verlangt eine ID und einen Titel, und eine Beschreibung ist wirklich sehr hilfreich.

12.1.1 Mit dem Inhaltstyp anfangen

Da Sie nun eine Vorstellung vom Inhaltstyp haben, den Sie in diesem Kapitel erstellen werden, können Sie anfangen, ihn zu erstellen, indem Sie Python-Code im Dateisystem schreiben. Dieser Inhaltstyp ist also auch ein Produkt, d.h., Sie erstellen ein neues Verzeichnis in Ihrem Produktverzeichnis. Der Name des zu erzeugenden Verzeichnisses ist der Name des Produkts, das Zope importieren wird. Treffen Sie also eine kluge Wahl für Ihren Namen. Ich habe mir spaßeshalber überlegt, das Produkt *SourceCode* oder *PloneSourceCode* zu nennen, fand diese Namen aber zu verwirrend, (man könnte denken, dass das Produkt der eigentliche Quellcode von Plone wäre). Aber *PloneSilverCity* schien ein netter Produktname zu sein, der auf seinen Ursprung hinweist und hinreichend obskur ist, dass ihn keiner mit etwas anderem verwechseln könnte.

Nachdem das Verzeichnis erstellt ist, füge ich normalerweise ein paar Dateien und Verzeichnisse hinzu, die ich benötige. Jedes Paket benötigt eine Datei

namens `__init__.py` darin. Dieser Dateiname wird von Python so verlangt und bedeutet, dass dieses Verzeichnis ein Python-Paket ist, d.h., dass es importiert werden kann. Wenn das Paket importiert wird, führt Zope diese Datei aus. In dieser Datei fügen Sie den Code zur Registrierung des Produkts in Zope hinzu.

Da Sie benutzerfreundlich sein möchten, können Sie auch ein paar Textdateien wie `readme.txt`, `install.txt` usw. hinzufügen. Eine andere nützliche Textdatei ist `refresh.txt`. Damit können Sie sich in das Refresh-Modul von Zope einklinken und das Produkt dynamisch neu laden, während Sie es schreiben. Das ist gerade bei Ihren ersten Schritten unglaublich hilfreich, wenn Sie eine Klasse schreiben, und ich werde ihnen später zeigen, wie Sie das in Zope konfigurieren.

Im Moment haben Sie ein Verzeichnis namens `PloneSilverCity` im Produktverzeichnis, das folgende leere Dateien enthält: `readme.txt`, `refresh.txt`, `install.txt` und `__init__.py`. Das stellt nun ein gültiges Python-Paket dar, das absolut nichts tut (aber nicht mehr lange).

Exkurs: Mit ZClasses entwickeln

Den Inhaltstyp erzeugen Sie mit Python, aber wahrscheinlich haben Sie schon von ZClasses gehört, sei es in andere Dokumentation oder im Internet. ZClasses sind ein Framework für Zope 2 und dienen zur Entwicklung von Klassen über das Web. Viele Leute haben ZClasses schon erfolgreich entwickelt und vertrieben, und für eine schnelle Entwicklung mögen sie eine Rolle spielen. Allerdings möchte ich sie wirklich nicht empfehlen. Es ist schwer, sie mit vorhandenen Werkzeugen zu entwickeln, sie in Quellcode zu setzen, zu vertreiben usw. Fast jeder, mit dem ich über ZClasses gesprochen habe, stimmt mir zu, dass sich der Aufwand lohnt, mit Python zu entwickeln, und ich habe schon mehrere Präsentationen gesehen, in denen ZClasses zu den Fehlerursachen gezählt wurden.

Wenn Sie Dokumentation oder andere Informationen bzgl. ZClasses sehen, so möchte ich Ihnen wirklich empfehlen, der Versuchung zu widerstehen, sie zu benutzen. Aus diesem Grund wird hier die Entwicklung mit ZClasses nicht weiter behandelt. Wenn Sie eine Methode suchen, schnell zu entwickeln, sollten Sie sich Archetypes anschauen, die einen leicht anderen Ansatz verfolgen.

12.1.2 Integration von SilverCity

Bevor Sie sich zu tief in den Zope-Code verstricken, könnte es sinnvoll sein herauszufinden, wie man SilverCity benutzt. Bei jeder Softwareentwicklung ist ein schichtweises Vorgehen, das das Testen einzelner Schichten erlaubt, von

größter Bedeutung. Aus diesem Grund sollten Sie damit anfangen, dass Sie sicherstellen, SilverCity direkt aus einem Python-Modul heraus benutzen zu können. Wenn das funktioniert, müssen Sie nur noch die Zope-Schicht hinzufügen.

Schauen Sie sich SilverCity also einmal an. Zuerst müssen Sie es installieren. Glücklicherweise folgt dieses Modul den in Kapitel 10 skizzierten Installationsanweisungen für Python-Module. Für eine Installation unter Windows laden Sie die Datei `SilverCity-0.9.5.win32-py2.3.exe` von *http://silvercity.sf.net* herunter und starten das grafische Installationsprogramm. Für Linux laden Sie die Datei `SilverCity-0.9.5.tar.gz` von der gleichen Website herunter und speichern sie auf der Platte. Dann packen Sie sie aus und starten das Programm `setup.py`. Beispiel:

```
$ tar -zxf SilverCity-0.9.2.tgz
$ cd SilverCity-0.9.2
$ python setup.py install
...
```

Danach können Sie unter dem Python-Prompt in Windows oder Linux schnell sehen, ob es funktioniert:

```
$ python
Python 2.3.2 (#1, Oct  6 2003, 10:07:16)
[GCC 3.2.2 20030222 (Red Hat Linux 3.2.2-5)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import SilverCity
>>>
```

Das bedeutet, dass SilverCity erfolgreich installiert worden ist. Falls Sie kein ähnliches Ergebnis erhalten und SilverCity nicht importieren können, unterbrechen Sie an dieser Stelle und lösen zuerst das Problem, denn sonst funktioniert überhaupt nichts.

Nun müssen Sie die API (Application Programming Interface) dieses Moduls herausfinden. Weil ich faul bin, habe ich mir ein Beispielscript in `PySilverCity/Scripts` namens `source2html.py` angeschaut. Dieses Script tut genau, was Sie wollen: Es spuckt HTML zu einem gegebenen Stück Code aus. Eine wirklich freche Art, es in Aktion zu sehen, besteht darin, dieses Script wie folgt auf sich selbst loszulassen:

```
$python source2html.py source2html.py --generator=python

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

```
  <title>source2html.py</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <link
    rel="stylesheet"
    href="default.css" />
</head>
...
```

Das heißt, Sie müssen sich nur diese API anschauen und sie leicht abändern. Fügen Sie ein Modul namens `source.py` im Verzeichnis `PloneSilverCity` hinzu. Dort schreiben Sie den Code, der die Schnittstelle zur Bibliothek enthält. Dieses neue Modul enthält momentan keinen Zope- oder Plone-spezifischen Code. Es hat drei Hauptbestandteile: Es nennt Ihnen alle möglichen Sprachen, die Sie benutzen können, es nimmt einen Text an, und es gibt den korrekten Parser zurück, und schließlich führt es die eigentliche Umwandlung aus.

Fügen Sie zuerst die folgende Funktion `create_generator` hinzu, mit der Sie den korrekten Parser erhalten:

```
from SilverCity import LanguageInfo
from StringIO import StringIO

def create_generator(source_file_name=None, generator_name=None):
    """ Make a generator from the given information
    about the object, such as its source and type """
    if generator_name:
        return LanguageInfo.find_generator_by_name(generator_name)()
    else:
        if source_file_name:
            h = LanguageInfo.guess_language_for_file(source_file_name)
            return h.get_default_html_generator()()
        else:
            raise ValueError, "Unknown file type, cannot create lexer"
```

Zweitens, wenn Sie in Plone sind, müssen Sie genau herausfinden, welche Sprachen verfügbar sind, damit Sie diese den Benutzern anzeigen können. Schreiben Sie die folgende Funktion namens `list_generators`, die diese Liste zurückgibt:

```
def list_generators():
    """ This returns a list of generators, a generator
    is a valid language, so these are things like perl,
    python, xml etc..."""
    lexers = LanguageInfo.get_generator_names()
    return lexers
```

Die Funktion `generate_html` nimmt schließlich eine Quelldatei als String an, einen optionalen Generator und einen optionalen Dateinamen. SilverCity benötigt eine Datei, z.B. `buffer`, um den Inhalt irgendwohin zu schreiben, d.h., Sie können das Python-Modul `StringIO` zu diesem Zweck verwenden. Die Funktion `generate_html` lautet wie folgt:

```
def generate_html(source_file, generator=None, source_file_name=None):
    """ From the source make a generator
    and then make the html """

    # SilverCity requires a file like object
    target_file = StringIO()
    generator = create_generator(source_file_name, generator)
    generator.generate_html(target_file, source_file)

    # return the html back
    return target_file.getvalue(), generator.name
```

Sie werden bemerken, dass sie die Funktion `create_generator` aufruft, die Sie zuvor geschrieben haben, um den richtigen Generator für diese Sprache zu finden. Das ist der ganze Code, den Sie benötigen, um zu einer Datei das entsprechende HTML zu generieren. Ich bin nicht auf irgendwelche Besonderheiten des eigentlichen Lexens des Quellcodes oder der Erzeugung des HTML-Codes eingegangen. Die SilverCity-Bibliothek erledigt das alles für Sie. Um den vorigen Punkt zu wiederholen: In diesem Modul haben Sie keine Referenz auf Zope oder Plone, d.h., dieses Modul ist völlig selbstständig. Die eigentlichen Details dieses Moduls müssen Sie nicht kennen, solange Sie wissen, dass Sie eine Bibliothek von dritter Seite importieren.

In Pyhon-Scripten gibt es die Tradition, mindestens einen Test in den Code einzubauen. Sie könnten auch eine vollständige Unittest-Suite erstellen, aber das geht über das aktuelle Thema hinaus. Stattdessen werden Sie ein wenig Code hinzufügen, um zwei Dinge zu testen, nämlich zum einen, dass es funktioniert, und zum anderen, dass die Sprachen verfügbar sind:

```
if __name__ == "__main__":
    import sys
    myself = sys.argv[0]
    file = open(myself, 'r').read()
    print generate_html(file, generator="python")
    print list_generators()
```

Wenn Sie dieses Script ausführen, öffnet es sich selbst und übergibt seinen Code an den Teil, der die HTML-Syntax hervorhebt. Dann wird eine Menge HTML-Code ausgespuckt. Das könnten Sie einfach ins Zope-spezifische Modul setzen,

das Sie gleich schreiben werden. Wenn sich allerdings alles in getrennten Scripten befindet, wird es später einfacher zu testen und zu ändern.

12.1.3 Die Klasse schreiben

Ein Inhaltstyp in Plone ist lediglich ein Objekt, das einige bestimmte Attribute und einige bestimmte Basisklassen hat. Sie müssen sich nicht einmal um das Lesen und Schreiben in der Datenbank kümmern, das machen alles die `Persistencebase`-Klassen. Erstellen Sie nun ein Modul namens `PloneSilverCity.py` im Paket.

Importieren Sie zuerst das Modul `source.py`, das Sie vor wenigen Augenblicken geschrieben haben. Das geht mit einer einfachen Zeile, da sich das Modul im gleichen Paket befindet. Die Zeile, die die Funktion importiert, lautet wie folgt:

```
from source import generate_html, list_generators
```

Als Zweites benötigen Sie eine Klasse `PloneSilverCity`, in der Sie die benötigte Funktionalität kapseln können. Bei dieser Klasse müssen Sie auf die folgenden vier Attribute aufpassen:

- **id**: Speichert die eindeutige ID dieser Instanz der Klasse `PloneSilverCity`.
- **_raw**: Speichert den rohen Quellcode in der Klasse.
- **_raw_as_html**: Speichert den Quellcode, nachdem er in HTML gelext wurde.
- **_raw_language**: Speichert die Sprache dieses Quellcodes.

Für jedes dieser Attribute werden Sie einen *Accessor* schreiben, d.h. eine Methode, die den Wert dieses Attributs zurückgibt, damit Sie nicht das Attribut, sondern die Zugriffsmethode aufrufen. Ein Beispiel für eine Accessor-Methode ist `getLanguage`, die den Wert des language-Attributs zurückgibt. Einen Accessor zu schreiben ist normalerweise eine gute Idee, besonders deswegen, weil Sie später Sicherheitsmechanismen auf diese Accessor-Methoden anwenden werden. In Zope stehen alle Methoden oder Attribute, die mit einem Unterstrich beginnen, nicht für webbasierte Methoden wie Page Templates oder Script (Python)-Objekte zur Verfügung. Ein gute Vorgehensweise besteht darin, am Anfang für all Ihre Attribute einen Unterstrich zu verwenden und dann die Sicherheitsmechanismen auf die Zugriffsmethoden anzuwenden.

Listing 12.1 zeigt die Basisklasse.

Listing 12.1: Die Basisklasse in Python

```
class PloneSilverCity:
    def __init__(self, id):
        self.id = id
```

```
        self._raw = ""
        self._raw_as_html = ""
        self._raw_language = None

    def getLanguage(self):
        """ Returns the language this code has been lexed with """
        return self._raw_language

    def getRawCode(self):
        """ Returns the raw code """
        return self._raw

    def getHTMLCode(self):
        """ Returns the html code """
        return self._raw_as_html

    def getLanguages(self):
        """ Returns the list of languages available """
        langs = []
        for name, description in list_generators():
            langs.append( {'name':lang, 'value':language} )
        langs.sort()
        return langs
```

Eine weitere Methode müssen Sie noch hinzufügen, nämlich eine `edit`-Methode, mit der Sie eine Datei oder einen String hochladen können. Diese Methode liest die Datei und prüft, ob irgendetwas in der Datei steht. Wenn ja, wird sie gelesen und ein Dateiname wird bestimmt. Dann werden der Code, die Sprache und der Dateiname an die Erzeugungsfunktion übergeben. All das speichern Sie in den zuvor genannten Attributen, wie Sie in Listing 12.2 sehen können.

Listing 12.2: Die Methode zur Behandlung von Bearbeitungsschritten

```
def edit(self, language, raw_code, file=""):
    """ The edit function, that sets
    all our parameters, and turns the code
    into pretty HTML """
    filename = ""
    if file:
        file_code = file.read()

        # if there is a file and it's not blank...
        if file_code:
            raw_code = file_code
            if hasattr(file, "name"):
                filename = file.name
```

```
            else:
                filename = file.filename
            # set the language to None so set by SilverCity
            language = None

    self._raw = raw_code

    # our function, generate_html does the hard work here
    html, language = generate_html(raw_code, language, filename)
    self._raw_as_html = html
    self._raw_language = language
```

Hinweis

Python-Entwickler, die sich gut auskennen, sehen eventuell ein Problem darin, `file.name` und `file.filename` zu verwenden. Zope-Dateiobjekte verfügen über ein Attribut namens `filename`, das den Dateinamen repräsentiert, während in Python das Attribut `name` heißt. Dieser Code funktioniert also in normalem Python oder in Zope.

Nun haben Sie also eine Python-Klasse, die das Objekt kapselt. An dieser Stelle sollten Sie die Klasse unter einem Python-Prompt sehr leicht ausführen können, um zu testen, ob sie das macht, was Sie wünschen. Beispiel:

```
$ python
Python 2.3.2 (#1, Oct  6 2003, 10:07:16)
[GCC 3.2.2 20030222 (Red Hat Linux 3.2.2-5)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from PloneSilverCity import PloneSilverCity
>>> p = PloneSilverCity("test.py")
>>> p.edit("python", "print 'hello world'")
>>> p.getRawCode()
"print 'hello world'"
>>> p.getHTMLCode()
'<span class="p_word">print</span>
<span class="p_default">nbsp;</span>
<span class="p_character">\'hellonbsp;world\'</span>'
>>> p.getLanguage()
'python'
```

12.1.4 Aus einem Paket ein Produkt machen

Nun haben Sie ein einfaches Paket, aber das ist noch kein Plone-Produkt. Sie müssen es mit Plone initialisieren. Das heißt, Sie müssen zusätzliche Informationen zum Modul `PloneSilverCity.py` hinzufügen. Insbesondere müssen Sie eine Factory-Funktion hinzufügen. Die Verwendung einer Factory ist ein wohlbekanntes Muster im objektorientierten Design. Sie definiert, wie ein Objekt erzeugt wird. Fügen Sie im Modul `PloneSilverCity.py` den folgenden Konstruktor hinzu:

```
def addPloneSilverCity(self, id, REQUEST=None):
    """ This is our factory function and creates
    an empty PloneSilverCity object """
    obj = PloneSilverCity(id)
    self._setObject(id, obj)
```

Die Funktion `addPloneSilverCity` gehört nicht zur Klasse `PloneSilverCity`. Als Konstruktor für die Klasse wird sie in das Modul außerhalb der Klasse gesetzt. Diese Funktion ist die erste Plone-spezifische Funktion. Der Methode werden drei Parameter übergeben: das Objekt `self`, der ID-String für das Objekt und `REQUEST`. Das Objekt self ist eigentlich der `context`, den Sie zuvor schon gesehen haben, wenn auch unter einem anderen Namen. Da die Objekte immer innerhalb eines Ordners erzeugt werden, zeigt self auf den Ordner, in dem dieses Objekt erzeugt wird. Diese Funktion erzeugt eine Instanz von `PloneSilverCity` namens `obj` und übergibt sie an die Methode `_setObject` des Ordners. Die Methode `_setObject` ist in Zope etwas Besonderes: Sie instanziiert das Objekt in der Datenbank und registriert das Objekt im umgebenden Ordner.

Als Nächstes fügen Sie die Factory-Typinformation hinzu, die in Kapitel 11 behandelt wurde (das ist Ihre erste Chance, sie selbst zu erzeugen). Die Factory-Typinformation enthält in einem Dictionary alle Informationen über den Inhaltstyp. Diese Angaben werden in `portal_types` geladen, wenn das Produkt in Ihrer Plone-Instanz installiert wird. Sie stellen jene Information dar, die Sie vorher gesehen haben, als Sie die Factory-Typinformation über das Web geändert haben.

Bevor ich die Factory-Information erstelle, erstelle ich normalerweise eine Konfigurationsdatei, die alle wiederholten Variablen zu dem Produkt enthält. Diese Datei heißt `config.py`, und darin schreiben Sie wie folgt den Namen des Produkts, den Namen seiner Ebenen in den Skins und den Namen, den der Benutzer dafür sieht:

```
product_name = "PloneSilverCity"
plone_product_name = "Source Code"
layer_name = "silvercity"
layer_location = "PloneSilverCity/skins"
```

Dann können Sie die Factory-Typ-Information erstellen und diese Strings verwenden. Die ID z.B. wird `Source Code` sein, da das den Benutzern in Plone angezeigt wird. Der Aktionsabschnitt der Typinformation ist ein Tupel von Dictionaries aller Aktionen, die auf diesem Objekt vorkommen können. Wenn diese Factory in Plone geladen wird, wird der ACTIONS-Reiter im Werkzeug `portal_types` mit diesem Inhalt gefüllt. All diese Aktionen haben eine entsprechende Methode, ein Template oder Script, das aufgerufen wird. Die meisten davon haben eine direkte Entsprechung zu Page Templates, was ich später in diesem Abschnitt noch erörtern werde.

Wie Sie nun wissen, ist eine Aktion etwas, das Benutzer mit einem Element in der Plone-Datenbank machen können. Mit Blick auf diese Beispielanwendung können die Benutzer zwei offensichtliche Dinge mit dem Quellcode anstellen. Sie können ihn anzeigen und den hübsch hervorgehobenen Code sehen, und sie können das Element bearbeiten und einen Quellcode hochladen. Tatsächlich verlangt Plone, dass es eine Aktion namens *view* und eine Aktion namens *edit* gibt, d.h., diese beiden passen hier gut ins Konzept. Sie werden auch eine dritte Aktion haben wollen: Es ist nett, den Quellcode in seiner ursprünglichen Form herunterladen zu können. In Sprachen wie Python, in denen die Formatierung ein wesentliches Element ist, ist das wirklich nützlich. Diese Aktion zeigt direkt auf `getRawCode`, was die Methode für den Zugriff auf den rohen Code ist.

Jede Aktion hat ein zugehöriges Recht, wie in Listing 12.3 gezeigt wird (wo es genau herkommt, werde ich im weiteren Verlauf dieses Abschnitts zeigen).

Listing 12.3: Die Factory-Typinformationen und Aktionen

```
factory_type_information = (
     'id': plone_product_name,
     'meta_type': product_name,
     'description': ('Provides syntax highlighted HTML of source code.'),
     'product': product_name,
     'factory': 'addPloneSilverCity',
     'content_icon': 'silvercity.gif',
     'immediate_view': 'view',
     'actions': (
                {'id': 'view',
                 'name': 'View',
                 'action': 'silvercity_view_form',
                 'permissions': (view_permission,)},
                {'id': 'edit',
                 'name': 'Edit',
                 'action': 'silvercity_edit_form',
                 'permissions': (edit_permission,)},
                {'id': 'source',
```

```
                'name': 'Source',
                'action': 'getRawCode',
                'permissions': (view_permission,)},
               ),
    }
```

Hinweis

An dieser Stelle kann das Produkt vom Python-Prompt aus nicht importiert werden, weil der Code noch unvollständig ist.

Die Rechte einstellen

Ein fundamentales Konzept bei Websites ist die Annahme, das man nichts und niemandem trauen darf. Bevor ein Zugriff auf eine Eigenschaft erfolgt oder irgendeine Methode aufgerufen wird, müssen Sie zuerst überprüfen, ob die Partei, die eine Aktion ausführen möchte, das auch tun darf. Bei den meisten Systemen gibt es drei Rechte: das Recht, ein Element hinzuzufügen, das Recht, ein Element zu löschen, und das Recht, ein Element zu bearbeiten. Plone kennt ein weiteres Recht: das Recht, ein Element über das Web (oder ein anderes Protokoll) anzuzeigen. In Plone hat der umgebende Ordner das Recht, etwas zu löschen. Wenn Sie etwas in diesem Ordner löschen können, können Sie auch den hier von Ihnen hinzugefügten Inhaltstyp löschen.

Das heißt, Sie müssen sich noch um drei Rechte kümmern. Es ist normal, jene zu benutzen, die im CMFCore-Paket enthalten sind: *Add portal content*, *Modify portal content* und *View*. Wieder zurück in der Konfigurationsdatei können Sie die benötigten Rechte wie folgt ändern:

```
from Products.CMFCore import CMFCorePermissions

add_permission = CMFCorePermissions.AddPortalContent
edit_permission = CMFCorePermissions.ModifyPortalContent
view_permission = CMFCorePermissions.View
```

Das heißt, die Variable `add_permission` steht für das aus CMFCorePermissions importierte Recht. Rechte haben nichts Magisches an sich, jedes einzelne Recht ist lediglich ein String. Die Verwendung des eingebauten Rechts ist bequem und für Ihre Benutzer leicht zu verstehen. Plone ist bereits so konfiguriert, dass es den richtigen Personen erlaubt, Inhalte mit Hilfe des Rechts *Add portal content* zu erstellen. Außerdem ist der Standard-Workflow so definiert, dass er diese Rechte

benutzt und ändert. Diese Rechte waren es, die Sie zur Factory-Typinformation hinzugefügt haben.

Sollten Sie eigene Rechte erstellen wollen, könnten Sie das recht einfach tun. Angenommen, Sie möchten das Recht *Add* in das eigenständige Recht *Add Source Code* ändern. Dann würden Sie die Datei wie folgt ändern:

```
add_permission = "Add Source Code"
```

Nachdem das Produkt importiert worden wäre, hätten Sie im SECURITY-Reiter ein neues Recht namens *Add Source Code*. Warum würden Sie das tun? Nun, es ist bequem, ein Recht zu benutzen, das jeder andere auch benutzt. Allerdings könnte es sein, dass Sie eine feinere Granularität oder andere Sicherheitseinstellungen benötigen. Aus diesem Grund können Sie Ihre eigenen Sicherheitseinstellungen einfach selbst anfertigen.

Die Initialisierung abschließen

Nun müssen Sie die Initialisierung des Produkts einrichten. Das machen Sie im Modul `__init__.py`, damit dann, wenn Zope diese Datei beim Hochfahren liest, es die Initialisierung des Produkts abschließt, wie in Listing 12.4 gezeigt wird.

Listing 12.4: Das Modul __init__.py

```
import PloneSilverCity

from Products.CMFCore import utils
from Products.CMFCore.DirectoryView import registerDirectory

from config import product_name, add_permission

contentConstructors = (PloneSilverCity.addPloneSilverCity,)
contentClasses = (PloneSilverCity.PloneSilverCity,)
contentFTI = (PloneSilverCity.factory_type_information,)

registerDirectory('skins', globals())

def initialize(context):
    product = utils.ContentInit(product_name,
        content_types = contentClasses,
        permission = add_permission,
        extra_constructors = contentConstructors,
        fti = contentFTI)
    product.initialize(context)
```

Was passiert in diesem Code? Nun, eigentlich nicht sehr viel, der Code ist nur ein wenig ausführlich. Zuerst legen Sie Referenzen auf Klassen und Konstruktoren

an, die in `contentClasses` und `contentConstructors` verwendet werden. Diese legen die Factory-Funktion zur Erstellung der Objekte und die eigentliche Klasse fest. Diese werden dann an die Funktion `ContentInit` übergeben, und zwar in `initialize`, einer speziellen Funktion, die während der Produktinitialisierung aufgerufen wird. `ContentInit` verrichtet die ganze Arbeit, um das Produkt in Plone einzurichten. Die Parameter dieser Funktion sind die folgenden:

- **product_name**: Der Name des Produkts, wie er in der Konfigurationsdatei definiert ist (in diesem Fall `PloneSilverCity`).
- **content_types**: Das Tupel der Klassen, die dieses Produkt definieren. Normalerweise ist das nur eine Klasse, es können aber auch mehrere sein.
- **permission**: Das Recht, das benötigt wird, um eine Instanz dieses Objekts zu erzeugen; in diesem Fall die Variable `add_permission`, die ich in `config.py` definiert habe.
- **fti**: Steht für Factory-Typinformation und ist das Dictionary mit der Factory-Typinformation, die Sie im Modul `PloneSilverCity.py` für den Inhalt definiert haben.

12.1.5 Produkt-Module ändern

Nun können Sie zum Modul `PloneSilverCity.py` und zu der Aufgabe zurückkehren, es in ein Plone-Produkt zu verwandeln. Am Anfang des Moduls werden Sie die `import`-Anweisungen erstellen. Diese holen wie folgt aus unterschiedlichen Orten verschiedene Dinge, die für die Plone-Initialisierung notwendig sind:

```
from Globals import InitializeClass
from AccessControl import ClassSecurityInfo
from Products.CMFDefault.DublinCore import DefaultDublinCoreImpl
from Products.CMFCore.PortalContent import PortalContent
```

Diese Import-Anweisungen stellen die Grundfunktionalität für das Produkt bereit und kommen bei den meisten Inhaltstypen vor. Die importierten Definitionen lauten wie folgt:

- **InitializeClass**: Diese Funktion initialisiert die Klasse und wendet alle Sicherheitsdeklarationen an, die diese haben wird. Diese Sicherheitsdeklarationen geben Sie durch die Verwendung der Klasse `ClassSecurityInfo` an.
- **ClassSecurityInfo**: Diese Klasse bietet eine Reihe von Sicherheitsmethoden, mit denen Sie den Zugriff auf die Methoden des Inhaltstyps beschränken können.

- **DefaultDublinCoreImpl**: Diese Klasse enthält eine Implementation von Dublin Core-Metadaten. Der Dublin Core wurde in Kapitel 11 behandelt. Sie verleiht einem Objekt alle Dublin Core-Attribute und -Methoden für den Zugriff darauf, wie Title, Description, Creator usw.
- **PortalContent**: Dies ist die Basisklasse für alle Inhalte in einer Plone-Site und einige der darin benötigten Schlüsselattribute. Durch die Verwendung dieser Basisklasse erhält das Objekt eine ganze Menge an Funktionalität, z.B. zur persistenten Speicherung des Objekts in der Datenbank, der Katalogisierung des Objekts für die Suche mit dem Objekt `portal_catalog` und seiner Registrierbarkeit mit dem Werkzeug `portal_types`.

Außerdem müssen Sie die Konfigurationsvariablen und Rechte importieren. Das erfolgt mit den beiden folgenden Zeilen:

```
from config import plone_product_name, product_name
from config import add_permission, edit_permission, view_permission
```

In der Klasse wiederum müssen Sie zwei Basisklassen hinzufügen, um sie vollständig Plone-kompatibel zu machen: `PortalContent` und `DefaultDublinCoreImpl`. Weiterhin müssen Sie der Klasse einen `meta_type` geben. In Zope hat jedes Produkt einen eindeutigen `meta_type`:

```
class PloneSilverCity(PortalContent, DefaultDublinCoreImpl):
    meta_type = product_name
```

Plone muss wissen, welche Basisklassen der Inhaltstyp implementiert. Andere Teile der Anwendung müssen wissen, welche Klassen implementiert werden. Geben Sie daher wie folgt explizit an, welche Klassen der Inhaltstyp implementiert:

```
__implements__ = (
    PortalContent.__implements__,
    DefaultDublinCoreImpl.__implements__
    )
```

Sicherheit zur Klasse hinzufügen

Wenn Sie bereits entschieden haben, die Aktionen abzusichern, müssen Sie diese Sicherheitsmechanismen auch auf die Klasse anwenden. In einer Umgebung wie Plone, in der Objekte veröffentlicht werden, kann jeder alle Methoden der Klasse über das Web aufrufen, es sei denn, sie beginnen mit einem Unterstrich. Das ist natürlich nicht so gut, und daher müssen Sie all Ihre Methoden schützen.

Um das in der Klasse selbst zu tun, erstellen Sie eine Instanz der Klasse `ClassSecurityInfo` mit der folgenden Zeile:

```
security = ClassSecurityInfo()
```

Dieses Sicherheitsobjekt bietet eine Schnittstelle zum Sicherheitsapparat von Zope. Dann können Sie Methoden auf das Objekt anwenden. Meine Lieblingsmethode dafür besteht darin, direkt über der Methode eine Zeile hinzuzufügen, in der die Sicherheit angebracht wird. Auf diese Weise kann man sich leicht merken, wo die Sicherheit angewendet wird, und später werden Sie nicht vergessen, sie zu aktualisieren, falls das nötig sein sollte. Die Methode `declareProtected` erwartet das Recht und den Methodennamen, um die `edit`-Methode zu schützen. Tun Sie also Folgendes, damit nur diejenigen sie aufrufen können, die das Recht zum Bearbeiten haben:

```
security.declareProtected(edit_permission, "edit")
```

Das wiederholen Sie für jede Methode, wobei Sie das passende Recht und den entsprechenden Methodennamen angeben. Die einzige Methode, die geschützt werden muss, ist `__init__`, weil sie mit einem Unterstrich beginnt. Um diese ganze Sicherheit anzuwenden, müssen Sie die Klasse initialisieren. Ohne diesen Schritt wird die gesamte anschließend deklarierte Sicherheit *nicht* angewendet, d.h., Ihr Objekt wird öffentlich zugänglich sein.

Mit anderen Worten, vergessen Sie die folgende Zeile nicht:

```
InitializeClass(PloneSilverCity)
```

Die API für `ClassSecurityInfo` enthält die folgenden Methoden für die Klasse:

- **declarePublic**: Erwartet eine Liste von Namen. Alle Namen werden für alle Benutzer als öffentlich zugänglich über eingeschränkten Code wie auch über das Web deklariert.
- **declarePrivate**: Erwartet eine Liste von Namen. Alle Namen sind privat und sind nicht über eingeschränkten Code zugänglich.
- **declareProtected**: Erwartet ein Recht und eine beliebige Zahl von Namen. Alle Namen sind nur mit dem angegebenen Recht zugänglich.
- **declareObjectPublic**: Setzt das gesamte Objekt als öffentlich zugänglich.
- **declareObjectPrivate**: Setzt das gesamte Objekt als privat und unzugänglich für eingeschränkten Code.

Mit diesen Methoden können Sie fast jede gewünschte Sicherheit einstellen. Allerdings fand ich es fast immer ausreichend, den Schutz für alle Methoden mit einem bestimmten Recht explizit zu setzen.

Integration mit der Suche

Im vorigen Kapitel habe ich Ihnen gezeigt, wie die Suche funktioniert und welche Indizes existieren. Da die Indizes mit Dublin-Core-Objekten arbeiten und Sie Dublin Core als Basisklasse verwendet haben, werden Titel, Beschreibung, Erzeuger, Änderungsdatum usw. Ihres Objekts alle für Sie indiziert, d.h., es ist keine weitere Arbeit dazu nötig. Durch das Ableiten von der Klasse `PortalContent` wird weiterhin bei jeder Änderung des Objekts der Katalog für Sie aktualisiert. Auch hierbei müssen Sie sich um nichts weiter kümmern.

Ein bestimmter Index jedoch benötigt ein wenig Unterstützung, nämlich `SearchableText`. Wie ich zuvor demonstriert habe, bietet der Index `SearchableText` einen Volltext-Index, den Plone verwendet, wenn eine Suche ausgeführt wird. Es wäre nett, wenn der Index auch im Quelltext suchen würde, damit beim Hochladen eines Codes mit einem `import` darin diese Anweisung von der Suche erfasst würde. Da der Katalog im Objekt nachschaut und versucht, ein Attribut oder eine Methode mit dem entsprechenden Namen des Indexes zu finden, müssen Sie lediglich eine Methode mit diesem Namen erstellen, die den gewünschten Wert zurückgibt.

Am einfachsten macht man das, indem man einen String aus den gewünschten Feldern erstellt, z.B. dem Titel, der Beschreibung und dem rohen Code. Mit dem Recht *View* lässt sich das schützen, denn jeder, der sich das Objekt anschaut, kann den Inhalt sowieso ohne weiteres sehen. Folgendes ist eine `SearchableText`-Methode, die diese Aufgabe erfüllt:

```
security.declareProtected(view_permission, "SearchableText")
def SearchableText(self):
    """ Used by the catalog for basic full text indexing """
    return "%s %s %s" % ( self.Title()
                        , self.Description()
                        , self._raw
                        )
```

Der Unterschied zwischen einer Python- und einer Plone-Klasse

Wie Sie sehen, besteht ein beträchtlicher Unterschied zwischen einem normalen Python-Produkt und einem, das in Plone registriert ist. Allerdings liegen die meisten dieser Unterschiede darin, wie das Produkt registriert und die Sicherheit gewährleistet wird. Die eigentliche Klasse ist sehr ähnlich. Listing 12.5 beschreibt alle Unterschiede zwischen der reinen Python- und der Plone-Implementierung.

Listing 12.5: Die Plone-Version der Klasse

```
from Globals import InitializeClass
from AccessControl import ClassSecurityInfo
from Products.CMFDefault.DublinCore import DefaultDublinCoreImpl
from Products.CMFCore.PortalContent import PortalContent

from config import meta_type, product_name
from config import add_permission, edit_permission, view_permission
from source import generate_html, list_generators

factory_type_information = {
     'id': plone_product_name,
     'meta_type': product_name,
     'description': ('Provides syntax highlighted HTML of source code.'),
     'product': product_name,
     'factory': 'addPloneSilverCity',
     'content_icon': 'silvercity.gif',
     'immediate_view': 'view',
     'actions': (
                 {'id': 'view',
                  'name': 'View',
                  'action': 'silvercity_view_form',
                  'permissions': (view_permission,)},
                 {'id': 'source',
                  'name': 'View source',
                  'action': 'getRawCode',
                  'permissions': (view_permission,)},
                 {'id': 'edit',
                  'name': 'Edit',
                  'action': 'silvercity_edit_form',
                  'permissions': (edit_permission,)},
                 ),

     }

def addPloneSilverCity(self, id, REQUEST=None):
    """ This is our factory function and creates
    an empty PloneSilverCity object inside our Plone
    site """
    obj = PloneSilverCity(id)
    self._setObject(id, obj)

class PloneSilverCity(PortalContent, DefaultDublinCoreImpl):

    meta_type = product_name
```

```
__implements__ = (
    PortalContent.__implements__,
    DefaultDublinCoreImpl.__implements__
    )

security = ClassSecurityInfo()
def __init__(self, id):
    DefaultDublinCoreImpl.__init__(self)
    self.id = id
    self._raw = ""
    self._raw_as_html = ""
    self._raw_language = None

security.declareProtected(edit_permission, "edit")
def edit(self, language, raw_code, file=""):
    """ The edit function, that sets
    all our parameters, and turns the code
    into pretty HTML """
    filename = ""
    if file:
        file_code = file.read()

        # if there is a file and its not blank...
        if file_code:
            raw_code = file_code
            if hasattr(file, "name"):
                filename = file.name
            else:
                filename = file.filename
            # set the language to None so set by SilverCity
            language = None

    self._raw = raw_code

    # our function, generate_html does the hard work here
    html, language = generate_html(raw_code, language, filename)
    self._raw_as_html = html
    self._raw_language = language

security.declareProtected(view_permission, "getLanguage")
def getLanguage(self):
    """ Returns the language that code has been lexed with """
    return self._raw_language

security.declareProtected(view_permission, "getLanguages")
def getLanguages(self):
    """ Returns the list of languages available """
```

```
        langs = []

        for name, description in list_generators():
            # these names are normally in uppercase
            langs.append( {'name':lang, 'value':language } )

        langs.sort()
        return langs

    security.declareProtected(view_permission, "getRawCode")
    def getRawCode(self):
        """ Returns the raw code """
        return self._raw

    security.declareProtected(view_permission, "getHTMLCode")
    def getHTMLCode(self):
        """ Returns the html code """
        return self._raw_as_html

    security.declareProtected(view_permission, "SearchableText")
    def SearchableText(self):

        """ Used by the catalog for basic full text indexing """

        return "%s %s %s" % ( self.Title()
                            , self.Description()
                            , self._raw
                            )

InitializeClass(PloneSilverCity)
```

12.1.6 Skins hinzufügen

Nun, da Sie den Hauptcode haben, bleiben noch zwei Dinge für Sie zu tun: die Skins und eine Installationsmethode erstellen. Die Skins gehören zu den einfacheren Teilen, weil ein Großteil der notwendigen Arbeit bereits vom Plone-Framework erledigt wird. Skins habe ich detailliert in vorangegangenen Kapiteln behandelt, in denen ich erörtert habe, wie man im Dateisystem eine Skin für eine Plone-Site erstellt. Jedes Produkt, das eine eigene Benutzerschnittstelle haben muss (UI, User Interface), macht das mit einem eigenen FSDV (File System Directory View), d.h., hier werden Sie erneut das Gleiche machen.

Skins kommen in das Verzeichnis `skins` des Produkts. Dieser Verzeichnisname wird in der Datei `__init__.py` definiert, wo Sie das Verzeichnis mit der Funktion `registerDirectory` registrieren. Wenn Sie diesen Namen ändern möchten, sollten

Sie ihn auf jeden Fall registrieren. Sie können so viele Verzeichnisse registrieren, wie Sie wollen, aber das ist rekursiv und dabei wird auch alles in und unter diesem registrierten Verzeichnis registriert.

Die einfachste all Ihrer Aufgaben bei diesem Produkt besteht darin, ein Icon für das Objekt hinzuzufügen, das in Plone erscheint. Der Name dieses Icons wird bereits in der Factory-Typinformation mit der Zeile `'content_icon': 'silvercity.gif'` definiert, d.h., Sie müssen lediglich ein Icon namens `silvercity.gif` im Verzeichnis `skins` hinzufügen. Dieses Icon wird immer dann angezeigt, wenn Sie das Objekt in der Benutzerschnittstelle von Plone anzeigen. Wenn SilverCity eine Datei lext, gibt es HTML aus, in dem CSS-Tags benutzt werden, d.h., Sie müssen sicherstellen, dass die entsprechende CSS-Datei auch verfügbar ist. Bei diesem Produkt kopieren Sie einfach das CSS aus dem SilverCity-Produkt und setzen es in das Verzeichnis `skins`.

Diese zwei Dinge sind nun erledigt. Nun müssen Sie als Nächstes wirklich die Seiten zum Anzeigen und Bearbeiten der Seiten schreiben. Ich habe zuvor die Ähnlichkeit zu einem Dokument beschrieben, wenn Sie also nach Seiten zum Anzeigen und Bearbeiten suchen, dann sind die Seiten zu einem Dokument der beste Platz dafür. Dies sind `document_view.pt` und `document_edit_form.cpt`, die sich im Verzeichnis `CMFPlone/skins/plone_content` befinden.

Die Anzeigen-Seite erstellen

Um die ANZEIGEN-Seite zu ändern, nehmen Sie die ANZEIGEN-Seite zu einem Dokument, kopieren sie in das Verzeichnis `skins` Ihres Produkts und benennen sie in `silvercity_view.pt` um. Es gibt keinen Grund, die gesamte Seite neu zu erstellen, wenn diese ANZEIGEN-Seite so ähnlich ist und Sie lediglich zwei kleine Änderungen vornehmen müssen.

Wie schon erwähnt, spuckt SilverCity HTML aus, in dem der ganze Code mit Hilfe von CSS hervorgehoben wurde, zu dem Sie ein eigenes Stylesheet haben. Sie müssen dafür sorgen, dass die ANZEIGEN-Seite dieses CSS einfügt und das Haupt-Template einen Slot für CSS namens *css_slot* besitzt. Um die eigene CSS-Datei in diesen Slot einzufügen, müssen Sie dafür nur einen Wert angeben. Beispiel:

```
<metal:cssslot fill-slot="css_slot">

<link
   rel="stylesheet"
   href=""
   tal:attributes="href string:$portal_url/silvercity.css" />
</metal:cssslot>
```

Hierbei referenzieren Sie eine CSS-Datei namens `silvercity.css`. Diese befindet sich im Verzeichnis `skins`, und Sie werden aus der Skin darauf zugreifen, wenn diese dargestellt wird. Das Originaldokument hat eine Eigenschaft namens `cookedBody`, was ein Attribut eines Dokuments ist. Ich habe diesen Teil des Codes entfernt und stattdessen den Code eingefügt. Wie Sie bis hierher gesehen haben, gibt die Funktion `getHTMLCode` das HTML zurück, also müssen Sie nur noch Folgendes tun:

```
<div id="bodyContent">
    <div tal:replace="structure here/getHTMLCode" />
</div>
```

Wenn Sie irgendetwas anderes Bestimmtes in diesem Page Template ändern möchten, dann ist jetzt die Gelegenheit dazu. Es wäre eventuell nett, z.B. die Sprache anzuzeigen, in der es geschrieben wurde, oder ein Icon, oder seinen Verlauf zu ändern.

Die Bearbeiten-Seite erstellen

Wie bei der ANZEIGEN-Seite können Sie auch die BEARBEITEN-Seite nehmen, in die Skin kopieren und dort in `silvercity_edit_form.cpt` umbenennen. Das größte Problem dabei ist, dass das BEARBEITEN-Formular so entworfen ist, dass es mit einem WYSIWYG-Editor (What You See Is What You Get) wie Epoz benutzt werden kann. Solange kein guter WYSIWYG-Editor für Quellcode in Webbrowsern verfügbar ist, müssen Sie das ausschalten, weil Sie in einem HTML-Editor z.B. kein SQL schreiben können.

Dies ist eine ziemlich große Änderung des Page Templates. Erinnern Sie sich daran, dass Sie es von der Website zum Plone-Buch herunterladen können. Entfernen Sie in diesem Template alle Stellen, in denen der Editor erwähnt wird, und ersetzen Sie sie durch einen einfachen HTML-Textbereich. Lassen Sie den Namen des HTML-Felds unverändert, denn es gibt keinen guten Grund, ihn zu ändern. Außerdem verhält es sich später dann so, wie es soll mit dem Script, das das Formular auswertet. Ein Dokument hat unten eine Reihe von Auswahlmöglichkeiten für das Format, was normalerweise Einträge wie *Einfacher Text*, *HTML* usw. sind. Das werden Sie durch ein Dropdown-Menü für alle Sprachen ersetzen, über die die SilverCity-Hauptbibliothek verfügt. Die zuvor geschriebene Methode `getLanguages` gibt eine Liste aller Sprachen zurück. Jedes Element ist ein Dictionary, das den Wert, z.B. CPP, und einen netten Namen dafür, z.B. C oder C++, enthält.

Listing 12.6 geht in einer Schleife über die zuvor geschriebene Methode `getLanguages`. Sie können auch eine Laufvariable für die aktuelle Sprache definieren, damit Sie die aktuelle Sprache in der Schleife über die Sprachen hervorheben können.

Listing 12.6: Hinzufügen eines Dropdown-Menüs für die Sprachauswahl

```
<div class="field">
  <label
   for="language"
   i18n:translate="label_silvercity_language">
   Language</label>

    <div class="formHelp" i18n:translate="help_silvercity_language">
        Select the name of the language that you are adding
    </div>
    <select name="text_format"
            tal:define="l here/getLanguage">
        <option tal:repeat="item here/getLanguages"
            tal:content="item/name"
            tal:attributes="value item/value;
                            selected python:test(item['value'] == l, 1, 0)" />
    </select>
</div>
```

Wenn die BEARBEITEN-Seite abgeschickt wird, müssen Sie die Validierer und Aktionen so einrichten, dass sie etwas mit dem Formular machen. Die Validierung sollte überprüfen, ob ein gültiger Titel und eine gültige ID angegeben wurden. Fügen Sie Folgendes zur Datei `silvercity_edit.cpt.metadata` hinzu:

```
[validators]
validators..Save = validate_id,validate_title
validators..Cancel =
```

Woher kommen diese Validierungen? Nun, ich habe ein wenig geschummelt und habe mir wieder die Validierungen eines Dokuments angeschaut. Dabei erfolgen drei Validierungen, von denen Sie aber nur zwei benötigen. Durch die Überprüfung dessen, was diese Validierung ergibt, können Sie sehen, welche benötigt werden und welche nicht. Sie finden alle Validierungen in `plone_skins/plone_form_scripts`, und deren Objektname beginnt mit `validation`.

Nun benötigen Sie die Aktion, also nehmen Sie das Bearbeiten-Script eines Dokuments (`document_edit.cpy`) und kopieren es nach SilverCity. Zum größten Teil ist das Script so in Ordnung, d.h., Sie können es mit nur einer Änderung beibehalten. Ändern Sie die Meldungen von `Document` in `Source code`. Listing 12.7 zeigt das Bearbeiten-Script.

Listing 12.7: Das Bearbeiten-Script

```
##parameters=text_format, text, file='', SafteyBelt='', ~↵
title='', description='', id=''
##title=Edit a document
```

```
filename=getattr(file,'filename', '')
if file and filename:
    # if there is no id, use the filename as the id
    if not id:
        id = filename[max( filename.rfind('/')
                         , filename.rfind('\\')
                         , filename.rfind(':') )+1:]
    file.seek(0)

# if there is no id specified, keep the current one
if not id:
    id = context.getId()

new_context = context.portal_factory.doCreate(context, id)
new_context.edit( text_format
                , text
                , file
                , safety_belt=SafetyBelt )

from Products.CMFPlone import transaction_note
transaction_note('Edited source code %s at %s' % ~↵
  (new_context.title_or_id(), new_context.absolute_url()) ~↵
  )

new_context.plone_utils.contentEdit( new_context
                                   , id=id
                                   , title=title
                                   , description=description )

return state.set(context=new_context, ~↵
 portal_status_message='Source code changes saved.')
```

Dieses Script macht ein paar Dinge. Zuerst holt es den Dateinamen, falls einer existiert. Falls keine ID angegeben ist, wird die ID auf diesen Dateinamen gesetzt. Das bedeutet: Wenn ein Benutzer `library.c` hochlädt, wird die ID dieses Objekts `library.c` sein. Als Zweites weist es `portal_factory` an, ein Objekt zu erstellen (siehe den Exkurs »Portal Factory« für weitere Informationen darüber, was das bedeutet). Dann ruft es die `edit`-Methode auf dem Objekt (die Sie zuvor geschrieben haben) und `contentEdit` im Werkzeug `plone_utils` auf. Ohne jetzt tiefer ins Werkzeug `plone_utils` zu schauen, nimmt `contentEdit` die angegebenen Schlüsselwörter und ändert diese Attribute, falls die Klasse Dublin Core implementiert. Da Sie das Attribut `__implements__` vorher eingerichtet haben, wird die `edit`-Methode in Listing 12.7 diese Arbeit für Sie erledigen. Alle Änderungen an Titel, ID oder Beschreibung werden im Objekt geändert.

Exkurs: Portal-Factory

Es gibt ein Problem mit der Art, wie Objekte erzeugt werden. Bevor Sie überhaupt zum BEARBEITEN-Formular kommen können, müssen Sie ein Objekt erzeugen. Dann wird das BEARBEITEN-Formular zu diesem Objekt angezeigt. In der Praxis erzeugen die Leute oft unabsichtlich ein Objekt, kommen zum BEARBEITEN-Formular und stellen dann fest, dass es den falschen Typ hat. Das ist ärgerlich und führt dazu, dass ungenutzte Objekte in Ihrer Datenbank herumliegen. Das ist, als ob Sie eine Datei im Dateisystem erzeugen, dann feststellen, dass es eine falsche Datei ist, und sie dann liegen lassen.

Um das zu lösen, können Sie mit dem Werkzeug `portal_factory` Objekte temporär erzeugen. Es erzeugt ein temporäres Objekt und lässt Sie es dann bearbeiten. Aber erst nachdem Sie den BEARBEITEN-Button angeklickt haben, wird Ihr Objekt erzeugt. Um ein Objekt an `portal_factory` zuzuweisen, gehen Sie zum Werkzeug `portal_factory` und wählen im Formular alle Inhaltstypen, für die Sie dieses Werkzeug benutzen möchten. Der einzige Nachteil ist der, dass Sie garantieren müssen, dass Ihre Bearbeiten-Scripten korrekt mit dem Werkzeug zusammenarbeiten, wie in diesem Beispiel gezeigt wird.

12.1.7 Installation des Produkts in Plone

Es gibt eine standardisierte Art, ein Produkt in Plone zu installieren: Sie gehen ins Plone-Control Panel und klicken auf das Produkt, um es zu installieren. Dieses Script benutzt das Werkzeug `portal_quickinstaller` für die Installation. Damit dieses Produkt funktioniert, müssen Sie eine gewisse Funktionalität offen legen, die das Werkzeug lesen kann. Schließlich wollen Sie, dass so viele Leute wie möglich Ihr Produkt benutzen. Wenn Sie etwas nur für den internen Gebrauch schreiben und Sie es nicht an andere Leute verteilen, können Sie diesen Schritt auslassen. Aber Sie müssen diese Schritte sowieso von Hand ausführen, und es ist immer besser, man hat ein Script für die Installation.

Für eine Integration mit dem Quick Installer müssen Sie ein spezielles Modul namens `Install.py` im Verzeichnis `Extensions` erstellen. Dieses Modul muss eine Funktion namens `install` enthalten. Das Quick Installer-Werkzeug führt die Funktion `install` aus, und die Ausgabe landet in einer Datei auf dem Server. Da die Methode `install` das Produkt in den Portal-Typen installieren muss, fügen Sie nun ein FSDV hinzu, das auf das Verzeichnis `skins` zeigt, und fügen dieses neue Verzeichnis zu den Skin-Ebenen hinzu.

Hinweis

Der Quick Installer macht aus dieser Installationsfunktion eine externe Methode und führt sie hinter den Kulissen für Sie aus. Außerdem führt er auch noch einige weitere Aufgaben aus. Das heißt, wenn Sie möchten, könnten Sie eine externe Methode erstellen, um das zu machen. Deswegen steht in den Installationsanweisungen zu den meisten Produkten, dass Sie eine externe Methode erstellen sollen.

Importieren Sie nun die Funktionen, und richten Sie die Variablen wie üblich ein. Die `factory_type_information` müssen Sie aus dem Produkt installieren, damit Sie es im Script verwenden können, wie in Listing 12.8 zu sehen ist.

Listing 12.8: Der Anfang der Installationsfunktion

```
from Products.CMFCore.TypesTool import ContentFactoryMetadata
from Products.CMFCore.DirectoryView import createDirectoryView
from Products.CMFCore.utils import getToolByName
from Products.PloneSilverCity.PloneSilverCity import factory_type_information

from Products.PloneSilverCity.config import plone_product_name, product_name
from Products.PloneSilverCity.config import layer_name, layer_location

def install(self):
    """ Install this product """
```

Danach ist alles generisch und könnte auf jedem Produkt laufen, außer dann natürlich, wenn Sie möchten, dass es bei der Installation irgendwas Spezielles tut. Um Ihr Produkt zum Werkzeug `portal_types` hinzuzufügen, überprüfen Sie zuerst, ob Ihr Produkt schon registriert ist. Es könnte sein, dass ein anderer schon ein anderes Produkt mit dem gleichen Namen installiert hat. Dazu rufen Sie die Methode `manage_addTypeInformation` auf, wie in Listing 12.9 angegeben.

Listing 12.9: Rest der Installationsfunktion

```
out = []
typesTool = getToolByName(self, 'portal_types')
skinsTool = getToolByName(self, 'portal_skins')

if id not in typesTool.objectIds():
   typesTool.manage_addTypeInformation(
       add_meta_type =  factory_type_information['meta_type']
       id = factory_type_information['id']
       )
```

```
    out.append('Registered with the types tool')
else:
    out.append('Object "%s" already existed in the types tool' % (id))
```

Als Nächstes müssen Sie ein FSDV im `skins`-Verzeichnis hinzufügen. Wieder müssen Sie als Erstes prüfen, ob Sie nicht schon eines haben. Dann fügen Sie die Verzeichnisansicht wie folgt hinzu:

```
if skinname not in skinsTool.objectIds():
    createDirectoryView(skinsTool, skinlocation, skinname)
    out.append('Added "%s" directory view to portal_skins' % skinname)
```

Und schließlich iterieren Sie über alle Skins und fügen Ihr neues FSDV zu allen Skins hinzu. Dies ist eine generische Funktion. Jede Skin wird als String aufgeführt, in dem alle Ebenen mit Kommata voneinander getrennt sind. Nun müssen Sie den String nur noch aufteilen und Ihre neue Skin nach der Ebene namens `custom` einfügen wie in Listing 12.10 zu sehen ist.

Listing 12.10: Die Skin in der Installationsmethode setzen

```
skins = skinsTool.getSkinSelections()
for skin in skins:
    path = skinsTool.getSkinPath(skin)
    path = [ p.strip() for p in p.split(',') ]
    if skinname not in path:
        path.insert(path.index('custom')+1, skinname)

        path = ", ".join(path)
        skinsTool.addSkinSelection(skin, path)
        out.append('Added "%s" to "%s" skins' % (skinname, skin))
    else:
        out.append('Skipping "%s" skin' % skin)
return "\n".join(out)
```

Das wars. Nun ist Ihr Produkt bereit, ausgeführt zu werden.

12.1.8 Das Produkt testen

Um das Produkt zu testen, starten Sie Ihre Plone-Instanz neu, damit sie das Produktverzeichnis einliest. Wenn Sie Ihr Produkt noch nicht im entsprechenden Produktverzeichnis entwickelt haben, dann kopieren Sie es jetzt dorthin, damit es Teil Ihres normalen Installationsprozesses wird. Wenn es mit Ihrem Produkt irgendwelche Probleme gibt, dann startet Zope eventuell noch, aber das Produkt wird dann unter Umständen im Control Panel als defekt angezeigt.

Dann installieren Sie es in Plone mit Hilfe der Seite PRODUKTE HINZUFÜGEN/LÖSCHEN im Plone-Control Panel. Nun sollten Sie zu einem Ordner gehen und ein Quellcode-Objekt hinzufügen können. Das Icon wird Ihr Icon in der Skin sein, und der Name ist jener, den Sie im Dateisystem definiert haben. Danach erhalten Sie die BEARBEITEN-Seite. Beachten Sie, dass die URL nun mit `silvercity_edit_form` endet und das passend geänderte BEARBEITEN-Formular anzeigt.

Sie könnten weiteren Code hinzufügen, eine Sprache auswählen und auf SPEICHERN klicken, oder Sie könnten eine Datei von Ihrem Computer hochladen. Nach einem Klick auf SPEICHERN gelangen Sie zu der Anzeigefunktion zurück, und es wird hoffentlich der Code mit der hervorgehobenen Syntax angezeigt.

Dieses Produkt ist ein kleines Beispiel dafür, wie einfach man ein Produkt in Plone schreiben kann. Auch wenn es viele Seiten waren, wurden auf den meisten die Infrastruktur und die Skins eingerichtet. Es läge jetzt nahe, das mit anderen Web-Scriptsprachen wie PHP zu vergleichen. Aber Sie müssen daran denken, dass Sie dadurch, dass Ihr Code in Plone ist, eine ganze Menge Dinge erreicht haben, ohne dafür etwas neu schreiben zu müssen. Insbesondere haben Sie Folgendes erreicht:

- Volltext-Suche im Inhalt
- Integration mit dem Workflow
- Integration mit Portal-Mitgliedschaft und -Authentifikation
- Persistenz dank der Plone-Datenbank, ohne SQL schreiben oder andere Arbeit verrichten zu müssen

Außerdem kann später dadurch Ihr ganzes Produkt wirklich besser skalieren. Wenn Sie z.B. ein Bug-Tracking-System benötigen, nehmen Sie das Collector-Produkt hinzu, und wenn Sie ein Produkt zur Bilderverwaltung benötigen, nehmen Sie CMFPhoto. Indem Sie das Framework verwenden, verleihen Sie Ihrer ganzen Site eine Menge an Flexibilität und Skalierbarkeit.

Auch wenn dieses Produkt insofern ein wenig mogelt, als dass es eine Menge an vorhandenem Code wiederverwendet, so demonstriert es doch eine ganze Reihe von Schlüsselfunktionen beim Schreiben eines Produkts in Plone.

12.1.9 Fehlersuche bei der Entwicklung

Wenn Sie Ihr eigenes Produkt entwickeln, werden Sie früher oder später zwei Dinge feststellen (es sei denn, Sie verfügen über so viel Zen, dass Sie eigentlich am Code des Zope-Kerns mitschreiben sollten): Ihr Produkt versagt, und Sie müssen den Fehler finden.

Während der Entwicklung möchten Sie eventuell versuchen, das Produkt vom Python-Prompt aus zu importieren, um zu sehen, wie es funktioniert. Leider werden Sie dabei sehr wahrscheinlich einen Fehler bekommen. Das liegt daran, dass Sie bei diesem Import eine Lawine an Zope-bezogenen Imports auslösen. Mit einigen davon können Sie fertig werden, aber nicht mit allen. Eines der häufigen Probleme besteht darin, dass Sie den folgenden Fehler bekommen:

```
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  File "PloneSilverCity/__init__.py", line 1, in ?
    import PloneSilverCity
  File "PloneSilverCity/PloneSilverCity.py", line 4, in ?
    from Globals import InitializeClass
  File "/opt/Zope-2.7/lib/python/Globals.py", line 23, in ?
    import Acquisition, ComputedAttribute, App.PersistentExtra, os
  File "/opt/Zope-2.7/lib/python/App/PersistentExtra.py", line 15, in ?
    from Persistence import Persistent
ImportError: cannot import name Persistent
```

Das können Sie lösen, indem Sie sicherstellen, dass Sie vor dem Import von PloneSilverCity erst Zope importieren und die Startup-Methode ausführen. Um Zope importieren zu können, muss das Verzeichnis, das Zope enthält, in Ihrem Pfad enthalten sein. Auf meinem Computer ist das `/opt/Zope-2.7/lib/python`. Allerdings werden Sie dann beim Versuch, CMFCore zu importieren, Fehler erhalten, falls Sie Zope mit der Option »Instance Home« konfiguriert haben, was vermutlich der Fall ist.

Am leichtesten kann man PloneSilverCity importieren, indem man Zope mit `zopectl` von der Kommandozeile im Debug-Modus ausführt. Dabei wird ein Python-Prompt aufgemacht, unter dem Sie von Python direkt auf die Zope-Datenbank zugreifen können. Kapitel 14 behandelt das etwas detaillierter, aber Sie können es auch leicht jetzt tun (vorausgesetzt, dass Ihr Zope gerade nicht läuft). Das Script `zopectl` finden Sie im Verzeichnis `bin` Ihrer Zope-Instanz. Auf meinem Computer z.B. ist das `/var/zope/bin`. Listing 12.11 zeigt ein Beispiel eines laufenden `zopectl` mit PloneSilverCity.

Hinweis

Beim Niederschreiben dieser Zeilen funktioniert `zopectl` nicht unter Windows. Auf Linux jedoch ist es eine bequeme Art, Ihren Code zu testen. Leider verlangt `zopectl`, dass man die Zope-Objektdatenbank (ZODB) sperrt, was man nicht machen kann, während Zope läuft, es sei denn, Sie setzen ZEO ein (was ich in Kapitel 14 erörtern werde).

Listing 12.11: Fehlersuche im Produkt mittels Zope

```
$ cd /var/zope/bin
$ ./zopectl debug
Starting debugger (the name "app" is bound to the top-level Zope object)
>>> from PloneSilverCity.PloneSilverCity import PloneSilverCity
>>> p = PloneSilverCity("test")
>>> p.edit("python", "import test")
>>> p.getRawCode()
'import test'
>>> p
<PloneSilverCity at test>
```

Falls Ihr Produkt defekt ist, bekommen Sie einen Traceback zu einer von zwei Stellen, entweder zur Fehlerprotokollseite oder zu einem der Ereignisprotokolle. Und wenn Sie es wirklich vermasselt haben, dann startet Plone erst gar nicht. Das passiert normalerweise, wenn ein Import fehlschlägt. Wenn das der Fall ist, startet Plone gar nicht erst. Ich empfehle dann, Plone von der Kommandozeile aus zu starten, egal ob unter Windows oder Linux. Mit der Ausgabe des Fehlers in der Konsole haben Sie eine sofortige Fehlermeldung. Listing 12.12 z.B. zeigt, was passiert, wenn Sie versuchen, Plone mit SilverCity zu starten, falls dabei ein beliebiger Import-Fehler auftritt.

Listing 12.12: Ein Beispielfehler beim Hochfahren

```
$ bin/runzope
------
2003-12-19T17:44:05 INFO(0) ZServer HTTP server started at Fri Dec 19 17:44:05
2003
        Hostname: laptop
        Port: 8080
------
2003-12-19T17:44:05 INFO(0) ZServer FTP server started at Fri Dec 19 17:44:05
2003
        Hostname: basil.agmweb.ca
        Port: 8021
------
2003-12-19T17:44:16 ERROR(200) Zope Could not import Products.PloneSilverCity
Traceback (most recent call last):
  File "/opt/Zope-2.7/lib/python/OFS/Application.py", line 533, in
import_product
    product=__import__(pname, global_dict, global_dict, silly)
  File "/var/zope.zeo/Products/PloneSilverCity/__init__.py", line 1, in ?
    import PloneSilverCity
  File "/var/zope.zeo/Products/PloneSilverCity/PloneSilverCity.py", line 1
     import ThisModuleDoesNotExist
```

```
                ^
ImportError: No module named ThismoduleDoesNotExist
```

An diesem Punkt stoppt Zope, und Sie müssen diesen Import reparieren, bevor Sie es erneut starten können. Wahrscheinlich ist das der am leichtesten zu reparierende Fehler. Er kommt aber wahrscheinlich nur dann vor, wenn Sie das neueste hipp Produkt aus dem Internet installieren, um festzustellen, dass es von einem Dutzend anderen Modulen abhängt.

Die nächste Art möglicher Fehler ist die von Fehlern innerhalb des Programms oder der Programmlogik. Angenommen, Ihr Produkt addiert zwei Zahlen, von denen eine aber ein String ist (was in Python ein Fehler ist). Es wird eine Fehlerausnahme aufgelöst, die Plone mit dem Fehlerwert und -typ an die Benutzerschnittstelle weitergibt. An dieser Stelle sollten Sie auf PLONE-KONFIGURATION und FEHLERPROTOKOLL klicken, um den Traceback anzuschauen, den Fehler zu finden und das Problem zu lösen.

Wenn Sie etwas im Produkt ändern, wird diese Änderung in Python nicht sofort umgesetzt. Um das zu erzwingen, müssen Sie ein Produkt namens *Refresh* verwenden. Das ist ein unglaublich nützliches Werkzeug für neue Entwickler, das Sie dadurch aktivieren, dass Sie eine Datei namens `refresh.txt` in Ihrem Verzeichnis `Products` haben. Sie werden bemerken, dass PloneSilverCity eine solche hat. Dann wählen Sie im ZMI das Control Panel, dann PRODUCTS, gefolgt von PLONESILVERCITY (oder dem Namen Ihres Produkts), und klicken anschließend auf den REFRESH-Reiter. Falls Ihr Produkt eine Datei namens `refresh.txt` enthält, können Sie auf den REFRESH-Button klicken. Plone wird Ihr Produkt dann dynamisch mitsamt dem ganzen neuen Code neu laden. Falls Sie Zope im Debug-Modus betreiben, können Sie das Produkt so einstellen, dass es diese Überprüfung bei jedem Lauf dynamisch vornimmt, so dass Sie nicht jedes Mal zu diesem Bildschirm zurückkehren müssen.

Leider gibt es hierbei nicht nur Vorteile. Hinter den Kulissen passieren nämlich einige recht »interessante« Dinge mit Python, damit das funktioniert. Tatsächlich kann das Refresh-Produkt zu unerwarteten Ergebnissen in Ihrem Produkt führen, wenn auch zu nichts, was nicht durch einen Neustart von Zope zu reparieren wäre. Relativ einfache Produkte, die lediglich Manipulationen an Daten vornehmen, werden keine Probleme haben, andere hingegen schon. Zu Beginn werden Ihre Produkte jedoch eher einfacher Natur sein, und die Chancen stehen gut, dass Sie an dieser Stelle keine Probleme haben werden.

Und wenn schließlich doch etwas schief geht und Sie nicht herausfinden können, was es ist, dann müssen Sie zu einem Debugger greifen. Sie haben derart viele Möglichkeiten bei der Fehlersuche mit Zope, dass ich hier nur die eine beschreiben möchte, die ich selbst am meisten benutze, den Python-Debugger. Diesen

Python-Debugger rufen Sie auf, indem Sie die folgende Zeile zu einem Stück Code hinzufügen:

```
import pdb; pdb.set_trace()
```

Tipp

In Python ist es unüblich, zwei Programmzeilen mit einem Semikolon hintereinander zu setzen. Hier ist das aber sehr praktisch, da später beim Löschen oder Auskommentieren nur eine Zeile betroffen ist.

Das bewirkt, dass ein Breakpoint in der Ausführung des Codes gesetzt wird, an dem Zope die Abarbeitung anhält und den Debugger aufruft. Das ist der Grund, warum man während der Entwicklung Plone wirklich besser von einer Konsole startet. Mit einem Dienst oder Daemon funktioniert das nicht, weil es keine Konsole gibt, mit der eine Verbindung möglich wäre. Wenn Sie Ihr Problem nun reproduzieren, gelangen Sie zum Python-Debugger, mit dem Sie den Fehler in Ihrem Produkt suchen können. In meinem nun reparierten und korrekt importierenden PloneSilverCity habe ich z.B. die folgende `pdb`-Trace-Funktion in die Methode `getLanguages` gesetzt:

```
def getLanguages(self):
     """ Returns the list of languages available """
     import pdb; pdb.set_trace()
     langs = []
     ...
```

Wenn Sie Zope nun starten und sich mit der Skin verbinden (was Sie bald hinzufügen werden), wird diese Funktion aufgerufen, und in der Konsole, in der Sie Zope gestartet haben, werden Sie Folgendes sehen können:

```
--Return--
> /var/tmp/python2.3-2.3.2-root/usr/lib/python2.3/pdb.py(992)set_trace()->None
-> Pdb().set_trace()
(Pdb)
```

Um eine Liste von Hilfestellungen zu erhalten, können Sie **help** eingeben. Die zwei wesentlichen Optionen sind `n` für nächstes und `s` für einen Schritt in ein Element hinein. Beispiel:

```
(Pdb) n
> /var/zope.zeo/Products/PloneSilverCity/PloneSilverCity.py(97)getLanguages()
-> langs = []
```

```
(Pdb) n
> /var/zope.zeo/Products/PloneSilverCity/PloneSilverCity.py(99)getLanguages()
-> for value, description in list_generators():
(Pdb) langs
[]
```

Für weitere Informationen zum Debugger empfehle ich die Online-Dokumentation auf der Python-Website (*http://python.org/doc/current/lib/module-pdb.html*). Sie haben auch andere Möglichkeiten, Fehler mit Zope zu suchen, z.B. können Sie ZEO verwenden, um einen Interpreter zu bekommen. ZEO wird in Kapitel 14 behandelt. Mit integrierten Entwicklerumgebungen wie *Wing* (*http://wingide.com/wingide*) oder *Komodo* (*http://www.activestate.com/Products/Komodo*) können Sie Fehler in Zope-Instanzen auch auf entfernten Rechnern suchen und haben dabei noch eine nette grafische Benutzerschnittstelle.

12.2 Eigene Werkzeuge schreiben

Ein Werkzeug ist vor allem deswegen einfacher zu schreiben als ein Inhaltstyp, weil man wenig tun muss, um das Produkt zu registrieren und weil die Benutzerschnittstelle einfach ist. Beispiel: ich benutze ein einfaches Statistikwerkzeug auf meiner ZopeZen-Website (*http://www.zopezen.org*), das mir Informationen über die Menge an Inhalten, die Anzahl der Benutzer usw. gibt. Dieses einfache Werkzeug gibt ein paar Zahlen aus, die mich als Manager der Site interessieren. Abbildung 12.2 zeigt meine ZopeZen-Statistik.

Das sind Statistiken über eine Website, die ich auch bekommen kann, indem ich die Web-Protokolldateien für meinen Plone-Server parse. Werkzeuge wie *Analog*, *Webalizer*, *WebTrends* usw. nehmen Ihnen gerne die Arbeit ab, Ihre Plone- oder Apache-Protokolldateien zu parsen. Auch hierbei gilt, dass Sie den gesamten Code zu diesem Projekt im Kollektiv unter *http://sf.net/projects/collective* im Paket PloneStats finden.

12.2.1 Das Werkzeug starten

Das Werkzeug sollten Sie auf die gleiche Weise in ein Produktverzeichnis setzen, wie Sie es beim Inhaltstyp gemacht haben, d.h., indem Sie ein Verzeichnis innerhalb des Produktverzeichnisses der Instanz erstellen. In diesen Ordner fügen Sie die Dateien `refresh.txt`, `install.txt`, `readme.txt` und `__init__.py` hinzu.

In diesem Verzeichnis lautet der Name des Hauptmoduls `stats.py`. Es enthält den gesamten Code zur Erstellung der Statistiken. Auch hier behandle ich das Aussehen des Moduls, ohne zusätzlichen Zope-Code zu betrachten. Da Sie es aber direkt mit den anderen Plone-Werkzeugen koppeln, macht es allerdings wenig Sinn, es außerhalb von Zope zu benutzen.

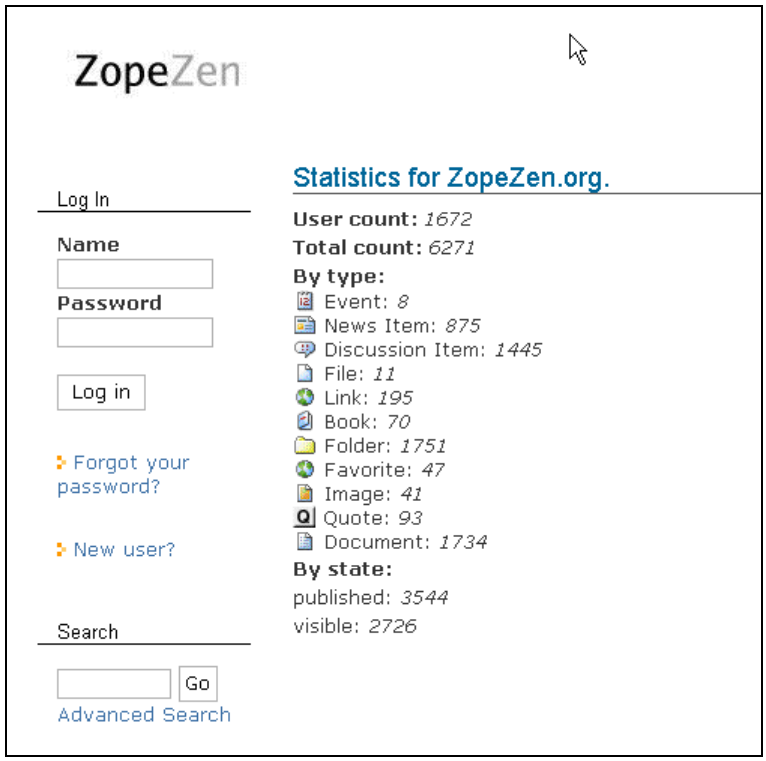


Abbildung 12.2: PloneStats auf ZopeZen

Listing 12.13 zeigt den Anfang des Werkzeugs. Dies ist eine einfache Version, die über zwei Methoden verfügt: eine, die die Anzahl der Inhaltstypen nach Typ und Workflow-Zustand zurückgibt, und eine andere für Benutzer, die die Gesamtzahl der Benutzer der Site zurückgibt.

Listing 12.13: Das grundlegende Statistik-Objekt

```
class Stats:
    def getContentTypes(self):
        """ Returns the number of documents by type """
        pc = getToolByName(self, "portal_catalog")
        # call the catalog and loop through the records
        results = pc()
        numbers = {"total":len(results),"bytype":{},"bystate":{}}
        for result in results:
            # set the number for the type
            ctype = str(result.Type)
            num = numbers["bytype"].get(ctype, 0)
            num += 1
```

```
            numbers["bytype"][ctype] = num

            # set the number for the state
            state = str(result.review_state)
            num = numbers["bystate"].get(state, 0)
            num += 1
            numbers["bystate"][state] = num
        return numbers

    def getUserCount(self):
        """ The number of users """
        pm = getToolByName(self, "portal_membership")
        count = len(pm.listMemberIds())
        return count
```

12.2.2 Das Paket in ein Werkzeug umwandeln

Um das Paket in ein Werkzeug umzuwandeln, müssen Sie den gleichen Prozess wie beim Inhaltstyp durchmachen. Mit anderen Worten, Sie müssen das Werkzeug im Modul `__init__.py` registrieren. Genau wie beim Beispiel mit dem Inhaltstyp erstellen Sie eine Datei namens `config.py`, die alle Konfigurationen enthält. Diese Datei sieht wie folgt aus:

```
from Products.CMFCore import CMFCorePermissions

view_permission = CMFCorePermissions.ManagePortal

product_name = "PloneStats"
unique_id = "plone_stats"
```

Die Sicherheit bei diesem Produkt ist einfacher, was daran liegt, dass das Produkt selbst recht einfach ist. Es interagiert lediglich mit anderen Werkzeugen und produziert einige Statistiken. Es gibt nichts, was Benutzer hinzufügen, bearbeiten oder löschen könnten, oder etwas, mit dem sie sonstwie interagieren könnten. Das heißt, Sie haben es nur mit einem einzigen Recht, *ManagePortal*, zu tun, dem Recht nämlich, die Konfiguration von Plone zu verwalten, das normalerweise nur an Manager vergeben wird. Also können nur Manager ins ZMI gehen und die Information sehen, die das Werkzeug bietet. Wenn Sie wollten, könnten Sie recht einfach eine hübsch ausschauende Skin für das Plone-Control Panel oder ein Portlet hinzufügen, das diese Information in Ihrer Site darstellt.

Was `__init__.py` anbelangt, fügen Sie nun den Initialisierungscode für das Werkzeug hinzu. Es gibt ein besonderes Initialisierungsscript für Werkzeuge namens `ToolInit`. In diesem Werkzeug sieht die Datei `__init__.py` wie folgt aus:

```
from Products.CMFCore import utils
from stats import Stats
from config import product_name

tools = (stats.Stats,)

def initialize(context):
    init = utils.ToolInit( product_name,
                    tools = tools,
                    product_name = product_name,
                    icon='tool.gif'
                    )
init.initialize(context)
```

Die Funktion `ToolInit` kann mehrere Werkzeuge annehmen. In diesem Fall haben Sie es aber nur mit einem zu tun. Bei mehreren Werkzeugen können Sie nur einen Produktnamen und ein Produkt-Icon haben, das im ZMI angezeigt wird. Das ist alles, was man braucht, um das Werkzeug zu registrieren. Nun müssen Sie das Hauptmodul vervollständigen, um es in ein echtes Werkzeugobjekt zu verwandeln.

12.2.3 Den Werkzeug-Code ändern

Als Nächstes fügen Sie den Code zu der Klasse hinzu, um ihn in ein Werkzeug umzuwandeln. Wie beim Inhaltstyp besteht dieser Schritt nur aus dem Hinzufügen der Sicherheit durch Ableitung von den korrekten Basisklassen, z.B. so:

```
from Globals import InitializeClass
from OFS.SimpleItem import SimpleItem
from AccessControl import ClassSecurityInfo

from Products.CMFCore.utils import UniqueObject, getToolByName
```

Die Klasse `SimpleItem` ist die vorgegebene Basisklasse für ein einfaches Objekt in Zope (nicht für einen Ordner). Tatsächlich erben alle Inhaltstypen von einer Klasse, die irgendwo in der Klassenhierarchie von `SimpleItem` erbt. Es ist nur so, dass Sie die ganzen zusätzlichen Attribute dieser anderen Klassen nicht benötigen. `UniqueObject` garantiert, dass es genau eine Instanz dieses Objekts in Ihrer Plone-Site gibt und dass sie nicht umbenannt oder verschoben werden kann. Das heißt, Ihr Objekt wird immer verfügbar sein.

Als Nächstes importieren Sie wie üblich die Variablen aus der Konfigurationsdatei. Durch die Zuweisung an die ID Ihres Objekts garantieren Sie, dass das Werkzeug die ID dessen haben wird, was immer `unique_id` in der Konfigurationsdatei ist – in diesem Fall `plone_stats`. Die zwei Basisklassen für das Werkzeug

sind `UniqueObject` und `SimpleItem`, die das Minimum dessen darstellen, was es benötigt. Beispiel:

```
from config import view_permission, product_name, unique_id

class Stats(UniqueObject,  SimpleItem):
    """ Prints out statistics for a Plone site """
    meta_type = product_name
    id = unique_id
```

Dann müssen Sie die Sicherheit einrichten, wobei Sie wiederum die Klasse `ClassSecurityInfo` benutzen werden, um explizit Rechte an den Methoden zu setzen. Beispiel:

```
security = ClassSecurityInfo()
security.declareProtected(view_permission, 'getContentTypes')
def getContentTypes(self):
    ...
```

12.3 Einige Elemente zur Benutzerschnittstelle hinzufügen

Der Hauptcode ist vollständig, also wäre es hübsch, dem Benutzer eine Antwort anzuzeigen, wenn Sie im ZMI auf das Werkzeug klicken, etwa in Form eines Beispiels dafür, wie das Produkt verwendet wird. Dazu werden Sie das ZMI so abändern, dass Sie etwas darstellen können.

Konkret heißt das, dass Sie ein Page Template schreiben, das tut, was Sie von ihm wollen. In diesem Beispiel ist es ein einfaches Page Template, das sich ins ZMI einklinkt. Das ZMI ist eine anspruchslose Benutzerschnittstelle, die lediglich Webseiten für den Benutzer ausspuckt, d.h., die Seite wird nicht durch aufwendige Makros oder Slots erstellt. Sie müssen nur ein wenig HTML schreiben und Folgendes hinzufügen:

```
<span tal:replace="structure here/manage_tabs" />
```

Diese eine `tal:replace`-Funktion bekommt die MANAGEMENT-Reiter und sorgt dafür, dass sie oben auf der Seite erscheinen. Meine ZMI-Seite iteriert über die zwei Methoden des Werkzeugs `plone_stats` und spuckt die Ergebnisse für den Benutzer aus, wie es in Listing 12.14 zu sehen ist.

Listing 12.14: Eine Seite zur Anzeige in der Management-Schnittstelle

```
<html>
<body>
<span tal:replace="structure here/manage_tabs" />

<p>Statistics for this Plone site.</p>

<h3>Content Types</h3>
<span tal:define="numbers here/getContentTypes">
    <p>
        Total count: <i tal:replace="numbers/total" /><br />
        Content types by type:
    </p>

    <span tal:repeat="type python:numbers['bytype'].keys()">
        <ul>
            <li>
              <span tal:replace="type" />:
              <i tal:replace="python: numbers['bytype'][type]" />
            </li>
        </ul>
    </span>

    <p>Content types by state:</p>
    <span tal:repeat="type python:numbers['bystate'].keys()">
        <ul>
            <li>
              <span tal:replace="type" />:
              <i tal:replace="python: numbers['bystate'][type]" />
            </li>
        </ul>
    </span>
</span>

<h3>Users</h3>
<p>
  User count: <i tal:replace="here/getUserCount" />
</p>

</body>
</html>
```

Sie heißt `output.pt` und wird ins Verzeichnis `www` platziert. Sie müssen kein separates Verzeichnis benutzen, aber wenn Sie es tun, ist es leichter zu merken.

Der letzte Schritt besteht darin, dieses Page Template für Ihr Produkt ins ZMI einzuklinken. Das machen Sie dadurch, dass Sie zur Klasse `Stats` zurückkehren und Folgendes hinzufügen (zuerst importieren Sie die Klasse `PageTemplateFile`, die mit dem Template aus dem Dateisystem umgehen kann):

```
from Products.PageTemplates.PageTemplateFile import PageTemplateFile
```

Dann registrieren Sie das Page Template als Methode für das Produkt, auf das zugegriffen werden kann. Im Folgenden kann die Methode `outputPage` nun über das Web aufgerufen werden, und es wird das entsprechende Page Template zurückgegeben:

```
outputPage = PageTemplateFile('www/output.pt', globals())
security.declareProtected(view_permission, 'outputPage')
```

Und schließlich werden die Reiter oben im ZMI von einem Tupel namens `manage_options` bestimmt, das eine Liste aller Reiter enthält, die auf einer Seite angezeigt werden sollen. Dort müssen Sie die neue Management-Seite einfügen, was Sie wie folgt tun:

```
manage_options = (
    {'label':'output', 'action':'outputPage'},
    ) + SimpleItem.manage_options
```

12.3.1 Das Werkzeug testen

Nun ist das Werkzeug fertig, d.h., Sie können testen, ob es funktioniert. Zuerst starten Sie Ihre Plone-Instanz neu, damit sie das Produktverzeichnis einliest und Ihr neues Werkzeug registriert. Als Zweites gehen Sie ins ZMI und dort in die obere rechte Ecke zum Dropdown-Menü namens ADD. Sie werden feststellen, dass in der Liste nun PloneStats aufgeführt ist. Wählen Sie diese Option, und klicken Sie auf ADD. Das nächste Formular listet die verfügbaren Werkzeuge im Produkt PloneStats auf, in diesem Fall erscheint nur eines, wie in Abbildung 12.3 zu sehen ist.

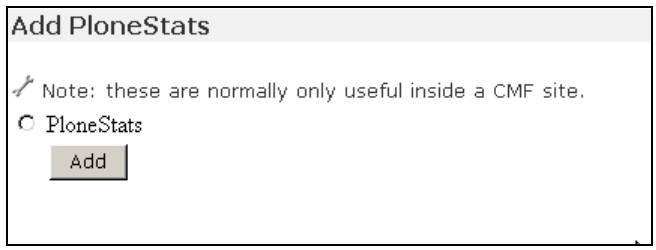


Abbildung 12.3: Hinzufügen des Werkzeugs

Wählen Sie das Werkzeug, und klicken Sie auf ADD. Klicken Sie dann auf das Werkzeug, um zu testen, ob es funktioniert. Sie sollten eine Reihe von Statistiken sehen, wie Sie sie zuvor für ZopeZen gesehen haben.

Dieses Werkzeug ist einfach, weil ich nicht wirklich weiß, welche Art von Darstellung die Leute gerne hätten. Wenn ich ein standardisiertes Berichtwerkzeug erstelle, dann können Sie es benutzen, wie Sie wollen. Einige Ideen, die einem dafür einfallen, sind z.B. eine Seite im Control Panel, ein kleiner Portlet-Kasten, eine PDF-Datei (Portable Document Format) mit hübschen Grafiken, die via E-Mail an einen Manager verschickt wird, oder eine API, damit externe Berichtswerkzeuge wie *Crystal Reports* mein Werkzeug benutzen können. An dieser Stelle warte ich ab und beobachte, was in der Zukunft alles passiert.

13 Entwickeln mit Archetypes

Archetypes ist ein Framework, um die Entwicklung von Plone-Produkten zu automatisieren. Wenn einmal eine formale Beschreibung für einen Inhaltstyp vorliegt, erledigt Archetypes praktisch alles, einschließlich der Erstellung von Views und Edit-Formularen für den Entwickler. Dies erlaubt Ihnen, neue Inhaltstypen schnell und mit einer minimalen Codegröße zu entwickeln. Weniger Code bedeutet weniger Fehlerquellen, weniger Code, den Sie verwalten müssen, wenn Plone sich ändert, schnellere Entwicklungszyklen und generell weniger Kosten.

Da das gesamte Produkt auf der Objektbeschreibung (Schema) basiert, ermöglicht es Ihnen, Tools zur automatischen Codegenerierung zu verwenden. Beispielsweise können Sie mit ArchGenXML, das später in diesem Kapitel behandelt wird, aus einem UML-Modell (Unified Modeling Language), das Sie mit einem UML-Designer Ihrer Wahl erstellt haben, ein lauffähiges Plone-Produkt generieren. Sie haben am Schluss ein lauffähiges Plone-Produkt vorliegen, ohne eine Zeile Code programmiert zu haben. Wenn Sie das Gefühl gehabt haben, dass sich das Schreiben von Produkten in Plone im Kapitel 12 ein wenig nach harter Arbeit anfühlt, wird dieses Kapitel eine willkommene Erleichterung bringen.

Dies bedeutet nicht notwendigerweise, dass Archetypes für jede Art von Produkten das Mittel der Wahl ist; manchmal ist Archetypes für den Zweck nicht optimal. Zum Beispiel hatte in einem Fall mein Inhaltstyp nur ein Feld, aber 16 unterschiedliche Permutationen auf den Daten des Feldes zu präsentieren, was bedeutete, dass nur wenig vom Archetypes-Framework benutzt wurde. Zugegeben, das war ein extremer Fall. Auch ist Archetypes nicht primär gedacht, um Low-level-Programmieraufgaben abzudecken. Aber in den allermeisten Fällen werden Sie sehen, dass Archetypes genau das ist, was Sie brauchen.

Einige Menschen beschweren sich, dass Archetypes das Leben für den Programmierer zu leicht mache und dass es dadurch schwer sei, Geld für eine Arbeit zu verlangen, die nur zehn Minuten dauert. Persönlich hatte ich nie ein Problem damit, denn Archetypes hat mir oft aus einer Sackgasse geholfen, wenn sich plötzlich die Projektanforderung von vier auf vierzehn Inhaltstypen änderte. Hier macht sich die Flexibilität der Archetypes bezahlt.

Eine Anekdote, von der ich einmal gehört habe, betrifft eine Webentwicklungsfirma. Wenn diese Firma Kundenbesuche macht, nehmen sie einen Programmierer mit. Während der Kunde seine Anforderungen vorträgt, tobt sich der Programmierer im Archetypes-Framework aus, und am Ende der Besprechung steht ein schneller Prototyp zur Demonstration zur Verfügung.

Insgesamt gesehen haben die meisten Plone-Entwickler Archetypes als den Weg zur Produktentwicklung akzeptiert, wodurch sich bereits eine große Wissensbasis entwickelt hat und Archetypes sich als Standard in der Plone-Entwicklung etabliert hat. Einige der Schlüsseleigenschaften von Archetypes sind:

- Archetypes erzeugt automatisch Seiten zum Ansehen und Editieren, so dass man zu einem gewissen Ausmaß ohne Schreiben von Template-Code auskommt.
- Für jedes Objekt wird automatisch eine systemweit eindeutige ID verwaltet, die der Benutzer nicht ändern kann. Dies bedeutet, dass Objekte wiedergefunden werden können, auch nachdem sie innerhalb des Systems verschoben worden sind. Diese UIDs kommen auch bei Referenzen zwischen Objekten zum Einsatz.
- Es erzeugt Referenzen zwischen Objekten. Jedes Objekt kann beliebige Referenzen zu anderen Objekten haben. Zum Beispiel können Sie eine beliebige Anzahl von Linkobjekten an ein News-Item hängen (natürlich nur, wenn dieses mit Archetypes entwickelt wurde).
- Es besitzt standardmäßige Sicherheitseinstellungen. Die gesamte Sicherheitsarbeit wird für Sie erledigt. Wenn Sie also mit den Standardeinstellungen zufrieden sind (und das ist meistens der Fall), brauchen Sie dahingehend nichts manuell entwickeln.
- Es bietet optional alternative Speichermöglichkeiten für Attribute. So können Sie zum Beispiel bestimmte Attribute in eine relationale Datenbank speichern statt in die ZODB.
- Es steht ein Framework zur transparenten Transformation von Feldinhalten zur Verfügung, zum Beispiel eine automatische Umwandlung von Word-Dokumenten in HTML.

Von seinem prinzipiellen Aufbau her ist Archetypes nicht notwendigerweise Plone-spezifisch, es könnte also zum Beispiel direkt in Zusammenhang mit dem Content Management Framework (CMF) verwendet werden. Jedoch kommt es derzeit nur in der Plone-Entwicklung zum Einsatz. Allerdings bringt die schemabasierende Entwicklung eine gewisse Zukunftssicherung mit sich, da in mittlerer Zukunft Plone auf Zope 3 portiert werden wird, und wenn das Archetypes-Programmiermodell dort verfügbar ist, wird eine Anpassung von Plone-Produkten wesentlich einfacher sein.

In diesem Kapitel werde ich die Erstellung von neuen Inhaltstypen mit Archetypes behandeln. Dieses Kapitel bringt tatsächlich all die Informationen, die Sie in den letzten Kapiteln gesammelt haben, in einen Zusammenhang und erläutert schnell einige grundlegende Konzepte. Nach einer Anleitung zum Installieren von Archetypes führe ich Sie in die Erstellung eines einfachen Inhaltstyps ein.

13.1 Einführung in Archetypes

Archetypes wird mit den Installationspaketen von Plone geliefert, daher ist es wahrscheinlich, dass Archetypes bereits in Ihrer Plone-Installation enthalten ist. Wenn Sie sich nicht sicher sind, ob das der Fall ist, sollten Sie überprüfen, ob Archetypes in der Produktliste im ZMI im Control Panel erscheint. In den Codebeispielen habe ich Archetypes 1.2.5 verwendet, da diese Version mit Plone 2.0 geliefert wird. Mittlerweile ist Archetypes in der Version 1.3.x erhältlich, die einige Erweiterungen bietet, auf die ich an entsprechender Stelle hinweisen möchte.

Um Archetypes zu installieren, gehen Sie zur Website *http://sf.net/projects/archetypes* und holen sich von dort die entsprechende Release von Archetypes. In meinen Beispielen verwendete ich `archetypes-1.2.5-rc4.tgz`.

```
$ tar -zxf archetypes-1.2.5-rc4.tgz
$ cd archetypes-1.2.5-rc4l
```

An diesem Punkt werden Sie entscheiden, was zu installieren ist. Das Minimum, das zu installieren ist, sind das `Archetypes`-Verzeichnis sowie `generator`, `validation` und `PortalTransforms`. Ab Archetypes 1.3 kommt hier noch `MimetypesRegistry` hinzu. Zur Installation schieben Sie diese in das `Products`-Verzeichnis Ihrer Plone-Installation. In meinem Fall ist dies `/var/zope/`, also führe ich folgende Befehle aus:

```
$ mv Archetypes /var/zope/Products
$ mv generator /var/zope/Products
$ mv validation /var/zope/Products
$ mv PortalTransforms /var/zope/Products
$ mv MimetypesRegistry /var/zope/Products
```

ArchExample und ArchGenXML sind beide optional, und Sie werden sie nicht unbedingt für die Funktion von Plone benötigen. Sie werden aber beide in diesem Kapitel behandelt, so empfiehlt es sich hier, diese zu installieren. In Archetypes 1.3 sind diese Produkte nicht im Basispaket enthalten und müssen von *http://sf.net/projects/archetypes* heruntergeladen werden.

Um ArchExample zu installieren, verschieben Sie ArchExample wie folgt in das `Products`-Verzeichnis Ihrer Plone-Installation:

```
$ cd ..
$ mv ArchExample /var/zope/Products
```

Wenn Sie ArchGenXML einsetzen wollen, können Sie es an einem beliebigen Ort installieren, von wo Sie es als Kommandozeilenbefehl aufrufen können. Ich installiere es normalerweise ebenfalls in mein `Products`-Verzeichnis, zusammen mit allen anderen Produkten. Dort stört es nicht, und ich finde es leicht, wenn ich es zum Generieren von Produkten benötige. Ich installiere es mit folgendem Befehl:

```
$ mv ArchGenXML /var/zope/Products
```

Wie in der ArchGenXML-Dokumentation erwähnt wird, braucht ArchGenXML das PyXML-Modul für Python. Wenn Sie den Windows- oder Mac-Installer für Plone benutzt haben, ist PyXML bereits installiert. Wenn nicht, können Sie es von *http://pyxml.sf.net* beziehen. In meinem Fall installierte ich die Version 0.8.3, deshalb führte ich nach dem Herunterladen folgende Befehle aus:

```
$ tar -xvf PyXML-0.8.3.tar.gz
$ cd PyXML-0.8.3
$ python setup.py install
```

Hinweis

Normalerweise benötigen Sie unter Unix Root-Rechte, um Python-Pakete wie PyXML ins globale Python-Verzeichnis zu installieren.

Nun da Sie alles installiert haben, möchte ich Ihnen einige Beispiele zeigen.

13.1.1 Einblick in Archetypes

Eine ganze Menge von großartigen Beispielen sind für Archetypes verfügbar, deshalb möchte ich, anstatt ein neues zu erfinden, ArchExample zeigen, das beim Archetypes-Repository dabei ist. Darin wird ein neuer Inhaltstyp namens *Article* eingeführt, anhand dessen die Stärken von Archetypes demonstriert werden.

`Article.py` enthält den Kern des Sourcecodes, und Sie werden sehen, dass der Code ziemlich anders aussieht als die früheren Codebeispiele. Eine wichtige Neuerung ist die Verwendung von Schemata. Der Begriff *Schema* ist hier angelehnt an den Schemabegriff aus der Datenbankentwicklung.

```
StringField("blurb",
            searchable = 1,
            widget = TextAreaWidget(),
            ),
```

Dieses Stück Code zeigt ein Attribut namens `blurb` innerhalb des Inhaltstyps, das den Datentyp `string` hat und das im Bearbeiten-Formular als TextArea angezeigt werden soll. Ich werde später auf alle Optionen für Felder und Kontrollelemente (Widgets) eingehen. Abbildung 13.1 zeigt den neuen Inhaltstyp *Article* im Einsatz beim Editieren.

Abbildung 13.1: Edit-Formular des neuen Inhaltstyps

Mit lediglich vier Codezeilen wurde zu unserem Inhaltstyp ein Feld hinzugefügt. Für jedes Standard-Formelement in Plone gibt es eine Felddefinition in Archetypes; alle »langweiligen« Aufgaben wie z.B. Feldvalidierung, Fehlerbehandlung, richtige Darstellung werden vom Archetypes-Framework für Sie erledigt. Um die Einfachheit zu demonstrieren, ändern Sie einfach die Beschriftung (Label) des Kontrollelements (Widget) auf *Article blurb* und deklarieren das Eingabefeld als Muss-Feld (required). Diese Änderungen erreichen Sie durch unten stehenden Code:

```
StringField("blurb",
            required = 1,
            searchable = 1,
            widget = TextAreaWidget(label="Article Blurb"),
            ),
```

Es wurde der Parameter `required = 1` hinzugefügt, der das Feld zu einem Muss-Feld macht, sowie ein `label`-Parameter, der an das Widget übergeben wird. Wenn Sie Plone neu starten und einen neuen Artikel hinzufügen, werden Sie Ihre Änderungen sehen. Das Feld *blurb* hat nun eine andere Beschriftung, und durch den roten Punkt beim Eingabefeld wird deutlich, dass es sich um ein Muss-Feld handelt.

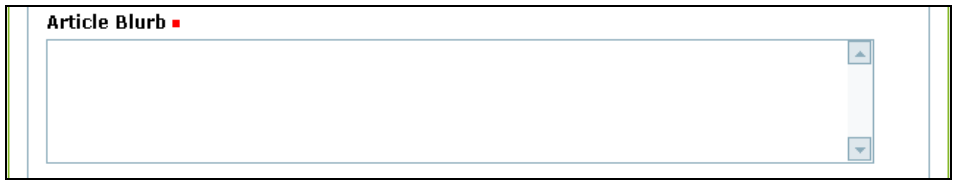


Abbildung 13.2: Das veränderte Eingabefeld

Dies ist nicht nur eine kosmetische Änderung, denn sie spiegelt die Modifikation im darunter liegenden Schema wider, und das ist die wirkliche Stärke von Archetypes. Wenn Sie dies mit dem Schreiben von Python-Produkten vergleichen, werden Sie sehen, dass viel von der Knochenarbeit des sich ständig wiederholenden Kodierens entfällt und im Hintergrund durch das Framework erledigt wird. Man kann es auch so formulieren: Wenn Sie für einen Inhaltstyp das Schema definieren können, ist das Erstellen der Software keine Hexerei mehr. Außerdem sind Änderungen an existierenden Inhaltstypen leicht durchzuführen, was einem bei den schnelllebigen Projekten der heutigen Zeit sehr zugute kommt.

Wenn Sie Änderungen am Code durchführen, um die folgenden Beispiele auszuprobieren, müssen Sie Plone neu starten, um jene wirksam zu machen. Mehr Informationen darüber gibt es später im Abschnitt »Entwickeln mit Archetypes«.

13.1.2 Schemata, Felder und Kontrollelemente

In Archetypes hat jeder Inhaltstyp, der im Prinzip eine registrierte Python-Klasse mit ganz bestimmten Basisklassen ist, ein Schema. Dieses Schema enthält Felddefinitionen enthält, die ihrerseits noch ihre Darstellung über Kontrollelemente (Widgets) definieren.

Abbildung 13.3 zeigt die Beziehung zwischen Schemata, Feldern und Kontrollelementen.

Schemata und das Basisschema

Um ein Schema zu erzeugen, werden die Felddefinitionen in einem Tupel (unveränderliche Liste in Python) an den Konstruktor des Schemata übergeben. Zum Beispiel hat das Article-Schema drei Felder: *group*, *blurb* und *body*. Der folgende Code stellt den Beginn der Schemadefinition dar:

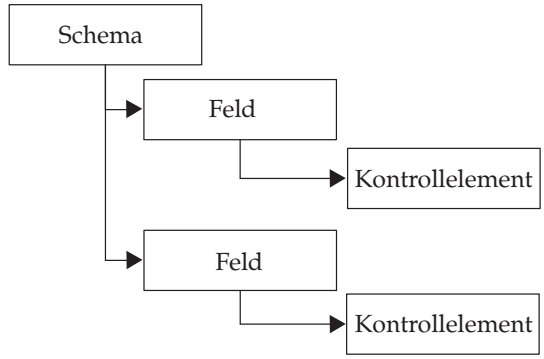


Abbildung 13.3: Schemata, Felder und Kontrollelemente

```
Schema((
    StringField('group',
        vocabulary=ARTICLE_GROUPS,
        widget=SelectionWidget(),
        ),
        # other fields here
        )
```

Es ist möglich, ein Schema aus anderen Schemata zusammenzusetzen. Das passiert auch hier, denn bei jeder Schemadefinition wird das Basisschema (BaseSchema) zum selbst definierten Schema hinzugefügt.

Das *BaseSchema* enthält die zwei Elemente, die bei jedem Inhaltstyp in Plone enthalten sind: einen Titel und eine ID (Kurzname), die beide den Konventionen des Plone-Namensschemata entsprechen. In ArchExample passiert dies durch Addieren von *BaseSchema* zum eigentlichen Schema, zum Beispiel:

```
schema = BaseSchema +  Schema((
    StringField('group',
                vocabulary=ARTICLE_GROUPS,
                widget=SelectionWidget(),
                ),
    ...
```

Die Einträge im Schema werden beim Abrufen der Schemadefinition, wenn das Objekt dargestellt wird, in der gleichen Reihenfolge retourniert wie bei der Schemadefinition. Dies bedeutet, dass Sie durch Verschieben der Felddefinitionen innerhalb des Schemata die Reihenfolge in der Darstellung beliebig einstellen können. Deswegen wurde das Basisschema auch am Beginn addiert, damit ID und Titel als erste Felder erscheinen.

Felder

Bis jetzt haben Sie das *StringField* gesehen, das ein allgemeines Feld für Textinhalte eines Inhaltstyps darstellt. Eine große Anzahl von Feldtypen ist in Archetypes verfügbar, wie Sie in Tabelle 13.1 sehen. Im Laufe der Zeit entstehen immer weitere Felddefinitionen, und wenn Sie es benötigen, können Sie sich Ihre eigenen Feldtypen dazudefinieren.

Jedes Feld hat ein Default-Kontrollelement, das benutzt wird, wenn Sie kein eigenes spezifizieren. Im Falle von *StringField* ist dies zum Beispiel *StringWidget*, ein einzeiliges Textfeld. Im vorigen Beispiel habe ich aber ein *TextAreaWidget* festgelegt, um eine mehrzeilige Eingabe zu bewirken. All diese Feld- und Widget-Definitionen werden vom Modul `Products.Archetypes.public` importiert, zum Beispiel:

```
from Products.Archetypes.public import BooleanField
```

Alle Felder werden auf dieselbe Art instanziiert – durch den Konstruktoraufruf an die Klasse `Field` mit mindestens dem zwingenden Parameter `name`. Sie können eine zusätzlich beliebige Anzahl von Schlüsselwortparametern übergeben, zum Beispiel:

```
from Products.Archetypes.public import IntegerField
# a simple field for age
age = IntegerField('age')
```

Name	Typ	Default-Widget	Beschreibung
`BooleanField`	Boolesche Werte	`ComputedWidget`	Einfache Speicherung von `True` oder `False` für ein Feld.
`DateTimeField`	Datums- und Zeitwerte	`CalendarWidget`	Speichert Datums- und Zeitwerte.
`FileField`	Dateien	`FileWidget`	Speicherung von größeren Datenpaketen wie etwa großen Textdateien, Word-Dateien oder anderen Binärdaten.
`FixedPointField`	Fixpunktzahlen	`DecimalWidget`	Zum Speichern von Zahlen mit fixer Kommastelle (z.B. Geldbeträge).
`FloatField`	Fließkomma	`DecimalWidget`	Speichern von Zahlen mit Fließkomma. Entspricht dem Datentyp `float` in Python.
`ImageField`	Bilddaten	`ImageWidget`	Speichert ein Bild und unterstützt automatische Skalierung.
`IntegerField`	Ganzzahl	`StringWidget`	Ganzzahlige Werte, entspricht dem Python-Typ `int`.
`LinesField`	Listen	`LinesWidget`	Speichert Listen von Strings.

Tabelle 13.1: In Archetypes verfügbare Felder

Name	Typ	Default-Widget	Beschreibung
`PhotoField`	Image	`PhotoWidget`	Dasselbe wie `ImageField`, aber mit mehr Skalierungsstufen.
`Reference-Field`	Referenzen	`ReferenceWidget`	Enthält Referenzen zu anderen Objekten.
`String-Field`	Zeichenketten (String)	`StringWidget`	Zeichenketten für einzeilige Texte.
`TextField`	String	`TextWidget`	Dasselbe wie `StringField`, aber für größere, mehrzeilige Texte. Der Text kann außerdem in andere Formate transformiert werden.
`Computed-Field`	Pseudofeld	`ComputedWidget`	Ein Pseudofeld, das nicht als Wert gespeichert wird, sondern aufgrund eines Ausdrucks (Expression) ausgewertet wird.

Tabelle 13.1: In Archetypes verfügbare Felder (Forts.)

Jeder Feldtyp hat Attribute, die einzelnen Formularfeldern zugewiesen werden können. Zwei dieser Attribute sind bereits vorgekommen: *name* und *widget*. *name* ist der einzige erforderliche Parameter und soll keine Umlaute, Leer- oder Sonderzeichen enthalten. Als Konvention hat sich etabliert, dass Feldnamen immer mit Kleinbuchstaben beginnen. Das Attribut *name* agiert als ID des Feldes und wird intern verwendet, vergleichbar mit dem *id*-Attribut von Zope-Objekten.

Name	Beschreibung	Mögliche Werte
`accessor`	Der Name der Zugriffsmethode auf ein Feld. Somit können Sie den Namen einer beliebigen Methode einsetzen, die den Wert dieses Feldes ermittelt (z.B. `specialGetMethod`).	Siehe »Überschreiben von Standardmethoden« später im Kapitel.
`default`	Der Defaultwert für das Feld, wird beim Instanziieren des Objekts gesetzt.	Der Wert sollte dem Datentyp des Feldes entsprechen.
`default_method`	Alternative zu `default`: Ein Stringwert, der den Namen der Methode enthält, die den `default`-Wert errechnet. Zum Beispiel kann man damit ein Datumsfeld mit dem aktuellen Datum vorbelegen.	Stringwert (zum Beispiel `getSpecialDescription`).
`Edit_accessor`	Ähnlich wie `accessor` gibt dieses Attribut den Namen einer Zugriffsfunktion an, jedoch gibt diese Funktion den Rohwert eines Feldes zurück, der im Gegensatz zum Accessor nicht transformiert wird.	Beliebiger Methodenname (zum Beispiel `rawGetMethod`).

Tabelle 13.2: Feldattribute

Name	Beschreibung	Mögliche Werte
`enforce-Vocabulary`	Wenn dieses Attribut gesetzt ist, kann ein Feld mit Werteliste (Vokabular) nur Werte annehmen, die im Vokabular enthalten sind.	`True` oder `False`.
`index`	Wenn Sie ein Feld in einem eigenen Katalogindex haben wollen, dann spezifizieren Sie den Indextyp hier als String. Wenn Sie `:schema` anhängen, wird dies auch als Metadatenspalte hinzugefügt.	Der Name eines Indextyps, zum Beispiel `KeywordIndex` oder `KeywordIndex:schema`.
`Name`	Eindeutiger Name des Feldes.	Stringwert; per Konvention sollte der Wert mit einem Kleinbuchstaben beginnen, ansonsten muss der Feldname den Python-Namensregeln für Variablen entsprechen (zum Beispiel `description`, `user_name` oder `coffee_bag_6`).
`Mode`	Zugriffsmodus für das Feld; standardmäßig ist ein Feld für Lesen und Schreiben (`rw`) eingestellt.	Für lesenden Zugriff: `r`, für schreibenden Zugriff: `w`, für Lesen und Schreiben: `rw`.
`multiValued`	Legt fest, ob ein Feld Mehrfachwerte enthalten kann, zum Beispiel bei Referenzfeldern.	`True` oder `False`.
`mutator`	Das Gegenstück zum Accessor. Dieses Attribut legt den Namen der Set-Methode fest. Wenn Sie nichts angeben, wird wie beim Accessor eine Standardfunktion dafür auto-generiert.	Methodenname (zum Beispiel `specialSetMethod`).
`primary`	Wenn dieses Attribut auf `True` gesetzt ist, dann wird dieses Feld zum Primärfeld, dessen Inhalt bei WebDAV- und FTP-Anfragen zurückgegeben wird (zum Beispiel das `body`-Feld bei Plone-Dokumenten). Pro Inhaltstyp kann es naturgemäß nur ein Primärfeld geben, und wenn mehrere angegeben werden, wird das erste angegebene Primärfeld eines Schemata genommen.	`True` oder `False`.
`required`	Legt fest, ob ein Feld eine Eingabe zwingend vorschreibt.	`True` oder `False`.

Tabelle 13.2: Feldattribute (Forts.)

Name	Beschreibung	Mögliche Werte
`schemata`	Innerhalb von Schemata kann man Felder wiederum gruppieren. Felder mit gleichem `schemata`-Namen werden zum Beispiel beim Edit-Formular eines Inhaltstyps (*base_edit*) zu Teilformularen gruppiert. Dies empfiehlt sich speziell bei Klassen mit sehr vielen Feldern.	Stringwert.
`metadata`	Legt fest, ob ein Feld ein Metadatenfeld ist (z.B. `Title`).	`True` oder `False`.
`searchable`	Legt fest, ob ein Feld für die Volltextsuche mitindiziert wird, indem dieses Feld bei `SearchableText` mit berücksichtigt wird.	`True` oder `False`.
`validators`	Die Validierer, die in Bezug auf dieses Feld ausgeführt werden. Beim Ändern des Feldinhaltes werden diese Validierer der Reihe nach aufgerufen.	Beliebiger Validierer; siehe den Abschnitt »Eingabevalidierung« später im Kapitel.
`vocabulary`	Eine Auswahlliste von Werten, aus denen der Benutzer den Wert für dieses Feld festlegen kann. Im Formular stehen diese Werte z.B. als Dropdown-Liste zur Verfügung.	Liste von Strings (z.B. `["Gruen", "Rot", "Blau"]`) oder Liste von Wertepaaren aus Beschriftung und Wert, z.B. `(("#00ff00", "Gruen"), ("#ff0000","Rot"), ("#0000ff","Blau")).`
`storage`	Legt fest, wie ein Wert abgespeichert werden soll; standardmäßig wird `AttributeStorage` verwendet, was bedeutet, dass Feldwerte als normale Python-Attribute gespeichert werden.	Jede beliebige Storage-Klasse, wie zum Beispiel `AttributeStorage` oder `SQLStorage`. Sie können eine vollständige Liste der Storage-Klassen dem Archetypes-API entnehmen. Später im Kapitel wird die Verwendung von `SQLStorage` näher erläutert.
`widget`	Das Kontrollelement, das für dieses Feld verwendet werden soll.	Konstruktoraufruf zu einer passenden Widget- Klasse (z.B. `TextAreaWidget()`).

Tabelle 13.2: Feldattribute (Forts.)

Nachdem die Felder und deren Attribute behandelt wurden, ist es Zeit, sich die einzelnen Kontrollelemente anzusehen.

Kontrollelemente

Ein Kontrollelement, auch Widget genannt, enthält die Information, wie ein Feld eines Objekts dargestellt werden soll. Die Auswahl eines Kontrollelements hängt natürlich mit dem Feldtyp zusammen, so kann ein *StringField* auf verschiedenste Arten dargestellt werden z.B. mit einem *StringWidget* oder einem *TextAreaWidget*. Archetypes enthält eine große Zahl von Kontrollelementen, die alle über `Products.Archetypes.public` importiert werden können. Zum Beispiel:

```
from Products.Archetypes.public import BooleanWidget
```

Alle Widgets werden auf dieselbe Art instanziiert – durch einen Konstruktoraufruf an die entsprechende Widget-Klasse mit optionalen Parametern. Zum Beispiel:

```
from Products.Archetypes.public import IntegerField
from Products.Archetypes.public import IntegerWidget
# a simple field for age
age = IntegerField('age',
       widget=IntegerWidget(label="Your age")
       )
```

Widgets können Extra-Attribute haben, die vom Typ des Widgets abhängen; zum Beispiel kann man bei einem *StringWidget* ein `size`-Attribut setzen, das dann im HTML-Code auch als `size`-Parameter erscheint. Um etwa ein Textfeld mit 20 Zeichen Breite zu definieren, schreiben Sie folgenden Code:

```
bankAccountNumber = StringField('bank',
   widget=StringWidget(
           label="Bank account number",
           size=20)
         )
```

Tabelle 13.3 beschreibt alle Widgets, die von Archetypes zur Verfügung gestellt werden.

Name	Beschreibung	Weitere Attribute
`BooleanWidget`	Zeigt eine Checkbox zur Eingabe von booleschen Werten an.	--
`CalendarWidget`	Dient zur Eingabe von Datumswerten. Zeigt Eingabefelder für Jahr, Monat, Tag und Uhrzeit sowie ein Popup-Fenster, um den Tag aus einem Kalender auszuwählen.	--

Tabelle 13.3: Die verfügbaren Widgets

Name	Beschreibung	Weitere Attribute
`ComputedWidget`	Das Widget für `ComputedField`. Es hat folglich kein Eingabefeld, sondern nur ein Anzeigefeld (sichtbar in `base_view`).	--
`DecimalWidget`	Das Widget für das `FixedPointField`.	`size`
`FileWidget`	Zeigt ein HTML-Element zum Upload von Dateien an; dient als Standard-Widget für das `FileField`.	--
`IdWidget`	Einfaches HTML-Eingabefeld für autogenerierte IDs.	--
`ImageWidget`	Dient zum Anzeigen und Editieren von Bildern.	Es gibt einen Parameter namens `display_threshold`, mit dem man angeben kann, welche Bildgröße (in Bytes angegeben) inline maximal angezeigt werden soll. Wenn die Bildgröße über diesem Wert liegt, wird das Bild nur als Link angezeigt.
`IntegerWidget`	Einfaches Eingabefeld für Ganzzahlen.	`size`.
`KeywordWidget`	Dieses Widget zeigt eine Liste von Schlüsselwörtern vom Katalog in einem komplexen Widget, wie Sie es vom EIGENSCHAFTEN-Reiter kennen, der bei jedem Plone-Objekt zur Verfügung steht.	--
`LabelWidgets`	Zeigt Beschriftungen (Labels) für Formelemente an. Dieses Widget editiert keine Werte.	--
`LinesWidget`	Dient zur Eingabe von `LinesField`s, also Listen von Strings.	`rows` und `columns`.
`MultiSelectionWidget`	Auswahlliste; standardmäßig ist es ein HTML-select-Element (Listbox), bei dem mehrere Werte gleichzeitig ausgewählt werden können.	`format`, kann entweder `select` oder `checkbox` sein.
`PasswordWidget`	HTML-Element zum Eingeben von Passwörtern.	--

Tabelle 13.3: Die verfügbaren Widgets (Forts.)

Name	Beschreibung	Weitere Attribute
`RichWidget`	Kontrollelement, das den vom System eingestellten WYSIWYG- Texteditor (Epoz oder Kupu) als RichText-Editor einsetzt. Erlaubt die Eingabe in verschiedenen Formaten, die automatisch transformiert werden.	Mögliche Werte sind `format`, `rows`, `mode` und `cols`.
`ReferenceWidget`	Auswahlelement, um Referenzen zu anderen Objekten zu editieren.	--
`SelectionWidget`	HTML-Auswahlliste. Wenn der Modus `flex` ist (Standardwert), wird bei mehr als 4 Auswahloptionen eine HTML-Dropdown-Liste angezeigt, bei kleinerer Zahl eine Liste mit Radiobuttons.	`format` kann einer von den folgenden Werten sein: `flex`, `select` oder `radio`.
`StringWidget`	Eingabeelement für einzeilige Textfelder.	`size` und `maxlength`.
`TextAreaWidget`	Mehrzeiliges Texteingabefeld, das auch das Heraufladen von Content in verschiedenen Formaten ermöglicht.	

Tabelle 13.3: Die verfügbaren Widgets (Forts.)

Für jedes der Widgets, die in Tabelle 13.3 aufgelistet sind, beschreibt Tabelle 13.4 die Attribute, die für alle Widgets gelten. Zum Beispiel haben Sie schon das Attribut `label` gesehen, das die Beschriftung Ihres Widgets festlegt. In Verbindung mit den extra Attributen jedes Widgets haben Sie das komplette Set von Widget-Attributen.

Name	Beschreibung	Mögliche Werte
`label`	Legt die Beschriftung fest, die zusammen mit diesem Feld im Benutzerinterface erscheint.	Beliebiger Text, z.B. `Beginn` für ein Feld mit Namen `start_date`.
`modes`	Anzeigemodi des Feldes, es gibt hier zwei Modi: `view` (Ansicht) und `edit` (Bearbeiten).	Liste von Strings; Standardwert: `("view", "edit")`.

Tabelle 13.4: Attribute der Widgets

Name	Beschreibung	Mögliche Werte
`populate`	Gibt an, ob ein Widget mit dem Wert des Feldes befüllt werden soll. Normalerweise ist dies der Fall, aber in speziellen Fällen wie zum Beispiel bei Passwortfeldern ist es nicht erwünscht, den Feldinhalt angezeigt zu bekommen. Standardmäßig ist der Wert auf `True` gesetzt.	`True` oder `False`.
`postback`	Wenn beim Ausfüllen des Edit-Formulars ein Fehler auftritt, d.h. eine Validierung fehlschlägt, und man nach der Eingabe wieder auf das Formular geleitet wird, um die Eingabe zu korrigieren, legt dieses Attribut fest, ob ein Widget mit dem bereits eingegebenen Wert wieder befüllt werden soll. Standardmäßig ist dieses Attribut auf `True` gesetzt, aber zum Beispiel bei Passwortfeldern ist dies nicht erwünscht.	`True` oder `False`.
`visible`	Legt die Sichtbarkeit eines Feldes fest. Es ist ein Python-Dictionary, das zu dem jeweiligen Anzeigemodus die Sichtbarkeit als String angibt. Mögliche Werte sind `visible`, `hidden` (als verstecktes HTML-Element angezeigt) und `invisible` (überhaupt nicht dargestellt).	Zum Beispiel bedeutet `{'view': 'visible', 'edit': 'hidden'}`, dass das Feld zwar in der Standardansicht (base_view) angezeigt wird, aber in der Bearbeitungsansicht (base_edit) versteckt wird.

Tabelle 13.4: Attribute der Widgets (Forts.)

Einige Beispiele für Feld- und Widget-Kombinationen

Dieser Abschnitt zeigt einige nützliche Kombinationen, die oft verwendet werden und als gute Beispiele dienen. In diesem Beispiel soll eine Auswahlliste von Früchten für ein Feld namens *fruit* zur Verfügung gestellt werden. Als Feldtyp wird hier `StringField` mit einem eigenen Vokabular genommen und als Widget das `SelectionWidget`, das für das Darstellen der Auswahlliste verantwortlich ist:

```
StringField('fruit'
    vocabulary = ["Apple", "Orange", "Mango", "Banana"],
    widget = SelectionWidget(label = "Favourite Fruit")
    )
```

Das `ImageField` ist sehr nützlich, um einen Inhaltstyp mit Bildern zu versehen. Hier ein einfaches Beispiel für ein `ImageField`:

```
ImageField('portrait',
    widget = ImageWidget(label = "My picture"),
    )
```

Das folgende Beispiel zeigt einen etwas komplizierteren Inhaltstyp. Die meisten Inhaltstypen haben ein Primärfeld, das Daten enthalten kann, wie zum Beispiel der Inhaltstyp `Document`, wo `body` das Primärfeld ist. Dieses Body-Feld ist der Haupttext des Inhaltstyps. Um dies zu erreichen, muss man nur wenige Attribute setzen.

Zuerst sollte dieses Feld volltextindiziert sein, was durch das Setzen des `searchable`-Attributs erreicht wird. Außerdem sollte dieses Feld bei FTP- und WebDAV-Anfragen zurückgegeben werden, deshalb setzen Sie hier das Attribut `primary` auf `1`. Lesen Sie mehr dazu im Exkurs »Das Primary Field: Marshalling und Behandeln von FTP- und WebDAV-Anfragen«. Wenn dieses Feld verschiedene Inhaltstypen akzeptieren soll, können Sie diese mit dem Attribut `allowable_content_types` einstellen. (Zur Information: `allowable_content_types` bezieht sich hier nicht auf Plone-Inhaltstypen, sondern auf mögliche MIME-Types des Textfeldes.) Durch die Verwendung von `RichWidget` wird außerdem das Editieren des Textes mit dem im System eingestellten WYSIWYG-Editor (Epoz oder Kupu) unterstützt.

```
TextField('body'
          searchable = 1,
          primary = 1,
          default_output_types = 'text/html',
          allowable_content_types = ('text/plain',
                                     'text/structured',
                                     'text/restructured',
                                     'text/html'),
          widget = RichWidget(label = 'Body'),
          )
```

Exkurs: Das Primary Field: Marshaling und Behandeln von FTP- und WebDAV-Anfragen

Plone ist ein objektorientiertes System, in dem ein Objekt mehrere Attribute hat und nicht einfach als flache Datei repräsentiert werden kann. Aber die meisten Transportprotokolle, wie etwa FTP oder WebDAV, behandeln den Inhalt leider genau so.

Deswegen benötigen wir eine Übersetzung zwischen diesen Welten, und der `PrimaryFieldMarshaller`, der das Objekt auf sein primäres Feld reduziert, ist eine der Möglichkeiten. Durch Festlegen eines Primärfeldes für eine Objektklasse wird dieses Feld zu dem, was anstatt des gesamten Objekts von diesen Protokollen zurückgegeben wird.

Dies ist eine pragmatische Lösung für ein kompliziertes Problem und als solche natürlich nicht perfekt, da die Metadaten verloren gehen. Wenn Sie schon einmal Plone-Dokumente über FTP oder mit dem External Editor geöffnet haben, werden Sie gesehen haben, dass er die Metadaten als durch Doppelpunkt getrennte Schlüssel/Wert-Paare am Beginn des Textdokuments platziert. Dies ist ein weiterer Ansatz, um dieses Problem zu lösen.

Deswegen erlaubt Archetypes, pro Klasse einen so genannten *Marshaller* festzulegen, der die Übersetzung eines Objekts in eine flache Datei und umgekehrt durchführt. Archetypes enthält zwei Marshaller, nämlich `PrimaryFieldMarshaller`, der nur das Primärfeld eines Schemata behandelt, und den `RFC822Marshaller`, der, wie gerade beschrieben, die Metadaten in den exportierten Text mit hineinpackt. Sie können sich aber selbstverständlich für eigene Bedürfnisse einen eigenen Marshaller schreiben, um z.B. Objekte als XML zu exportieren. Um einen bestimmten Marshaller für eine Klasse festzulegen, muss man in der Schemadefinition folgende Zeile hinzufügen: `marshall=irgendeinMarshaller()`, im Falle des `PrimaryFieldMarshaller` also: `marshall=PrimaryFieldMarshaller()`

Validieren von Eingaben

Obwohl die automatisch generierten Formulare von Archetypes das Editieren von Objekten korrekt behandeln und auch Fehler wie etwa das Nichtausfüllen von zwingenden Feldern abfangen, wird manchmal eine detailliertere Validierung von Feldeingaben benötigt. Um das zu erreichen, kann man pro Feld eine Kette von Validierern festlegen, die vor dem Speichern des Wertes der Reihe nach aufgerufen werden.

Um dies zu erreichen, ordnen Sie dem Feld den Validierer `IsInt` mit dem `validators`-Parameter zu, wie das folgende Beispiel zeigt:

```
from Products.Archetypes.public import IntegerField
from Products.Archetypes.public import IntegerWidget
# a simple field for age
age = IntegerField('age',
      validators=("isInt"),
```

```
    widget=IntegerWidget(label="Your age")
    )
```

Woher kommt `IsInt`? `IsInt` ist der Name des Validierers, der im `validation`-Framework registriert ist; in diesem Paket ist ein Satz von vordefinierten Validierern enthalten, die in Tabelle 13.5 aufgelistet sind. Um mit den genaueren Details vertraut zu werden, empfiehlt es sich, den Sourcecode des Pakets in `Products/validation/validators/__init__.py` zu studieren. Einige der Validierer sind spezifisch für die USA, so dass man sie im europäischen Markt nur bedingt einsetzen kann, wie zum Beispiel die Überprüfung, ob ein String eine gültige US-Sozialversicherungsnummer ist.

Name	Beschreibung
`isDecimal`	Überprüft, ob der eingegebene Text einer Dezimalzahl entspricht. Beinhaltet positive und negative Zahlenliterale sowie die Exponentialnotation.
`isInt`	Überprüft, ob der Text eine Ganzzahl ist.
`isPrintable`	Überprüft, ob ein Text druckbar ist, also nur aus Buchstaben, Ziffern und dem Dollarzeichen besteht.
`isSSN`	Prüft, ob ein Text eine gültige US-amerikanische Sozialversicherungsnummer ist.
`isUSPhoneNumber`	Prüft, ob ein Text eine gültige US-amerikanische Telefonnummer ist.
`isInternational-PhoneNumber`	Prüft, ob ein Text eine gültige internationale Telefonnummer ist.
`isZipCode`	Prüft, ob ein Text eine gültige US-Postleitzahl mit 5 oder 9 Zeichen ist.
`isURL`	Prüft, ob ein Text mit `http://`, `ftp://` oder `https://` beginnt.
`isEmail`	Prüft, ob ein Text eine korrekte E-Mail-Adresse ist.

Tabelle 13.5: Verfügbare Validierer

Es ist auch möglich und nicht sehr kompliziert, eigene Validierer zu schreiben und zu registrieren. Ein Validierer ist eine einfache Klasse, die das `IValidator`-Interface implementiert, einen einfachen Konstruktor hat und eine `__call__`-Methode aufweist. Es gibt schon zwei vorbereitete Validiererklassen, die nur noch mit den richtigen Parametern instanziiert und registriert werden müssen: `RegexValidator` und `RangeValidator`. Um zum Beispiel einen Validierer zu definieren, der Altersangaben auf einen Wertebereich von 0 bis 150 Jahren verifiziert, müssen folgende Zeilen am Beginn Ihres Python-Moduls hinzugefügt werden:

```
from validation.validators.validator import RangeValidator
from validation import validation
```

```
# the RangeValidator takes a name for the validator
# and a start and end value
validAge = RangeValidator("validAge", 0, 150)
validation.register(validAge)
```

Anschließend muss noch der Validierer dem Feld zugeordnet werden:

```
validators=("isInt","validAge"),
```

Zuerst überprüft der Code, dass Sie eine gültige Ganzzahl eingegeben haben, dann wird verifiziert, dass die Altersangabe plausibel ist. Wenn Sie einen gänzlich neuen Validierer schreiben wollen, müssen Sie eine Validiererklasse schreiben und eine Instanz davon im Validierersystem registrieren. In diesem Beispiel wird eine Validiererklasse geschrieben, die verifiziert, ob der Datumswert zwischen 2 angegebenen Zeitpunkten liegt. Mit diesem Validierer könnte man zum Beispiel prüfen, ob ein Urlaub innerhalb der Schulferien liegt. Dazu muss gesagt werden, dass der `IsInt`-Validierer nur der Vollständigkeit halber erwähnt wird. Zudem muss darauf hingewiesen werden, dass dieses Codebeispiel erst mit Archetypes 1.3.2 funktioniert , da `RangeValidator` in den vorigen Versionen einen Bug enthielt.

Dazu definieren Sie nun einen Validierer namens `DateRangeValidator` und registrieren ihn im Validation-Framework. Sie können dies in einem eigenen Python-Modul oder am Beginn des Moduls Ihres Inhaltstyps einfügen. Der `DateRangeValidator` arbeitet ähnlich wie der `RangeValidator`, nur dass er mit Datumswerten der Klasse `DateTime` arbeitet (mehr zur Klasse `DateTime` finden Sie in Anhang A). Eine Validiererklasse muss drei Voraussetzungen erfüllen: Sie muss das Interface `ivalidator` implementieren; sie muss ein Attribut namens `name` besitzen, mit dem die Validiererklasse registriert wird; und sie muss aufrufbar sein, indem sie eine Methode namens `__call__` besitzt, die im Moment der Validierung aufgerufen wird. Das folgende Beispiel zeigt die Validiererklasse `DateRangeValidator`:

```
from Products.validation.interfaces import ivalidator
from DateTime import DateTime

class DateRangeValidator:
   __implements__ = (ivalidator,)

   def __init__(self, name, begindate, enddate):
       self.name = name
       self.begindate=begindate
       self.enddate=enddate

   def __call__(self, value, *args, **kwargs):

       if not isinstance(value, DateTime):
```

```
            value = DateTime(value)

        if self.begindate < value and value < self.enddate:
            return True
        else:
            return "the given date is not between %s and
                   %s"%(str(self.begindate), str(self.enddate))

christmas = DateRangeValidator("ChristmasHolidays",
    DateTime('2004-12-18 00:00:00'),
    DateTime('2005-01-09 00:00:00'),)
validation.register(christmas)
```

Anschließend muss in der Felddeklaration des Schemata noch das `validators`-Attribut gesetzt werden:

```
validators=("ChristmasHolidays",)
```

13.1.3 Definition von Ansichten (Views) und Aktionen (Actions) in der Basisklasse

Archetypes erzeugt für jeden Inhaltstyp die Standard-Ansichten und -Aktionen basierend auf Anforderungen, die den meisten Anwendungen gerecht werden. Diese Standard-Aktionen sind *view*, *edit* und *properties*. In Ihrem Produkt werden Sie keine Page Templates dazu finden, denn diese Views werden automatisch zum Zeitpunkt der Darstellung umgesetzt. Aber selbstverständlich können Sie für Ihre Klasse eigene Ansichten programmieren und anpassen.

Meistens ist es so, dass man für den Inhaltstyp eine eigene *view*-Ansicht programmieren will, denn die Standardansicht zählt einfach nur die Attribute des Schemata mit den dazugehörigen Werten auf und erhebt auch nicht den Anspruch, eine perfekte Seite zu sein, wohingegen die *edit*- und *properties*-(Metadaten-) Ansichten von Haus aus den meisten Ansprüchen gerecht werden.

Im Prinzip wird für jeden Inhaltstyp eine Klasse geschrieben, vergleichbar mit dem Schreiben von Klassen im letzten Kapitel, wobei pro Plone-Produkt beliebig viele solcher Klassen enthalten sein können. Jede Klasse, die einen Inhaltstyp beschreibt, erbt von einer Archetypes-spezifischen Basisklasse, in den meisten Fällen entweder von `BaseContent` oder `BaseFolder`. Erstere Basisklasse wählt man für alle einfachen Inhaltstypen, wie zum Beispiel `Article`, während `BaseFolder` zum Einsatz kommt, wenn man einen Inhaltstyp schreiben will, der seinerseits andere Objekte enthalten soll (im Zope-Jargon *folderish object* genannt). In diesen Basisklassen ist alles enthalten, was vom Archetypes-Framework benötigt wird, so dass man sich in der eigenen Klasse auf das Wesentliche konzentrieren kann.

Wie ich jetzt zeigen werde, besteht dies im Wesentlichen aus zwei Teilen. Zuerst definieren wir eine Aktion, indem wir diese in der Factory-Type-Information (siehe voriges Kapitel) in Form eines Attributs namens `actions` definieren, das eine Liste der einzelnen Aktionsdefinitionen enthält. Zum Beispiel:

```
from Products.Archetypes.public import BaseContent

class Article(BaseContent):
    # other stuff
    actions = ({ 'id': 'view',
                 'name': 'View',
                 'action': 'string:${object_url}/article_view',
                 'permissions': (CMFCorePermissions.View,)
                },)
```

Zweitens benötigen Sie ein Page Template für den View namens `article_view`. Der Name des Templates muss zu der URL im `action`-Parameter passen, denn unter diesem Namen wird das Page Template in der Plone-Skin gesucht. In diesem Fall wird das passende Page Template im `skins/archexample`-Verzeichnis unseres Produkts abgelegt. Im Anhang B können Sie ein vollständiges Listing dieses Page Templates finden. Beim Schreiben dieses Views haben Sie volle Gestaltungsfreiheit.

Um die Änderungen wirksam werden zu lassen, muss Plone neu gestartet werden. In diesem Fall wurde eine Aktion neu definiert, die während des Installationsprozesses des Produkts im *portal_types*-Tool registriert wird. Deswegen muss das Produkt zusätzlich mit dem QuickInstaller reinstalliert werden, indem Sie in Plone auf PLONE KONFIGURIEREN klicken.

Alle Elemente in der Factory-Type-Information (FTI) können als gleichnamige Attribute der Klasse überschrieben werden. Um zum Beispiel das `content_icon`-Element der FTI zu überschreiben, können Sie in der Klasse einfach ein Attribut mit Namen `content_icon` definieren. Sehen Sie sich dazu folgendes Beispiel an:

```
class SomeProduct(BaseContent):
          """ Some product """
          content_icon = "some_icon.gif"
```

13.1.4 Überschreiben von Standardmethoden

In Tabelle 13.2 habe ich die Möglichkeit des Überschreibens von Standardmethoden erwähnt, die in einem Inhaltstyp auftauchen können, etwa `accessor` oder `mutator`. Dies ist eine Option für Fortgeschrittene, die es erlaubt, den Mechanismus zu beeinflussen, wie Felder editiert werden.

Wie im Sourcecode im letzten Kapitel greifen Sie auf die Attribute niemals direkt zu, sondern über die dazugehörigen Get- und Set-Methoden, die in Archetypes `accessors` und `mutators` genannt werden. Diese Zugriffsmethoden werden automatisch von Archetypes generiert. Im Falle eines Feldes namens *blurb* heißen diese beiden Methoden `getBlurb` und `setBlurb`, das Präfix `set` oder `get` wird dem großgeschriebenen Feldnamen vorangestellt.

Jedoch werden Sie vielleicht im Accessor oder Mutator etwas anderes machen wollen. Nehmen wir an, Sie wollen beim Speichern den Wert eines Feldes filtern und korrigieren, wie zum Beispiel die Schreibweise Ihrer Firma korrigieren oder beim Ändern eines Feldinhaltes auch andere Felder ändern. Dies können Sie erreichen, indem Sie die oben erwähnten Standardmethoden überschreiben. Im folgenden Beispiel werden Sie eine neue Methode namens `getSpecialBlurb` schreiben, die im `blurb`-Wert den Text `Perl` durch `Python` ersetzt, bevor der Wert zurückgegeben wird.

```
class Article(BaseContent):

    def getSpecialBlurb(self):
        """ The view for an article """
        blurb = self.getField('blurb').get(self)
        blurb = blurb.replace('Perl', 'Python')
        return blurb
```

Außerdem müssen Sie noch Ihr Feld ändern, damit es diese Methode verwendet:

```
StringField('blurb',
    searchable=1,
    widget=TextAreaWidget(),
    accessor="getSpecialBlurb",
),
```

In diesem Beispiel wird bei jedem lesenden Zugriff auf das *blurb*-Feld in einer View- oder Edit-Seite der Wert von `getSpecialBlurb` zurückgegeben. Archetypes weiß, dass es diese Methode verwenden soll, weil deren Name im `accessor`-Parameter der Felddeklaration festgelegt ist. Ein bisher noch nicht erwähnter Griff in die Trickkiste kommt hier zum Einsatz: Um von `getSpecialBlurb` den Feldinhalt zu bekommen, muss man zuerst das Feldobjekt bekommen und dort die `get`-Methode aufrufen, was man durch den etwas komplexen Ausdruck `blurb-self.getField('blurb').get(self)` erreicht. Dieses Muster, ein Feldobjekt zu holen und dann eine Methode darauf aufzurufen, ist in Archetypes recht gebräuchlich.

Das praktische Ergebnis davon ist nun, dass, sobald jemand das Wort `Perl` im Feld *blurb* eingibt, dieses durch `Python` ersetzt wird.

13.1.5 Den Rest des Inhaltstyps zusammensetzen

Ich habe nun alle Hauptelemente des Inhaltstyps behandelt. Listing 13.1 zeigt den gesamten Code. Sie werden bemerken, dass der Rest des Programmcodes kompakter ist als der Python-Code im letzten Kapitel, da Archetypes sehr viel im Hintergrund für Sie erledigt.

Listing 13.1: Article.py

```
from Products.ArchExample.config import ARTICLE_GROUPS
from Products.Archetypes.public import BaseSchema, Schema
from Products.Archetypes.public import StringField, TextField
from Products.Archetypes.public import SelectionWidget, TextAreaWidget
from Products.Archetypes.public import RichWidget
from Products.Archetypes.public import BaseContent, registerType
from Products.Archetypes.Marshall import PrimaryFieldMarshaller
from Products.CMFCore import CMFCorePermissions
from config import PROJECTNAME

schema = BaseSchema +  Schema((
    StringField('group',
                vocabulary=ARTICLE_GROUPS,
                widget=SelectionWidget(),
                ),
    StringField('blurb',
                searchable=1,
                widget=TextAreaWidget(),
                ),
    TextField('body',
              searchable=1,
              required=1,
              primary=1,
              default_output_type='text/html',
              allowable_content_types=('text/plain',
                                       'text/structured',
                                       'text/restructured',
                                       'text/html',
                                       'application/msword'),
              widget=RichWidget(label='Body'),
              ),
          ),
     marshall=PrimaryFieldMarshaller(),
     )

class Article(BaseContent):
    """This is a sample article; it has an overridden view for show,
```

```
    but this is purely optional
    """

    schema = schema

    actions = ({
        'id': 'view',
        'name': 'View',
        'action': 'string:${object_url}/article_view',
        'permissions': (CMFCorePermissions.View,)
        },)

registerType(Article, PROJECTNAME)
```

Abgesehen von den `import`-Anweisungen am Beginn und dem Schema, habe ich alle Elemente des Sourcecodes behandelt, mit der Ausnahme von `registerType`. Diese Funktion registriert die Klasse mit Ihrem Produkt im Archetypes-Framework. Jedes Produkt kann beliebig viele Inhaltstypen registrieren. So wird diese Funktion `registerType` mit zwei Parametern aufgerufen: der Klasse und dem Namen des Produktes. Der zweite Parameter ist optional und wird nur in Ausnahmefällen benötigt, da die Funktion `registerType` von der Klasse auf das Produkt schließen kann. In diesem Beispiel ist der Produktname in der Datei `config.py` definiert; es ist allgemein üblich, Einstellungen an zentraler Stelle in einer `config.py`-Datei innerhalb des Produkts vorzunehmen, wie zum Beispiel:

Listing 13.2: Die Konfigurationsdatei von ArchExample

```
from Products.CMFCore.CMFCorePermissions import AddPortalContent
from Products.Archetypes.public import DisplayList

ADD_CONTENT_PERMISSION = AddPortalContent
PROJECTNAME = "ArchExample"
SKINS_DIR = 'skins'

GLOBALS = globals()

ARTICLE_GROUPS = DisplayList((
    ('headline', 'Headline'),
    ('bulletin', 'Special Bulletin'),
    ('column', 'Weekly Column'),
    ('editorial', 'Editorial'),
    ('release', 'Press Release'),
    ))
```

Die Variable `ARTICLE_GROUPS` ist eine Liste von Wertepaaren für das Vokabular des *group*-Feldes in *Article*, wobei jeweils der erste Wert bei Auswahl im Feld gespei-

chert wird und der zweite als zugehöriger Text in der Auswahlliste erscheint. Man kann allerdings auch eine einfache Liste von Werten als Vokabular angeben, wenn man nicht zwischen den Werten und deren Beschriftung unterscheiden muss. In Fall dieses Beispiels sieht der erzeugte HTML-Code für das *groups*-Feld folgendermaßen aus:

```
<option value="headline">Headline</option>
<option value="bulletin">Special Bulletin</option>
...
```

Ich möchte hier noch auf die Verwendung von `globals()` hinweisen. Dies ist eine eingebaute Python-Funktion, die alle Symbole aus dem globalen Namespace zurückgibt. Sie wird hier verwendet, um den Pfad zum `skins`-Verzeichnis im Filesystem zu berechnen, so dass Sie ein Filesystem-Directory-View davon für das Skins-Tool erstellen können. Das Produktinitialisierungsmodul `__init__.py` ist dadurch also wesentlich einfacher. Mit einem neuen Zusatz sehen die Funktionen `process_types` und `listTypes` nun so aus:

```
from Products.Archetypes.public import process_types, listTypes
content_types, constructors, ftis = process_types(
    listTypes(PROJECTNAME),
    PROJECTNAME)
```

Die `listTypes`-Funktion aus dem Archetypes-Framework gibt alle Typdeklarationen zurück, die bisher im angegebenen Produkt registriert wurden. Diese Liste wird von `process_types` noch weiter behandelt, die dann die Inhaltstypen, Konstruktoren und Factory-Type-Informationen (FTI) zurückgibt, die sodann beim CMF mit der Funktion `ContentInit` registriert werden. Vieles ist ähnlich wie beim Schreiben von reinen Python-Produkten für das CMF, nur dass diese Funktionen das Ganze ein wenig vereinfachen.

Schließlich gibt es noch die `install`-Funktion im Script `Extensions/Install.py`. Im Gegensatz zum `__init__.py`-Modul, das aufgerufen wird, wenn das Produkt beim Hochstarten von Zope registriert wird, wird das Installationsscript aufgerufen, wenn ein Produkt in einer bestimmten Plone-Instanz zum Beispiel durch den QuickInstaller installiert wird. Dieses Script vereinfacht sich durch Archetypes ebenfalls wesentlich, da die meiste Arbeit durch `installTypes` und `install_subskin` abgehandelt wird. Der Begriff *Subskin* ist ein wenig irreführend – eigentlich ist damit ein Skin-Layer gemeint.

Listing 13.3: Installationsscript

```
from Products.Archetypes.public import listTypes
from Products.Archetypes.Extensions.utils import installTypes,
install_subskin
```

```
from Products.ArchExample.config import  PROJECTNAME, GLOBALS

from StringIO import StringIO

def install(self):
    out = StringIO()
    installTypes(self, out, listTypes(PROJECTNAME), PROJECTNAME)
    install_subskin(self, out, GLOBALS)
    out.write("Successfully installed %s." % PROJECTNAME)
    return out.getvalue()
```

Das vollständige Paket von ArchExample ist auf der Homepage des Autors des Plone-Buches unter *http://plone-book.agmweb.ca* zum Download verfügbar, Sie können es durch Auspacken in Ihrem Plone ausprobieren. Wie Sie gesehen haben, ist dieser Inhaltstyp schnell und einfach zu entwickeln und auch leicht zu ändern, ohne viel Code zu schreiben.

13.2 Entwickeln mit Archetypes

Dieser Abschnitt wird Sie in einige fortgeschrittene Programmiertechniken von Archetypes einführen, wie zum Beispiel Referenzen, Erstellen von eigenen Widgets und Transformationen.

Für diesen Abschnitt ist es wichtig zu sehen, wie die Änderungen, die Sie am Produkt vornehmen, in Plone wirksam werden. Wie Sie bisher gesehen haben, gibt es verschiedene Stadien in der Entwicklung des Produkts. Wenn Sie Ihr Produkt ändern, ist es nützlich, die notwendigen Schritte zu kennen.

Wenn Sie etwas in einer Skin ändern und Zope im Debug-Modus läuft, werden diese Änderungen sofort wirksam. Wenn Sie Dinge ändern, die im Installationsprozess involviert sind, wie etwa das *portal_actions*-Tool oder das *portal_types*-Tool, müssen Sie Plone neu starten und im QuickInstaller Ihr Produkt reinstallieren.

Wenn Sie das Schema ändern, reicht ein Neustart von Plone. Hingegen stellt sich die Frage, was dann mit bereits existierenden Objekten passiert. Archetypes bietet hier im *archetype-tool* ein *Update Schema*-Werkzeug, das auf Wunsch über alle Archetypes-Objekte iteriert und sie mit dem neuen Schema abgleicht. Zu diesem Zweck klicken Sie im ZMI auf ARCHETYPE_TOOL und wählen UPDATE SCHEMA. Nun können Sie die Klassen selektieren, deren Schema abgeglichen werden soll, und klicken sodann auf den UPDATE SCHEMA-Button; dies wird daraufhin alle Instanzen dieser Klassen mit dem neuen Schema abgleichen.

13.2.1 Verwendung von eindeutigen Schlüsseln (Unique IDs, UIDs)

Das Konzept von eindeutigen Schlüsseln ist einfach, aber hat in Zope bislang gefehlt. Ursprünglich glaubten Zope-Entwickler, dass der Pfad auf ein Objekt für dessen Adressierung ausreichend sei; unglücklicherweise hat sich das als unzureichend herausgestellt. Was passiert zum Beispiel, wenn ein Objekt verschoben wird? Alle Referenzen auf dieses Objekt würden nun ins Leere verweisen, da sich durch das Verschieben der Pfad ändert. Mit Hilfe der eindeutigen Schlüssel steht eine elegante Lösung des Problems zur Verfügung. Archetypes vergibt und speichert automatisch für jedes Objekt eine UID, die in einem speziellen Katalog namens `uid_catalog` indiziert wird.

Sie können diesen *uid_catalog* im ZMI ansehen. Dieser ist dem *portal_catalog* ziemlich ähnlich, nur dass er ausschließlich die für die Verwaltung und Suche von UIDs nötigen Indizes enthält. Es ist ziemlich einfach, ein Objekt aufgrund seiner UID über den *uid_catalog* zu erhalten. Sie müssen lediglich über den UID-Index nach der entsprechenden UID suchen; bei jedem Objekt können Sie dessen UID mit Hilfe der Funktion `UID()` abfragen. Das nachfolgende Python-Script kann jedes Objekt aus dem Katalog suchen, vorausgesetzt, Sie kennen dessen *UID*.

```
##paramaters=objectId
results = context.uid_catalog(UID=objectId)
return results[0].getObject()
```

Wirklich nützlich sind die eindeutigen Schlüssel bei der Verwaltung von Referenzen zwischen Objekten. Nehmen wir an, Sie wollen von Ihrem Artikel aus verschiedene Bilder referenzieren, die sich in Ihrem Portal befinden. Wenn diese Bilder Archetypes-Ojekte sind, können Sie wie folgt ein `ReferenceField` zu Ihrem Schema hinzufügen:

```
ReferenceField("images"
    allowed_types=("Archetype Images",),
    multivalued=1,
    ),
```

Der User bekommt nun eine Dropdown-Liste mit den Titeln aller *Archetype Image*-Objekte und kann eines auswählen. Im Hintergrund werden beim Speichern Referenzen zu den Zielobjekten angelegt. Verantwortlich für die Referenzlogik in Archetypes ist die *ReferenceEngine*; der interessierte Leser möge hierfür den Code in `Archetypes/ReferenceEngine.py` studieren. Im Archetypes-Subversion-Archiv unter *http://svn.plone.org* gibt es das Produkt *ACME*, das viele der fortgeschrittenen Programmiertechniken wie die Referenzen demonstriert. Die Installation ist aber aufgrund von weiteren Produkten, die benötigt werden, nicht ganz einfach.

13.2.2 Anpassen von Widgets

Eine häufig gestellte Frage ist: »Warum macht dieses Widget das eigentlich?« oder: »Warum sieht dieses Widget so aus?« Oft lautet die Antwort: »Weil es so geschrieben ist – Sie können es sich ja anpassen.« Da die Widgets das sind, was der Kunde sieht, ändern sich die Anforderungen oft, so dass dem Entwickler hier die Flexibilität von Archetypes zugute kommt.

Alle Widgets bestehen aus einer Python-Klasse – die vorgegebenen Widgets kann man in `Archetypes/Widget.py` sehen – und den dazugehörigen Page Templates in `Archetypes/skins/archetypes/widgets`. Wenn Sie im ZMI auf PORTAL_SKINS klicken und dort ARCHETYPES/WIDGETS wählen, sehen Sie alle mit Archetypes gelieferten Widgets. Unglücklicherweise hat das *portal_skins*-Tool von CMF einen kleinen Bug. Weil die Widgets in einem Unterverzeichnis sind, kann man leider nicht einfach eines der Templates ändern. Das ist aber nicht so schlimm, Sie können in Ihrem Produkt einen eigenen Skin-Layer definieren und dann in der Widget-Deklaration des Schemata spezifizieren, dass das Widget mit einem anderen Makro dargestellt werden soll. Jedes Widget-Objekt hat ein `macro`-Attribut, das definiert, welches Makro das Widget darstellen soll, und das als Pfad-Ausdruck zum Widget *Page Template* spezifiziert wird.

Nun ist der Begriff *macro* irreführend, denn das Attribut verweist auf ein Page Template, das seinerseits drei Makros enthält: `view`, `edit` und `search`. Diese Makros sind für das Anzeigen Ihres Widgets in den verschiedenen Situationen zuständig.

Das `view`-Makro ist eine benutzerfreundliche Ansicht mit rein lesendem Zugriff und wird zum Beispiel in der `base_view`-Ansicht pro Feld angezeigt. Das `edit`-Makro wird in der Bearbeitungsansicht angezeigt und ist das Makro, mit dem der Benutzer die Daten für das einzelne Feld eingibt. Es kommt zum Beispiel in der `base_edit`-Ansicht zum Einsatz. Das `search`-Makro wird verwendet, wenn Sie eine Suchmaske für Ihren Inhaltstyp zusammenstellen wollen; es sieht häufig wie das `edit`-Makro aus, muss aber nicht unbedingt so aussehen. Ein Stringfeld kann zum Beispiel in einem `RichWidget` eingegeben werden, aber der Suchbegriff dazu könnte mit einem `StringWidget` eingegeben werden.

Nun, für unser Beispielprodukt wollen wir ein `StringField` für die E-Mail-Adresse einer Person verwenden, die in der `view`-Ansicht als klickbarer E-Mail-Link dargestellt wird. Dafür benötigen wir ein neues `view`-Makro; die Makros für `edit` und `search` können wir vom `StringWidget` nehmen. Dazu erstellen Sie ein neues Page Template namens `email_widget.pt` in der Skin Ihres Produkts. Für `edit` und `search` werden einfach die entsprechenden Makros des `StringWidgets` aufgerufen:

```
<html xmlns:tal="http://xml.zope.org/namespaces/tal"
      xmlns:metal="http://xml.zope.org/namespaces/metal"
```

```
      i18n:domain="plone">

  <body>
    <div metal:define-macro="edit">
      <div metal:use-macro="here/widgets/string/macros/edit" />
    </div>

    <div metal:define-macro="search">
      <div metal:use-macro="here/widgets/string/macros/search" />
    </div>
```

Für `view` erzeugen Sie einen `mailto`:-Link, den Sie durch folgende kleine Änderung erreichen:

```
    <div class="field" metal:define-macro="view">
        <a href="#" tal:attributes="href string:mailto:${accessor}"
          tal:content="accessor">E-Mail</a>
    </div>
</body>
</html>
```

Nach dem Definieren des neuen Page Templates mit Ihrem eigenen Code können Sie einfach dessen Namen in der Widget-Deklaration als `macro`-Parameter angeben. Im folgenden Codebeispiel definieren Sie das E-Mail-Feld als `StringField` mit zugehörigem `StringWidget` wie üblich, nur dass Sie noch das zu verwendende Makro explizit angeben:

```
StringField('E-Mail',
    validators = ('isEmail',),
    widget = StringWidget(
        label='E-Mail',
        macro='email_template'
    )
)
```

Bis jetzt haben Sie lediglich das Makro eines bereits vorhandenen Widgets geändert. Ein ganz neues Widget zu erstellen erfordert das Schreiben einer eigenen Widget-Klasse und deren Registrierung. Alle Widgets haben eine gemeinsame Basisklasse. Im folgenden Beispiel schreiben wir ein eigenes Modul namens `EmailWidget.py` und speichern es in das `ArchExample`-Verzeichnis. Es enthält die neue Widget-Klasse und deren Registrierung. Beachten Sie, dass die `macro`-Property des Widgets auf das `email_template` voreingestellt ist:

```
from Products.Archetypes.Widget import TypesWidget
from Products.Archetypes.Registry import registerWidget

class EmailWidget(TypesWidget):
```

```
    _properties = TypesWidget._properties.copy()
    _properties.update({
       'macro' : "email_template",
       'size' : '30',
       'maxlength' : '255',
       })

registerWidget(EmailWidget,
    title='String',
    description='Renders a clickable E-Mail field',
    used_for=('Products.Archetypes.Field.StringField',)
)
```

Um das neue Widget in Ihrer Article-Klasse zu verwenden, importieren Sie `EmailWidget` und geben es wie folgt im `widget`-Parameter zum Feld an:

```
from EmailWidget import EmailWidget

StringField('E-Mail',
    validators = ('isEmail',),
    widget = EmailWidget(
        label='E-Mail',

    )
)
```

13.2.3 Das Entwickeln von ordnerartigen Objekten (Folderish Objects)

Sie haben vermutlich schon oft mit ordnerartigen Objekten in Plone gearbeitet, ohne sich dessen bewusst zu sein. Ein ordnerartiges Objekt ist eines, das ähnliche Eigenschaften wie ein Ordner aufweist, nämlich dass es seinerseits Unterobjekte enthalten kann. Das Erstellen eines ordnerartigen Objekts ist nichts Besonderes – die Klasse muss lediglich von einer besonderen Basisklasse erben, um die entsprechenden Fähigkeiten zur Verfügung zu stellen, und schon kann man zu unserem Objekt neue Unterobjekte hinzufügen.

Ordnerartige Objekte sind aus verschiedenen Gründen nützlich. Sie stellen einen einfachen Weg dar, um Sammlungen von einzelnen Objekten zu erzeugen. Sie erlauben außerdem, dass Plone-Benutzer ihre speziellen Inhaltstypen über die gewohnte Plone-Oberfläche verwalten können, ohne dass Sie dafür speziell programmieren müssen. Generell ist es am besten, den Ordner einfach zu halten, und die Logik sollte in Ihren Objekten und in deren Workflow enthalten sein. Natürlich gibt es Ausnahmen dazu, und ein klassisches Beispiel ist die Familie

der Bugtracker, *CMFCollector* und *PloneCollectorNG*, die beide komplexe ordnerartige Objekte sind, die einiges an Logik für die einzelnen Einträge bieten.

Der einfachste Weg, um ordnerartige Objekte zu erstellen, ist wiederum die Verwendung von Archetypes. Sie haben in den vorigen Beispielen gesehen, dass `BaseContent` alle nötigen Standardarbeiten für »normale« Objekte erledigt. Nun gibt es auch eine Klasse `BaseFolder`, die die entsprechenden Aufgaben für ordnerartige Objekte übernimmt. Außerdem besitzen Ordner ein eigenes Basisschema `BaseFolderSchema`. Auf den Punkt gebracht: Um einen ordnerartigen Inhaltstyp zu erstellen, ändern Sie die Basisklasse auf `BaseFolder` und fügen sie zum Schema `BaseFolderSchema` hinzu. Ein einfaches Beispiel für einen ordnerartigen Inhaltstyp sehen Sie hier:

```
from Products.Archetypes.public import BaseFolder, BaseFolderSchema

schema = BaseFolderSchema

class SimpleFolder(BaseFolder):
    """A simple folderish archetype"""
    schema = schema

registerType(SimpleFolder)
```

Wenn Sie beabsichtigen, eine große Menge von Unterobjekten in einem Ordner zu speichern, dann ist die Klasse `BaseFolder` weniger geeignet, da diese Klasse von der Zope-Basisklasse `Folder` erbt, die nicht auf große Datenmengen ausgelegt ist. Für große Mengen von Unterobjekten empfiehlt sich die Verwendung von Binary-Tree-Ordnern, die die Unterobjekte intern in einem Binärbaum effizienter speichern statt in einem Python-Dictionary, wie dies bei `Folder` der Fall ist. In diesem Fall lassen Sie Ihre Klasse von `BaseBTreeFolder` erben und nehmen als Basisschema `BaseBTreeFolderSchema`. Was die Produktentwicklung betrifft, sind die beiden gleich zu programmieren, nur dass auch bei einer Anzahl von mehr als 100.000 Objekten der `BtreeFolder` gut skaliert.

13.2.4 Arbeiten mit Microsoft Office-Dateien

Der Umgang mit Microsoft Office-Dateien wie zum Beispiel Word oder Excel ist ein Fluch, mit dem jedes Content-Management-System irgendwann einmal konfrontiert wird. Aber dies ist der Fall mit verschiedenen Dateiformaten – Microsoft Office, OpenOffice.org, Portable Document Format (PDF), Imagedateien usw. Der Umgang mit diesen Dateiformaten auf Webseiten verursacht normalerweise einige Probleme; dies ist natürlich wohlbekannt. Das Editieren ist umständlich, da durch das Anklicken das Dokument heruntergeladen wird oder – noch schlimmer – in Ihrem Webbrowser geöffnet wird. Wenn Sie es fertig editiert

haben, müssen Sie das Dokument abermals auf die Website hochladen, was wiederum eine Runde des Herumklickens bedeutet. Wenn das Dokument endlich online ist, ist dieses meistens nicht suchbar, weil Binärdateien nicht indiziert werden. Außerdem können Sie das Dokument nicht online betrachten, weil der Webbrowser das jeweilige Datenformat ohne spezielle Plugins nicht darstellen kann.

Es gibt einige Lösungen, um das Problem des Editierens von Inhalt zu lösen. Wenn Sie weiter annehmen, dass die meisten Ihrer Anwender Windows verwenden, dann kann die Verwendung von WebDAV mit Komplikationen verbunden sein, da Microsofts Implementation des WebDAV-Clients (Web Folders) von fraglicher Qualität ist. Hingegen erfüllt der External Editor diese Funktion recht gut. Wenn Plone dem External Editor mitteilen kann, dass es sich bei dem editierten Objekt zum Beispiel um eine Word-Datei handelt, wird der External Editor das Dokument mit Word öffnen.

Und für den Rest enthält Archetypes ein Transformationspaket namens *Portal Transforms*, das die Transformation von Inhaltstypen beherrscht. Dies kann ein Dokument in einem bestimmten Datenformat in HTML transformieren, wonach es katalogisiert werden kann, und als HTML kann es auch im Webbrowser dargestellt werden. Dies funktioniert durch die Verwendung eines externen Transformationsprozesses, um die Daten umzuwandeln und das Ergebnis auszulesen und darzustellen. Wenn Sie zum Beispiel Plone unter Windows (auf dem Server) verwenden, dann nimmt Portal Transforms ein auf dem Server installiertes Microsoft Office als COM-Server, um die Word-Datei in HTML umzuwandeln.

All dies passiert im Hintergrund. Sie müssen lediglich sicherstellen, dass die für die Transformationen benötigten Komponenten installiert sind und funktionieren. Es gibt verschiedene Typen von Transformationen, und oft gibt es mehr als eine Möglichkeit, Daten zu transformieren.

Das OpenOffice-Paket stellt hervorragende Transformationen von Microsoft Office-Daten zur Verfügung (oft sogar besser als MS Office selbst!) und bietet somit auch auf Nicht-Microsoft-Plattformen die Möglichkeit, Office-Daten zu verarbeiten. Ich habe auch schon *wvWare* (*http://wvware.sourceforge.net/*) verwendet, ein Kommandozeilen-Werkzeug, das es ermöglicht, verschiedene Microsoft-Formate in HTML zu übersetzen. All diese Optionen stehen Ihnen zur Verfügung; oft sind die Komponenten aber nicht einfach zu installieren. Nach dem Aufsetzen der Komponenten berichten Benutzer, dass die besten Ergebnisse mit Microsoft Office und Openoffice.org als Backend der Transformationen erzielt werden.

Ich empfehle, zuerst genau zu überlegen, was transformiert werden soll, und dann einen Blick in den Quellcode von *Portal Transforms* zu werfen, um zu sehen, ob Sie eine geeignete Transformation für Ihre Zwecke finden.

Einrichten eines Inhaltstyps

Sie werden nun einen einfachen Inhaltstyp erstellen, der Microsoft Office Word-Dokumente verarbeiten kann. Word ist sicher der Dokumenttyp, den Sie am häufigsten verarbeiten werden. In diesem Beispiel habe ich die Transformation unter Windows durchgeführt. Wenn Sie dies hingegen auf einem alternativen System wie zum Beispiel Linux aufsetzen, ändert sich am Code Ihres Inhaltstyps nichts, es kommt lediglich eine andere Transformation vom Dokumenttyp `application/msword` nach HTML zum Tragen. Welche Transformationen in einem Portal zur Verfügung stehen, wird im Werkzeug *portal_transforms* hinterlegt. In diesem Beispiel ist es am einfachsten, Ihre Plone-Instanz unter Windows aufzusetzen, wo Microsoft Office schon installiert ist. Der Plone Installer richtet eine Plone-Instanz mit allen notwendigen Win32-API-Modulen ein. Dann erstellen Sie einen Inhaltstyp für dieses Beispiel mit Namen `WordExample`. Im Wesentlichen ist ein Feld in diesem Schema für die Speicherung von Word-Dokumenten zuständig, so dass sich folgendes einfaches Schema ergibt:

```
schema = BaseSchema +  Schema((
    TextField('body',
              searchable=1,
              required=1,
              primary=1,
              default_output_type='text/html',
              allowable_content_types=('application/msword',),
              widget=RichWidget(label='Body'),
              ),
    ),
    marshall=PrimaryFieldMarshaller(),
    )
```

Der einzige Unterschied zu einem normalen `TextField` ist die Angabe des Multipurpose Internet Mail Extensions-(MIME-)Typs für Microsoft Word im `allowable_content_types`-Attribut. Für jeden Dokumenttyp, den dieses Feld zur Transformation akzeptieren soll, geben Sie den entsprechenden MIME-Typ im `allowable_content_types`-Attribut an. Wenn Sie beispielsweise Microsoft Office- und PDF-Dateien verarbeiten wollen, geben Sie folgende Zeile an:

```
allowable_content_types=('application/msword','application/pdf'),
```

Dieser Inhaltstyp ist das einfachstmögliche Beispiel in diesem Moment. Aber Sie können diesen Inhaltstyp erweitern wie jeden anderen, zum Beispiel um ein Beschreibungsfeld und andere Attribute. Eine sinnvolle, aber arbeitsintensive Erweiterung wäre zum Beispiel, die Metadaten des Word-Dokuments in die Metadaten des Plone-Objekts zu speichern.

Wenn Sie bereits Archetypes 1.3 verwenden, empfiehlt sich hier die Installation der *ATContentTypes* (*http://sf.net/projects/collective*), die die klassischen Inhaltstypen von Plone (Document, File,...) durch kompatible Archetypes-basierende Klassen ersetzen und spätestens ab Plone 2.1 standardmäßig mit Plone installiert werden. Der *Document*-Inhaltstyp verwendet hier bereits `PortalTransforms`, um alle möglichen Transformationen durchzuführen, so dass Sie hier keinen eigenen Inhaltstyp mehr schreiben müssen, um in den Genuss des Transformations-Frameworks von Archetypes zu kommen.

Transformationen unter Windows einrichten

Die Transformation sollte automatisch durch den Installationsprozess im Tool *portal_transforms* registriert sein. Hier gibt es nicht viel Arbeit an dem Tool; es liefert im Wesentlichen eine Liste von verfügbaren Transformationen. Eine Transformation hat eine Liste von eingehenden MIME-Typen (wie zum Beispiel `application/msword`), die sie transformieren kann, und einen ausgehenden MIME-Typ, der produziert wird (so wie in den meisten Fällen `text/html`). Jede Transformation ist ein Python-Modul im Dateisystem; in diesem Fall finden Sie den Code in `PortalTransforms/transforms/word_to_html`.

Um diese Transformation unter Windows zum Laufen zu bringen, müssen Sie ein Kommandozeilen-Tool aufrufen, das die COM-Bindungen für Python generiert. Um dies zu tun, starten Sie PYTHONWIN aus der PLONE-Gruppe in Ihrem Startmenü und selektieren TOOLS – COM MAKEPY UTILITY. Dieses listet alle verfügbaren COM-Interfaces auf; wenn Sie Office installiert haben, werden Sie hier einen Eintrag für Microsoft Office in der Liste finden. Auf meinem Windows-Server lautete dieser Eintrag MICROSOFT WORD 9.0 OBJECT LIBRARY (8.1). Wählen Sie diesen aus, und klicken Sie auf OK. PythonWin wird nun die entsprechende Information generieren. Wenn Sie dies durchgeführt haben, bekommen Sie die Meldung `'Generating to ...'`.

Sie können nun PythonWin schließen und Plone neu starten.

Testen des Inhaltstyps

Um den Inhaltstyp zu testen, starten Sie Plone neu, stellen sicher, dass Ihr Produkt wie üblich registriert ist, und installieren es in Ihrem Portal mit PRODUKTE HINZUFÜGEN/LÖSCHEN. Nun legen Sie eine Instanz Ihres neuen Inhaltstyps an.

Anschließend selektieren Sie eine Word-Datei aus Ihrem Dateisystem und klicken auf SAVE, um sie zu Plone hochzuladen. Daraufhin wird diese Datei hochgeladen, und Sie können das Resultat sehen. Verlieren Sie hier nicht die Geduld, denn dieser Vorgang kann länger dauern, weil zusätzlich zum normalen Upload der Daten Word als externes Programm aufgerufen werden muss, um die HTML-Datei mit eventuell eingebetteten Bildern zu erzeugen. Wenn alles richtig arbeitet, sollten Sie Ihr Word-Dokument als HTML etwa wie in Abbildung 13.4 sehen.

Abbildung 13.4: Die hochgeladene Datei

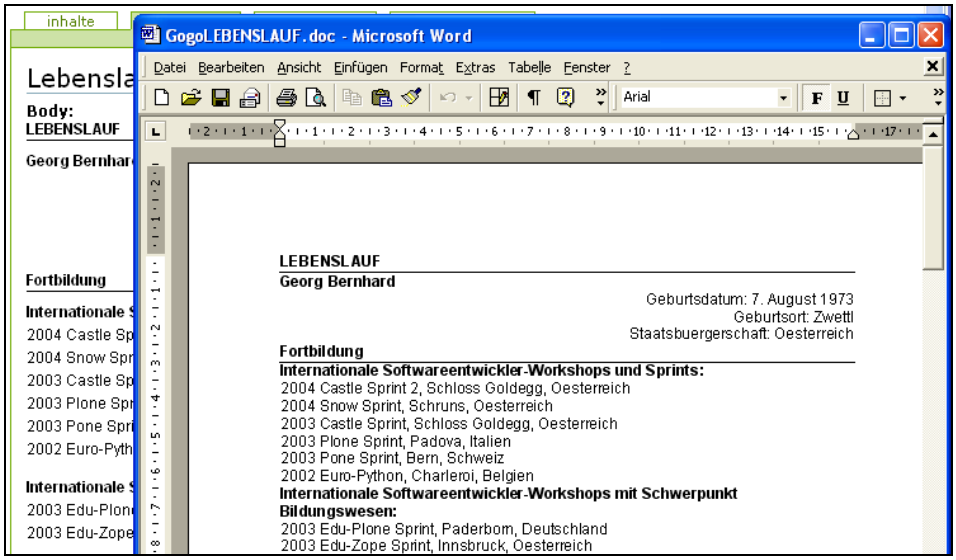


Abbildung 13.5: Das in Word geöffnete Dokument

Nun, da Sie das Dokument in Plone haben, wollen Sie es auch editieren können. Hierzu empfiehlt sich der Einsatz von External Editor. Wenn Sie External Editor auf Ihrem Arbeitsplatz installiert haben, klicken Sie auf das Bleistiftsymbol in der oberen rechten Ecke, und die Datei wird in Word geöffnet, wie in Abbildung 13.5

zu sehen ist. Sie können nun das Dokument ändern und editieren; jedes Mal, wenn Sie die Datei speichern, wird diese im Hintergrund zu Plone hochgeladen und wiederum zu HTML transformiert.

Alternative Transformationen von Office-Dokumenten

Die Verwendung von Microsoft Office setzt nicht nur Windows auf dem Plone-Server voraus, sondern auch ein dort installiertes Microsoft Office nebst dazu gültigen Lizenzen.

Die Installation von *wvWare* unter Linux gestaltet sich recht einfach, so dass Sie hier relativ schnell zu einer Word-Transformation kommen, da die Verfügbarkeit von *wvWare* von *Portal Transforms* automatisch erkannt wird. Dazu besorgen Sie sich das Paket von *http://wvware.sourceforge.net* und folgen den (etwas veralteten) Installationsanleitungen. Für Debian-Anwender gibt es das vorkonfigurierte Paket *wv*, das im Handumdrehen ohne manuellen Eingriff installiert ist. *wvWare* steht dadurch als Kommandozeilenbefehl zur Verfügung und wird von *Portal Transforms* automatisch erkannt. Allerdings ist die Qualität des Resultats nicht mit der Microsoft Office- oder OpenOffice.org-Transformation vergleichbar.

OpenOffice.org bietet vergleichbar mit Microsofts COM eine applikationsübergreifende Programmierschnittstelle namens UNO, zu der es auch ein Python-Binding mit Namen *PyUNO* gibt. Um OpenOffice innerhalb von Plone als Transformation-Backend verfügbar zu machen, muss die PyUNO-Schnittstelle in der Python-Installation, die von Plone verwendet wird, installiert werden, was jedoch einige manuelle Schritte erfordert.

Derzeit entsteht ein weiteres Produkt namens *BlueDCS*, das vielleicht bei Drucklegung dieses Buches als Betaversion verfügbar sein wird. Es teilt die Office-Transformation auf zwei Serverprozesse auf: in einen von Plone unabhängigen XMLRPC-Server, der sich um die Kommunikation mit OpenOffice kümmert, und in ein sehr einfach zu installierendes Plone-Produkt, das die Transformationsanforderung an jenen Server delegiert. Dies bringt den Vorteil, dass die Arbeit des Einrichtens von OpenOffice an zentraler Stelle nur einmal durchgeführt werden muss und etwaige Inkompatibilitäten zwischen der von Plone verwendeten Python-Version und PyUNO ausgeschlossen sind, da der XMLRPC-Server einfach und ohne großen Installationsaufwand mit der von OpenOffice mitinstallierten Python-Distribution gestartet wird. Der Installationsaufwand innerhalb von Plone beschränkt sich auf das Aktivieren des Produktes und auf die Angabe der Adresse des *BlueDCS*-Servers.

Tipp

Sie können *Portal Transforms* vom *Archetypes*-CVS unter *http://sf.net/projects/archetypes* herunterladen. Sie werden im Quellcode von *Portal Transforms* Dokumentation vorfinden, und es gibt auch eine Online-Version unter *http://www.logilab.org/projects/portaltransforms/documentation* .

13.2.5 Fortgeschrittenes Entwickeln: Erstellen von Inhaltstypen mit UML

Nun haben Sie ein Produkt mit mehreren komplizierten Inhaltstypen und finden die Idee, diese manuell zu schreiben, langweilig? Nun, keine Angst, denn ArchGenXML ist hier! Dies ist in hilfreiches Produkt, das es Ihnen erlaubt, Ihre Plone-Applikation mit UML in einem UML-Modellierungstool Ihrer Wahl zu modellieren und aus diesem Modell ein lauffähiges Plone-Produkt zu generieren.

Philipp Auersperg hat das ArchGenXML-Projekt begonnen, und Jens Klein hat vieles an Code und Dokumentation dazu beigetragen. Ich werde in diesem Abschnitt einige der Fähigkeiten von ArchGenXML vorstellen, das zur Drucklegung dieses Buches in der Version 1.2 vorliegt.

Um mit ArchGenXML zusammenarbeiten zu können, muss ein UML-Tool den Export des Modells als *XMI* (XML-Anwendung zum Austausch von Modell-Metadaten) unterstützen. ArchGenXML wurde erfolgreich mit den folgenden Produkten getestet:

- *ObjectDomain* (kommerziell, die Demoversion erlaubt das Editieren von bis zu 30 Klassen), erhältlich unter *http://www.objectdomain.com*
- *ArgoUML* (frei), erhältlich unter *http://argouml.tigris.org*
- *Poseidon* (kommerziell, basierend auf ArgoUML, kostenlose Community-Edition verfügbar), erhältlich unter *http://www.gentleware.com*
- *Sybase Powerdesigner* (kommerziell, zeitlich limitierte Demoversion verfügbar), erhältlich unter *http://www.sybase.com*

Der Export von XMI-Modelldateien wird von den verschiedenen UML-Tools leicht unterschiedlich gehandhabt (zum Beispiel exportiert *ObjectDomain* keine Statemachines). Aber wenn Sie einen Blick auf die ArchGenXML-Beispiele werfen, sehen Sie einige Beispielmodelle, die die Funktionalität von ArchGenXML demonstrieren. Abbildung 13.6 zeigt ein einfaches Modell mit einigen Klassen und Beziehungen.

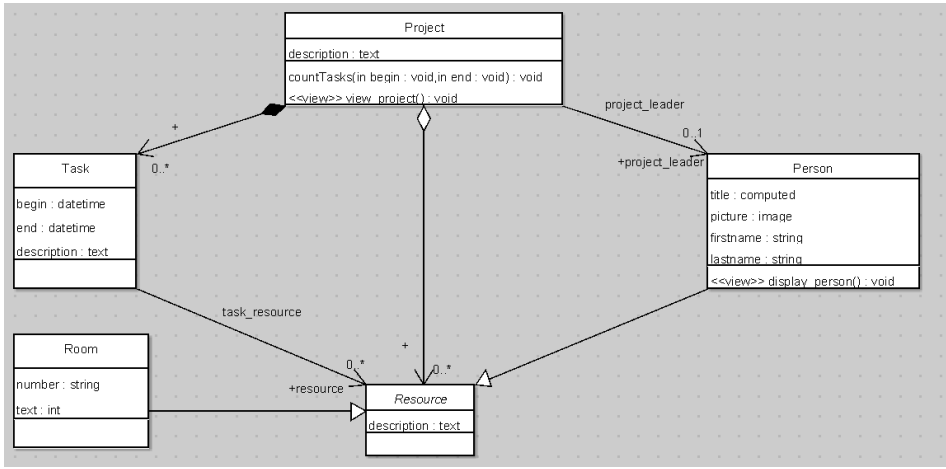


Abbildung 13.6: Ein komplexes Beispiel eines Datenmodells

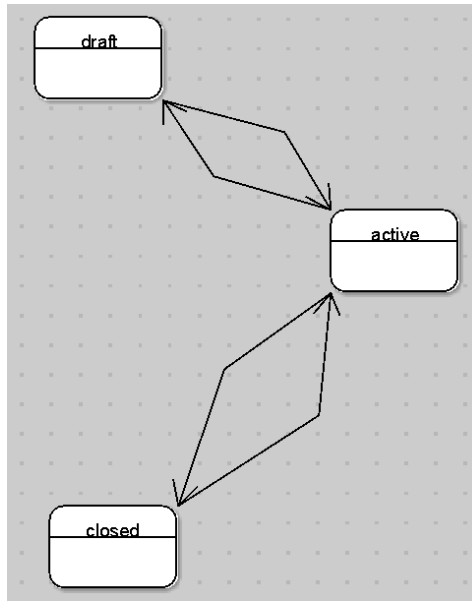


Abbildung 13.7: Zustandsdiagramm für die Klasse Project

Der einfachste Weg, um daraus ein lauffähiges Plone-Produkt zu generieren, besteht darin, in das Verzeichnis `samples/PloneBook` in Ihrem ArchGenXML-Verzeichnis zu wechseln und den unten stehenden Befehl aufzurufen. Damit wird aus dem ArgoUML-Modell ein lauffähiges Plone-Produkt namens *Project* erzeugt.

```
python ../../ArchGenXML.py product.zargo Product
```

Sie müssen sodann das Produkt in Ihren `Products`-Ordner Ihrer Plone-Installation kopieren und Zope neu starten. Wenn Sie einen Blick in das eben generierte Produkt werfen, werden Sie sehen, dass alles Notwendige für Sie generiert wurde: die Klassendateien, die Modulinitialisierungsdatei `__init__.py` und die Scripten zum Installieren des Produktes in Ihre Plone-Instanz im `Extensions`-Verzeichnis. Im neu gestarteten Plone müssen Sie nun lediglich PLONE KONFIGURIEREN wählen und das neu verfügbare Produkt installieren, und schon können Sie mit den neuen Inhaltstypen experimentieren. Sie werden die Attribute, die im Datenmodell für die Klassen vergeben wurden, im Edit-Bildschirm der Inhaltstypen wiedererkennen.

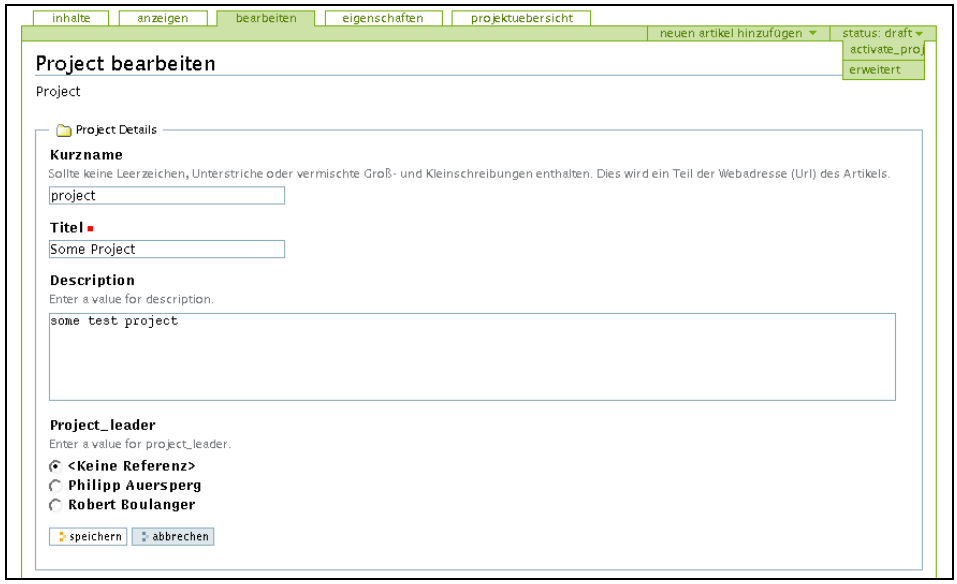


Abbildung 13.8: Das frisch generierte Produkt im Einsatz

Was passiert mit den manuellen Änderungen, die Sie an Ihrem Produkt-Quellcode vornehmen, wenn Sie mit ArchGenXML noch einmal das Modell über Ihr modifiziertes Produkt generieren? Nun, ArchGenXML stellt zwar kein Roundtripping zur Verfügung, aber es behält bestimmte Änderungen bei, die Sie im Quellcode vornehmen. So werden Methoden, die Sie in der Klassendefinition modifiziert oder hinzugefügt haben, beibehalten; außerdem werden Sie im Quellcode geschützte Codebereiche finden (mit `#code-section` markiert), die beim Generieren beibehalten werden. Ebenso werden sämtliche Page Templates beibehalten, die innerhalb des `skins`-Ordners generiert wurden.

Welche UML-Konzepte werden von ArchGenXML unterstützt? ArchGenXML stellt Ihnen eine Reihe von Konzepten aus der UML-Spezifikation für die Plone-Entwicklung zur Verfügung, ich möchte sie hier kurz anhand des oben erwähnten Beispiels anführen:

- **Klassen**: ArchGenXML generiert aus jeder Klasse im Modell den Code für genau einen Inhaltstyp. Klassen, die als Abstrakt markiert wurden, werden als nicht instanziierbare Inhaltstypen generiert (z.B. *Resource*).
- **Methoden**: Jede Methode im Modell entspricht einer Methode innerhalb der Klassendefinition des Inhaltstyps. Methoden, die den Stereotyp *view* oder *form* besitzen, werden hingegen als Page Template mit einer dazugehörigen Action generiert (z.B. *project_overview* in der Klasse `Project`). Methoden, die als Stereotyp *action* besitzen, werden nur als Actions generiert.
- **Attribute** werden als Feld in der Schemadefinition generiert.
- **Interfaces** werden als Python-Interface-Deklarationen generiert. Klassen, die eine Realisationsbeziehung zu einem Interface besitzen, bekommen das in der Zope-Entwicklung übliche `__implements__`-Statement verpasst.
- **Vererbung**: Generalisierungsbeziehungen zwischen Klassen werden auch in Python als Vererbungsbeziehung generiert; Mehrfachvererbung wird unterstützt. Beispiel: `Resource` und `Person` bzw. `Room`.
- **Aggregation und Composite**: Diese beiden Beziehungen drücken sich in ordnerartigen Objekten aus. Die Klasse, die andere enthalten kann, wird als ordnerartiges Objekt generiert, das den Clientteil der Beziehung enthalten kann (mit Hilfe von *allowed_content_types*). Beispiel: *Projekt-Resource* für eine Aggregation oder *Projekt-Task* für ein Composite.
- **Assoziationen** werden als *ReferenceFields* generiert, wobei hier zwischen den Kardinalitäten *0..1* und *0..n* unterschieden wird. Beispiel: *Projekt-Person* oder *Task-Resource*.
- **Zustandsmodelle (Statemachines)** werden als Workflow-Definition generiert, die automatisch beim Installieren des Produkts im Workflow-Tool eingerichtet und der entsprechenden Klasse zugeordnet wird. Beispiel: Das Zustandsdiagramm der Klasse `Project`.

Das ist alles über das Rapid Development, was im Rahmen dieses Buches angeführt werden kann. Es gibt noch eine Reihe von Optionen in ArchGenXML. Viele Feineinstellungen können über Stereotypen und so genannte *Tagged Values* sowie Kommandozeilenparameter eingestellt werden. Eine umfangreiche und ständig wachsende Online-Dokumentation wurde von Jens Klein initiiert und ist auf *plone.org* unter *http://plone.org/documentation/tutorial/archgenxml-getting-started* zu finden.

13.2.6 Daten in einer SQL-Datenbank speichern

Für fast alles, was in diesem Buch gezeigt wurde, sind die Daten in der *Zope Object Database* (ZODB) gespeichert worden. Ich habe gezeigt, wie Daten im Dateisystem gespeichert werden, aber eine häufig gestellte Frage ist, wie Daten in einer relationalen Datenbank gespeichert werden können. Eine Speicherung in einer relationalen Datenbank lohnt sich vor allem dann, wenn folgende Tatsachen zutreffen:

- Ihre Datenstrukturen sind starr, und das Schema ändert sich selten (obwohl Archetypes diesen Aspekt abschwächt).
- Sie haben andere Applikationen, die Zugriff auf die relationale Datenbank haben oder benötigen.
- Sie haben Tools und andere Anforderungen, die bereits von einer relationalen Datenbank erfüllt werden.
- Sie haben eine große Menge an Daten, die oft verändert werden.

In einer eher traditionellen CGI-Umgebung werden Sie wahrscheinlich einige SQL-Statements schreiben, um die Daten aus einer relationalen Datenbank zu beziehen. Ähnliches erreichen Sie in Zope durch die Verwendung von ZSQL-Methoden. Diese sind in vielen Situationen nützlich und werden sehr ausführlich im Online-Zope-Buch auf *zope.org* und in einigen anderen Büchern behandelt, aber sie bieten kein richtiges objektrelationales (OR) Mapping. ZSQL-Methoden speichern als Template das SQL-Statement und führen Abfragen aus, indem dieses SQL-Statement an die Datenbank abgesetzt wird. Dies ist recht nützlich, um einfache Abfragen durchzuführen, aber es eignet sich nicht, um ganze Objekte transparent in einer relationalen Datenbank abzuspeichern, und entspricht somit nicht der Gedankenwelt von Plone.

`SQLStorage` macht sich die Tatsache zunutze, dass Archetypes eine eigene konfigurierbare Schicht für das Speichern von Attributen bereithält. Für Zope gibt es eine Menge von Datenbankadaptern, die `SQLStorage` verwendet, um Attribute abzuspeichern. Derzeit sind es zwei Datenbankadapter, MySQL und Postgres, die von `SQLStorage` direkt unterstützt werden. Allerdings ist es mit vertretbarem Aufwand verbunden, `SQLStorage` auch an andere Datenbankadapter anzupassen, da hier hauptsächlich einige SQL-Statements zum Anlegen von Tabellen und Spalten ausgetauscht werden müssten. Aber wie immer steckt der Teufel auch hier sicher im Detail.

Wie mache ich meine Daten in einer relationalen Datenbank persistent?

Um Ihre Daten in einer relationalen Datenbank persistent zu machen, müssen Sie eine Zope-Datenbankverbindung einrichten. In diesem Fall verwende ich Postgres, weil ich Erfahrung damit habe und es allgemein eine universelle Datenbank

ist, die für alle Schichten von Anwendern geeignet ist. Um Zope mit Postgres zu verbinden, ist das Gespann aus *psycopg* und *ZPsycopgDA* die richtige Wahl. Sie finden Pakete für Debian und Windows unter *http://initd.org/software/psycopg*.

Installieren Sie Ihren Datenbankadapter, und richten Sie eine Datenbankverbindung zu Ihrer relationalen Datenbank ein. Dies ist, wie gesagt, im Zope-Buch im Detail unter *http://zope.org/Documentation/Books/ZopeBook/2_6Edition/RelationalDatabases.stx* beschrieben. Aber die kurze Antwort ist, dass Sie das ZMI öffnen und von der Liste der verfügbaren Objekte eine Datenbankverbindung auswählen und anschließend die Verbindungsparameter eingeben. Stellen Sie sicher, dass der Benutzer, als der Sie sich bei der Datenbank anmelden, die Berechtigung besitzt, Tabellen anzulegen, zu verändern und zu löschen (`create table`, `alter table` und `drop table`). Jene Berechtigungen benötigen Sie nur während der Entwicklungs- und Installationsphase, da `SQLStorage` automatisch die Tabellen und ihre Spalten anlegt. Im Betrieb, wenn sich am Schema nichts mehr ändert, werden diese Berechtigungen nicht mehr benötigt.

Anschließend klicken Sie im ZMI auf ARCHETYPES_TOOL in Ihrer Plone-Instanz und wählen CONNECTIONS. Dies erlaubt Ihnen, die Inhaltstypen einzelnen Datenbankverbindungen zuzuordnen. Der einfachste und in den meisten Fällen sinnvollste Weg ist es, die Defaultverbindung anzugeben, die dann für alle Inhaltstypen gilt, die noch nicht zugeordnet sind. Nun müssen Sie im Schema den einzelnen Feldern die neue Storage zuordnen, wofür Sie die passende Storage-Klasse von `SQLStorage` importieren (in diesem Fall ist dies `PostgreSQLStorage`).

Dann müssen Sie den `storage`-Parameter bei den Feldern angeben, die Sie in der Datenbank speichern wollen. Sie werden vielleicht bemerkt haben, dass Sie die Felder einzelnen verschiedenen Datenbanken zuordnen können. Sie können alle Felder in die relationale Datenbank speichern oder auch nur einzelne. Aber auch, wenn Sie alle Felder der relationalen Datenbank zuordnen, wird immer der Kopf des Objekts mit den Attributen `id` und `uid` in der ZODB abgespeichert. Im vorliegenden Beispiel werden zwei Felder in der SQL-Datenbank gespeichert: eine Ganzzahl (`age`) und ein String (`E-Mail`), wie in Listing 13.4 gezeigt wird:

Listing 13.4: Ein Inhaltstyp, der SQLStorage verwendet

```
from Products.Archetypes.public import BaseSchema, Schema
from Products.Archetypes.public import BaseContent, registerType
from Products.Archetypes.public import IntegerField, StringField
from Products.Archetypes.public import IntegerWidget, StringWidget
from Products.Archetypes.SQLStorage import PostgreSQLStorage
from config import PROJECTNAME

schema = BaseSchema + Schema((
```

```
  IntegerField('age',
      validators=("isInt",),
      storage = PostgreSQLStorage(),
      widget=IntegerWidget(label="Your age"),

      ),

  StringField('E-Mail',
      validators = ("isEmail",),
      index = "TextIndex",
      storage = PostgresSQLStorage(),
      widget = StringWidget(label='E-Mail',)
      ),

    ))

class PersonSQL(BaseContent):
    """Our person object"""
    schema = schema

registerType(PersonSQL, PROJECTNAME)
```

Schließlich starten Sie Plone neu und registrieren Ihr Produkt in Ihrem Plone-Portal. Nun können Sie Ihren Inhaltstyp testen. Wenn Sie ein `PersonSQL`-Objekt erzeugen, verhält sich auf der Oberfläche alles ganz normal. Der wirkliche Test findet statt, wenn Sie sich die Datenbank ansehen.

Sie werden sehen, dass in der Datenbank eine neue Tabelle namens `personsql` angelegt wurde und dass diese Tabelle vier Spalten enthält: `uid`, `parentuid`, `age` und `E-Mail`. Im Kommandozeilen-Client von Postgres *psqk* sollte sich dann Folgendes zeigen:

```
db=# \d
                 List of relations
 Schema |           Name           |   Type   | Owner
--------+--------------------------+----------+-------
 public | personsql                | table    | www-data
...
db=# \d personsql
  Table "public.personsql"
  Column   | Type | Modifiers
-----------+------+-----------
 uid       | text | not null
 parentuid | text |
 age       | int  |
 E-Mail    | text |
Indexes: personsql_pkey primary key btree (uid)
```

Die Spalte für `age` wurde als `int` und die Spalte für `E-Mail` wurde mit dem Datentyp `text` angelegt. Innerhalb von `SQLStorage` existiert eine Zuordnung von Archetypes-Feldtypen zu SQL-Datentypen. Diese Zuordnung kann bei Bedarf auch angepasst werden. Die `uid`-Spalte entspricht der eindeutigen ID `UID` in Plone, die bereits weiter oben in diesem Kapitel behandelt wurde. Die Spalte `parentuid` enthält die eindeutige ID des übergeordneten Objekts.

```
db=# SELECT * FROM personsql;
          uid              | parentuid | age | E-Mail
---------------------------+-----------+-----+------------------
 PersonSQL.2003-07-23.4935 |           | 30  | andy@clearwind.ca
```

Das war's, Ihre Daten sind persistent innerhalb Ihrer relationalen Datenbank. Kein SQL muss manuell geschrieben werden, und Sie können trotzdem einige Vorteile von relationalen Datenbanken nutzen. Joel Burton hat eine vorzügliche Anleitung zu `SQLStorage` geschrieben, die Sie unter *http://plone.sourceforge.net/archetypes/sqlstorage-howto.html* finden. Mit Joels freundlichen Erlaubnis basieren Teile dieses Abschnitts auf seinem Dokument.

Ein anderer Ansatz ist, eine ganze Plone-Instanz in einer relationalen Datenbank abzuspeichern und direkt an der Persistenzschicht der ZODB anzusetzen. Dieser Ansatz wird von der *Adaptable Persistence Engine*, kurz *APE*, verfolgt, die von der Zope Corporation entwickelt wurde und nun in der Version 1.0 vorliegt. Sie können sie unter folgender Adresse herunterladen: *http://hathawaymix.org/Software/Ape*. APE ermöglicht es, einen ganzen Teil der ZODB in einer relationalen Datenbank abzuspeichern, und speichert somit alle Objekte unabhängig von deren Klasse in diese Datenbank. Ein Artikel auf *plone.org* beschäftigt sich mit der Integration von APE und Archetypes: *http://plone.org/events/sprints/castlesprint/wiki/ApeSupport*. Zurzeit gibt es noch keine praktische Erfahrung mit Plone und APE, um es für den produktiven Einsatz zu empfehlen, doch kann sich das in absehbarer Zeit ändern.

14 Administration und Skalierung von Plone

In diesem Kapitel geht es um Aufgaben, um die Sie sich kümmern müssen, nachdem Sie Ihre Site aufgebaut haben und sie benutzen. Ich beginne mit der Administration einer Plone-Site, die prinzipiell ziemlich einfach ist. Danach beschreibe ich, wann und von welchen Dateien Sicherheitskopien angelegt werden sollten. Außerdem behandle ich Aktualisierungen von Plone.

Anschließend geht es um Performance-Fragen und Techniken zum Auffinden von Schwachstellen. Nachdem Sie diese gefunden haben, beschreibe ich häufig auftretende Probleme. Dann vertiefe ich das Thema Caching als wesentliche Technik dafür, Ihre Plone-Site wirklich schnell und skalierbar zu machen. Wenn es um Performance geht, müssen Sie definitiv wissen, wie Ihr Server mit mehreren Prozessen nach außen skalieren kann und wie man rechenintensive Anfragen behandelt. Dazu benötigen Sie *Zope Enterprise Objects* (ZEO), was am Ende dieses Kapitels behandelt wird.

14.1 Administration einer Plone-Site

Wie sich herausstellt, ist die Administration einer Plone-Site ziemlich einfach: Sie müssen lediglich ein paar Aufgaben erledigen, die für alle Dienste gleich sind. Gemeint ist Folgendes:

- Sie sollten regelmäßig ein Backup von der Datenbank machen.
- Sie sollten die Datenbank regelmäßig komprimieren.
- Sie sollten rotierende Backups der Protokolldateien anlegen.

Diese Aufgaben sollten Sie regelmäßig bei der Wartung Ihrer Website erledigen. In einer Firma haben Sie oftmals standardisierte Werkzeuge für Backups und rotierende Protokolldateien. Diese Werkzeuge lassen sich alle leicht integrieren, da Plone-Daten alle in Form von Dateien im Dateisystem vorliegen.

14.1.1 Backups Ihrer Plone-Site durchführen

Von einer Plone-Site sollten Sie regelmäßig Backups anfertigen. Die meisten Leute erstellen Backups in der Nacht. Ein Backup-Plan sollte aus den Anforderungen Ihrer Anwendung abgeleitet werden. Wenn große Datenmengen geschrieben werden, sind vielleicht häufigere Backups notwendig. Bei einer kleinen Site mit weniger Inhalten mögen weniger häufige Backups, z.B. einmal pro Woche, besser geeignet sein.

Bei einer normalen Plone-Site muss nur von einer Datei ein Backup erstellt werden: von der Zope-Datenbank, in der alle Inhalte der Plone-Site stecken. Diese Datei finden Sie, indem Sie auf das Control Panel im Zope Management Interface (ZMI) zugreifen, dort DATABASE MANAGEMENT wählen und dann auf MAIN klicken. Diese Seite zeigt die Größe und den Ort der Datenbank an. Die Datei heißt `Data.fs`, und Sie finden sie im Verzeichnis `var` Ihrer Instanzwurzel. Um das Backup auszuführen, kopieren Sie diese Datei lokal. Dann können Sie sie sicher auf einen externen Speicher kopieren, z.B. auf eine andere Festplatte, einen Server, ein Bandlaufwerk oder was Sie auch immer sonst für ein Backup-System haben. Dieses Backup können Sie sogar dann durchführen, während Plone läuft.

Zum Anlegen von Backups können Sie Ihre eigenen Scripten und Werkzeuge oder ein Werkzeug aus Zope benutzen. Als Beispiel für die erste Möglichkeit zeigt Listing 14.1 ein Linux-Bash-Script, das ich für das Backup einer Zope-Site benutze.

Listing 14.1: Bash-Script für ein Backup

```
#!/bin/bash
# script to copy, gzip, and then copy Zope databases
# to remote server
# make up a filename
fn=`uuidgen`.fs
# copy it locally, you'll want to change the
# path
cp /var/zope.test/var/Data.fs /tmp/$fn
# gzip the file up
gzip /tmp/$fn
# scp over to my backup server and then remove
# the temporary file
# change the destination file
scp /tmp/$fn.gz backup@backups.agmweb.ca:~/Zope
rm /tmp/$fn.gz
```

Als zweite Möglichkeit steht Ihnen ein Python-Script namens `repozo.py` in der Zope-Objektdatenbank (ZODB) zum Erstellen von Backups zur Verfügung. Sie

finden dieses Script online unter *http://cvs.zope.org/ZODB3/Tools/repozo.py*. Es funktioniert ziemlich gut unter Windows und Linux und kann eine ganze Menge, z.B. vollständige und inkrementelle Backups erstellen und die Datenbank wiederherstellen.

Um mit diesem Script ein Backup einer Datenbank anzufertigen, müssen Sie zuerst ein Verzeichnis anlegen, das das Backup aufnimmt. In den folgenden Beispielen lautet dieses Verzeichnis `/home/backups`, aber Sie haben die freie Wahl dabei. Führen Sie Folgendes aus, um ein vollständiges Backup einer Datenbank zu erstellen:

```
$ python repozo.py -B -F -v -r /home/backups -f /var/zope.test/var/Data.fs
looking for files b/w last full backup and 2003-11-21-18-33-17...
no files found
doing a full backup
writing full backup: 3601549 bytes to /home/backups/2003-11-21-18-33-17.fs
```

Um ein inkrementelles Backup anzufertigen, lassen Sie einfach die Option `-F` (für engl. *full*) weg. Das Script vergleicht dann die aktuelle ZODB mit dem letzten Backup und kopiert lediglich die Unterschiede. Wenn keine Aktualisierungen stattgefunden haben, erfolgt kein Backup. Folgendes ist ein Beispiel-Backup nach einer Änderung in Plone:

```
$ python repozo.py -B -v -r /home/backups -f /var/zope.test/var/Data.fs
looking for files b/w last full backup and 2003-11-21-18-39-09...
files needed to recover state as of 2003-11-21-18-39-09:
        /home/backups/2003-11-21-18-33-17.fs
repository state: 3601549 bytes, md5: ab9e46bcdf52641ad6f71db62a9da333
current state   : 3624968 bytes, md5: 73c871bbe2528e152342abea9e25ab27
backed up state : 3601549 bytes, md5: ab9e46bcdf52641ad6f71db62a9da333
doing incremental, starting at: 3601549
writing incremental: 23419 bytes to /home/backups/2003-11-21-18-39-11.deltafs
```

An diesem Punkt haben Sie ein vollständiges und ein inkrementelles Backup. Dasselbe Script kann diese Daten nun auch wiederherstellen. Dazu geben Sie die Option `-R` (für engl. *recovery*) und mit `-o` die Ausgabedatei wie folgt an:

```
$ python repozo.py -R -v -r /home/backups -o /var/zope.test/var/Data.fs
looking for files b/w last full backup and 2003-11-21-18-50-21...
files needed to recover state as of 2003-11-21-18-50-21:
 /home/backups/2003-11-21-18-33-17.fs
 /home/backups/2003-11-21-18-39-11.deltafs
Recovering file to /var/zope.test/var/Data.fs
Recovered 3624968 bytes, md5: 73c871bbe2528e152342abea9e25ab27
```

Eine vollständige Liste aller Optionen erhalten Sie, wenn Sie `repozo.py` mit `-h` ausführen, was alle vorhandenen Optionen ausgibt.

Die Protokolldateien befinden sich standardmäßig im `log`-Verzeichnis Ihrer Instanzwurzel. Dort gibt es zwei Dateien: eine für Zugriffe und eine für Ereignisse. Den Ort dieser Protokolldatei können Sie in der Konfigurationsdatei angeben, die Sie in Kapitel 2 kennen gelernt haben. In `z2.log` werden alle ankommenden Anfragen protokolliert und in `event.log` alle Fehler. Von diesen Dateien sollte regelmäßig ein Backup gemacht werden, zusammen mit eventuell vorhandenen Protokolldateien auf Proxy-Servern, z.B. solchen, die Apache oder IIS (Internet Information Services) produzieren.

Von Code, Templates und eigenen Produkten außerhalb der ZODB sollten Sie ebenfalls regelmäßige Backups anfertigen. Auch dann, wenn Sie diese in einem Versionsverwaltungsprogramm wie CVS (Concurrent Versioning System) aufbewahren, kann es nicht schaden, einen gültigen Schnappschuss Ihrer Installation zu machen.

Wenn Sie Inhalte, andere Datenbanken oder andere Daten haben, die nicht in der ZODB sind, sollten auch diese Teil des Backup-Plans sein, je nachdem, wie oft sie sich ändern. Das könnten Daten in relationalen Datenbanken und Inhalte im Dateisystem sein. All diese Daten werden vom Site-Entwickler erzeugt und kommen in einer normalen »taufrischen« Plone-Site nicht vor. Wenn Sie Zope oder Plone aktualisieren, kann es sinnvoll sein, ein Backup aller beteiligten Dateien zu machen, inklusive Zope und Plone, damit bei einer womöglich fehlerhaften Aktualisierung eine vollständige Wiederherstellung möglich ist.

14.1.2 Die ZODB komprimieren

Die ZODB speichert alle Änderungen an allen Objekten im System. Jedes Mal, wenn sich ein Objekt ändert, wird eine neue Kopie ans Ende der ZODB-Datei angehängt. Dies ist die Datei `Data.fs`, die ich im vorigen Abschnitt beschrieben habe. Falls die Datenbank große Teilinhalte oder eine große Anzahl von Änderungen enthält, kann das dazu führen, dass die ZODB merklich anwächst.

Eine große ZODB als solche ist kein Problem, auch sie funktioniert einwandfrei, und die Zeiten beim Hochfahren sind vergleichbar (es sei denn, der Index wurde entfernt). Die Zeiten für die Komprimierung werden länger, je größer die Datenbank ist. Daher macht es Sinn, diese alten Kopien von Objekten, die nicht mehr benötigt werden, gelegentlich zu entfernen, um die Datenbank kleiner zu machen. Es ist ganz wesentlich zu wissen, dass Sie beim Komprimieren lediglich Ihre vorhandene Datenbank säubern und einige alte Kopien wegwerfen.

Exkurs: Die alte 2-Gbyte-Grenze bei Datenbanken

Mit älteren Versionen von Python (vor Python 2.1 unter Unix und vor Python 2.2 unter Windows) gibt es ein Problem, da diese große Dateien nicht unterstützen. Wenn die ZODB dann zwei Gigabyte erreicht, bleibt die Plone-Site stehen und kann nicht neu gestartet werden. Um zu testen, ob Sie eine Python-Version verwenden, die große Dateien unterstützt, öffnen Sie eine Datei in Python und sehen wie folgt nach, ob deren Größe als Integer oder als Long angegeben wird:

```
>>> import os
>>> from stat import ST_SIZE
>>> type(os.stat('/tmp/test.txt')[ST_SIZE])
<type 'long'>
```

Diese Python-Version unterstützt große Dateien und solche, die über 2 Gbyte groß sind. Falls ein Integer zurückgegeben wird, dann müssen Sie Ihre Python-Version aktualisieren oder mit Unterstützung für große Dateien neu kompilieren (in neueren Versionen ist diese Unterstützung automatisch vorhanden). Wenn Sie versuchen, Plone mit einer Zope-Version zu kompilieren, die große Dateien nicht unterstützt, erhalten Sie folgenden Fehler:

```
andy@thorin:/tmp/Zope-2.7.0-b3$ ./configure
Configuring Zope installation
...
This Python interpreter does not have have 'large file support'
enabled.
```

Wenn das der Fall ist, dann müssen Sie Ihre Python-Installation auf Vordermann bringen. Weitere Details dazu finden Sie unter *http://www.python.org/doc/current/lib/posix-large-files.html*. Wenn Ihnen die Beschränkung auf 2 Gbyte nichts ausmacht, können Sie beim Konfigurationsscript die Option `--ignore-largefile` angeben. Mit einer Beschränkung auf eine 2 Gbyte große Datenbank werden Sie häufiger eine Komprimierung vornehmen müssen.

Komprimieren heißt in der Datenbank aufräumen

Eine Komprimierung kann aufwendig sein, und wenn dieser Prozess ausgeführt wird, passiert das in einem separaten Thread, d.h., auch wenn die Geschwindigkeit einer Site dadurch beeinflusst wird, kann sie weiterhin auf Anfragen reagieren. Wie Sie Sites komprimieren, während Plone trotzdem mit maximaler Perfor-

mance läuft, wird im Abschnitt »ZEO-Clients benutzen« später in diesem Kapitel erklärt. Um eine Komprimierung durchzuführen, gehen Sie ins Control Panel des ZMI, wählen DATABASE MANAGEMENT und klicken auf MAIN.

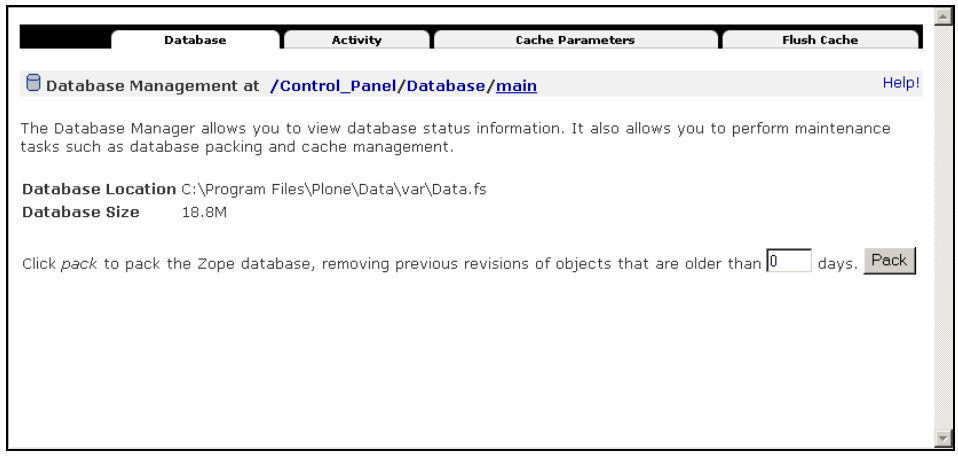


Abbildung 14.1: Komprimieren einer Datenbank

Geben Sie die Anzahl von Tagen an, für die Sie Objekte behalten möchten, und klicken Sie auf PACK. Wenn Sie z.B. die Anzahl von Tagen auf null setzen (die Voreinstellung), werden alle Versionen der Objekte entfernt. Noch einmal: Nicht das Objekt selbst wird gelöscht, sondern nur diese alten Kopien davon. Eine üblichere Einstellung ist ein Wert um sieben herum, wobei dann Versionen gelöscht werden, die älter als eine Woche sind. Durch die Wahl einer mit Ihrem Backup-Plan verträglichen Einstellung können Sie garantieren, dass Sie von jedem Objekt eine Kopie behalten. Die Komprimierung wird je nach Größe Ihrer ZODB einige Zeit und Verarbeitungskapazität in Anspruch nehmen. Plone läuft währenddessen weiter, wenn auch etwas langsamer, d.h., eventuell möchten Sie ZEO dafür benutzen.

14.1.3 Plone aktualisieren

Plone wird kontinuierlich weiterentwickelt und verbessert, d.h., es kommen ziemlich regelmäßig neue Versionen von Plone heraus. Bevor Sie auf eine neue Version von Plone aktualisieren, sollten Sie aber nachprüfen, ob Sie diese wirklich brauchen. Zwischen den einzelnen Releases gibt es recht häufig nur kleine, möglicherweise wenig relevante Änderungen. Zu jedem Release gibt es eine Liste von Änderungen auf der Seite, wo man es herunterladen kann. Es empfiehlt sich immer ein Blick auf diese Liste, um zu sehen, ob sich die Aktualisierung lohnt.

Machen Sie zuerst ein Backup, und laden Sie dann die Aktualisierung herunter. Am leichtesten führen Sie die Aktualisierung vermutlich durch, indem Sie die gleichen Schritte wie bei der Installation wiederholen. Wenn Sie die Installation mit dem Windows-Installationsprogramm durchgeführt haben, laden Sie das neue Installationsprogramm herunter und wiederholen die Installation. Wenn Sie eine Installation aus den Quellen oder einem Debian-Paket durchgeführt haben, wiederholen Sie diese Schritte. Die Aktualisierung umfasst folgende Schritte :

1. Die entsprechende Aktualisierung herunterladen
2. Plone anhalten
3. Ein Backup durchführen (wie zuvor beschrieben)
4. Die Aktualisierung installieren
5. Plone starten

An dieser Stelle möchte ich empfehlen, Plone im Debug-Modus zu starten. Unter Windows können Sie das tun, indem Sie START – PLONE – PLONE DEBUG wählen. Unter Linux machen Sie das mit dem Script `runzope` im `bin`-Verzeichnis Ihrer Instanzwurzel wie folgt:

```
bin/runzope -X "debug-mode=on"
```

Durch die Ausführung im Debug-Modus können Sie direkt eventuelle Fehler sehen, die bei der Aktualisierung auf die neue Version aufgetreten sind. Wenn Sie damit zufrieden sind, können Sie mit dem nächsten Schritt, der Migration, weitermachen.

Gehen Sie bei all Ihren Plone-Sites ins ZMI, und greifen Sie auf das Werkzeug `portal_migration` in Ihrer Plone-Site zu. Es wird ein hellrotes Ausrufezeichen daneben haben, was angibt, das die Site nicht auf dem neusten Stand ist. Nun müssen Sie eine Migration von Plone auf die aktuelle Version vornehmen. Eine Migration ist notwendig, da Sie das Dateisystem aktualisiert haben, aber Ihre Datenbank nicht mit dem Dateisystem synchronisiert ist. Es könnte Änderungen an den Inhaltstypen, den Werkzeugen oder anderen Teilen Ihrer Site geben, die vorgenommen werden müssen.

Die Migration wird versuchen, diese Änderungen für Sie durchzuführen. Solange Sie diese Migration nicht durchführen, könnte es sein, dass Ihre Plone-Site defekt ist. Je nachdem, was bei der Migration gemacht werden muss, kann das einige Zeit brauchen. Befolgen Sie diese Schritte, um eine Migration durchzuführen:

1. Klicken Sie unter PORTAL_MIGRATION auf den MIGRATE-Reiter.
2. Klicken Sie auf den UPGRADE-Button. Dieser Upgrade-Vorgang kann eine Weile dauern, besonders bei großen Sites oder wenn eine große Aktualisierung notwendig ist.

3. Das Ergebnis der Migration, eine ziemlich lange Meldung, wird auf dem Bildschirm angezeigt. Die Migration war erfolgreich, wenn die letzte Meldung »End of upgrade path, migration has finished« lautet. Eventuelle Fehlermeldungen werden in Rot hervorgehoben.

Wiederholen Sie diesen Vorgang für alle Plone-Sites in Ihrer Zope-Instanz. Wenn Sie mit der migrierten Site zufrieden sind, halten Sie das im Debug-Modus laufende Plone an. Starten Sie Plone wie üblich neu, und machen Sie ganz normal weiter.

14.2 Die Performance von Plone verbessern

Nun haben Sie also eine wunderbare Website, Millionen von Besuchern schauen vorbei, aber sie zeigt nicht ganz die gewünschte Performance. Nun, Plone wurde von vornherein als System entworfen, das reich an Möglichkeiten, aber nicht unbedingt sehr schnell ist, weil die Geschwindigkeit sehr stark von der jeweiligen Anwendung abhängt. Man kann Plone allerdings mit vielen Techniken sehr schnell machen, und es skaliert sehr leicht. In den folgenden Abschnitten behandle ich, wie Sie die langsamen Teile Ihrer Site ausfindig machen, und zeige Ihnen dann Methoden, wie Sie sie verbessern können.

14.2.1 Benchmarks einer Plone-Site erstellen

Bevor Sie versuchen, eine Site zu optimieren, besteht die wesentliche Aufgabe darin, sich einen Zahlenwert für die Performance der Site zu verschaffen. Die Benutzer geben oftmals Feedback der Art »sie ist zu langsam« oder »es braucht zu lange, bis sie geladen wird«. Für einen Entwickler sind diese Kommentare nahezu nutzlos. Sie müssen die Geschwindigkeit quantitativ angeben können, um zu wissen, wie schnell die Site jetzt ist und wie schnell sie gemacht werden muss. Erst dann können Sie mit der Optimierung beginnen.

Um einen Benchmark zu erhalten, können Sie ein Werkzeug namens *ab* bzw. Apache Bench benutzen. Dieses Werkzeug ist Teil des Apache-Servers. Falls Sie Apache 1.3 oder höher unter Linux installiert haben, ist *ab* bereits enthalten. Unter Windows ist es ab der Version 2 von Apache enthalten. Man kann *ab* sehr einfach ausführen: Geben Sie einfach die URL (Uniform Resource Locator) an, die Sie testen möchten, z.B. so:

```
ab http://localhost/
```

Das Werkzeug *ab* gibt dann wie folgt zuerst einige Daten über die getestete Site aus:

```
Benchmarking localhost (be patient).....done
Server Software:        Zope/(unreleased
Server Hostname:        localhost
Server Port:            80

Document Path:          /
Document Length:        20594 bytes
```

Anschließend gibt es wie folgt einige zusammenfassende Statistiken aus:

```
Concurrency Level:      1
Time taken for tests:   0.771151 seconds
Complete requests:      1
Failed requests:        0
Write errors:           0
Total transferred:      20933 bytes
HTML transferred:       20594 bytes
Requests per second:    1.30 [#/sec] (mean)
Time per request:       771.151 [ms] (mean)
Time per request:       771.151 [ms] (mean, across all concurrent requests)
Transfer rate:          25.94 [Kbytes/sec] received
```

Das sagt Ihnen, wie lange die Anfrage bearbeitet wurde, wie viele Fehler aufgetreten sind, und wie lang es dauerte, bis eine Anfrage abgearbeitet wurde, was vermutlich die wichtigste Angabe ist. Der nützlichste Wert, den man angeben kann, ist normalerweise `Requests per second`, d.h. Anfragen pro Sekunde – in diesem Beispiel also `1.30 [#/sec]`. Weitere Statistiken, die das Werkzeug *ab* bietet, geben Informationen darüber an, wie lange es dauert, bis eine Verbindung erstellt wird, wie lange sie bearbeitet wird bzw. wie lange es braucht, bis ein Ergebnis für eine Anfrage vorliegt. Beispiel:

```
Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:   770  770   0.0    770     770
Waiting:      766  766   0.0    766     766
Total:        770  770   0.0    770     770
```

Diese letzte Information ist hilfreich und beinhaltet die Zeit für den Aufbau einer Verbindung. Da mein Server auf dem gleichen Rechner wie der Client läuft, ist sie hier ziemlich kurz. Dieser Test zeigt, dass es 1,30 Sekunden dauert, eine Anfrage abzuarbeiten. Der Server wurde dabei natürlich so gut wie gar nicht getestet. Um die reale Welt besser zu simulieren, möchten Sie den Server beim

Testen vermutlich mit einigen Anfragen gleichzeitig belasten. Das können Sie tun, indem Sie die Anzahl der Anfragen insgesamt mit der Option `-n` und die Anzahl der gleichzeitigen Anfragen (Thread) mit `-c` angeben. Beispiel:

```
ab -n 20 -c 4 http://localhost/
```

Hierbei werden insgesamt 20 Anfragen über vier parallele Threads geschickt. Das Endergebnis ist eine leicht andere Anzahl von Anfragen pro Sekunde, nämlich 1,78. Weitere Angaben zu allen verfügbaren Optionen lesen Sie bitte im Handbuch zu Apache Bench unter *http://httpd.apache.org/docs/programs/ab.html* nach.

Einer der Vorteile von *ab* besteht darin, dass Sie nicht wirklich die Seiten auf dem Client zusammensetzen. Stattdessen werden sie nur heruntergeladen und dann weggeworfen. Wenn Sie eine Seite mit einer Menge an Scripten oder mit großen Bildern haben, wird die Zeit nicht mitgerechnet, die ein Client braucht, um diese Seite zu etwas zusammenzusetzen, was für den Benutzer Sinn macht. Ein klassisches Beispiel hierfür ist der alte Netscape-Browser, der durch eine große Zahl von Tabellen stark verlangsamt wird oder sogar abstürzen kann. Mit *ab* wäre das nicht festzustellen, denn es gibt Ihnen eine unabhängigere Zahl, mit der Sie arbeiten können.

Lügen, schlimme Lügen und Benchmark-Zahlen

An dieser Stelle sind Sie vermutlich über diese Zahlen besorgt. Sie scheinen für eine sehr langsame Site zu sprechen. Der Rechner in diesen Beispielen ist mein Toshiba-Laptop mit einem 1,8-GHz-Celeron-Prozessor, 256 Mbyte RAM, Red Hat Linux 9.0 und einer Beta-Version von Plone 2. Außerdem läuft Plone im Debug-Modus gleichzeitig mit KDE, OpenOffice.org, Instant Messenger und mehreren anderen Entwicklerwerkzeugen, inklusive des eigentlichen Benchmark-Werkzeugs. Das heißt, weder ist Plone irgendwie optimiert, noch läuft es in einer idealen Umgebung. Ein ähnlicher Test auf einem schnelleren Server ergab einen Wert von etwa 20 Anfragen pro Sekunde.

Der wesentliche Punkt ist, dass Sie mit Hilfe einer objektiven Zahl für die Site-Performance den Erfolg Ihrer Optimierungen messen können. Entwickler können Tricks und Kniffe anwenden und diese testen, um die Zahlen »davor« und »danach« zu vergleichen. Wenn möglich, sollten Sie die Performance-Tests auf einem Rechner durchführen, der möglichst ähnlich zu dem Server in Produktion ist, um vernünftige Zahlen zu erhalten. In diesem Kapitel ist es nicht wichtig, dass die Site X Anfragen pro Sekunde schafft, sondern es ist wichtig, dass eine Änderung eine signifikante Verbesserung der Performance bewirkt.

Denken Sie außerdem daran, dass Zahlen darüber, wie schnell ein bestimmter Teil Ihrer Site ist, für sich allein genommen ziemlich nutzlos sind. Sie müssen verschiedene andere Dinge ebenfalls bedenken, z.B. wie oft die Seite besucht wird, die Erwartungen der Benutzer sowie realistische Anforderungen. Die penibel

genaue Messung eines Teils einer Site kann nützlich sein, um ein bestimmtes Problem zu finden, macht Ihre Site aber eventuell kaum schneller. Wie bei den meisten Dingen brauchen Sie auch hier einen vernünftigen Ansatz bei Ihrer Optimierung.

Produktions- vs. Debug-Modus

Eines der Dinge, die am meisten Geschwindigkeit fressen, ist, wenn Sie Plone im Debug-Modus betreiben. Im Debug-Modus werden alle Templates, Scripten und Objekte im Werkzeug `portal_skins` mit dem Dateisystem verglichen, um zu überprüfen, ob sie aktuell sind. Diese Überprüfung erfolgt bei jeder einzelnen Anfrage, was beim Testen hilfreich ist, aber einen riesigen Performance-Einbruch bedeutet. Auf meinem Testserver z.B. erhöht sich nach dem Abschalten des Debug-Modes die Geschwindigkeit von 1,30 auf 2,41 Anfragen pro Sekunde, eine signifikante Verbesserung.

Um herauszufinden, ob Ihre Site im Debug-Modus läuft, gehen Sie im ZMI zum Objekt `portal_migration` Ihrer Plone-Site. Am unteren Rand dieser Seite steht eine Liste von Informationen, darunter auch der Status des Debug-Modus. Um ihn zu ändern, ändern Sie die Konfigurationsdatei, wie in Kapitel 2 beschrieben.

Andere Gründe für geringe Performance

Ein Server kann aus verschiedenen Gründen, die nichts mit Plone zu tun haben, langsam laufen. Wenn Sie eine Optimierung durchführen, sollten Sie immer zuerst einen Blick darauf werfen, da Sie hierdurch mit kleinem Aufwand schnell Verbesserungen bei der Geschwindigkeit erreichen können.

Prozessorauslastung

Wenn Sie eine große Anzahl von Anwendungen laufen haben oder nur wenige sehr aufwendige, wird die für Plone verfügbare Prozessorzeit dadurch beschränkt. Das Zusammensetzen von Seiten in Plone kann eine Menge CPU-Zeit in Anspruch nehmen. Eine Anwendung, die durch die verfügbare CPU-Zeit beschränkt ist, heißt auch *CPU-beschränkt*.

Um herauszufinden, unter welcher Last ein Server unter Linux ist, benutzen Sie den Befehl *top*. Unter Windows erhalten Sie eine ähnliche Statistik vom Task Manager (den Sie mit der Tastenkombination Strg+Alt+Entf erreichen). Die empfohlene Geschwindigkeit für Ihre CPU hängt von der Größe und Last des Verkehrs auf Ihrem Plone-Server ab, aber ein 2-GHz-Prozessor ist eine gute Ausgangsbasis.

Größe des Hauptspeichers

Zope benutzt gern einen großen Teil des Hauptspeichers, wenn Objekte aus der ZODB geladen werden. Von allen wesentlichen Eigenschaften eines Zope-Ser-

vers hat ein größerer Hauptspeicher vermutlich den größten Effekt. Eine Anwendung, die durch den verfügbaren Hauptspeicher beschränkt ist, heißt auch *speicherbeschränkt*. Sehr wahrscheinlich wird Ihr Server den Geist aufgeben, wenn er auf diese Grenze stößt, während der Swap-Speicher auf die Festplatte ausgelagert wird.

Um herauszufinden, unter welcher Last ein Server unter Linux ist, benutzen Sie den Befehl *top*. Unter Windows erhalten Sie eine ähnliche Statistik vom Task Manager (den Sie mit der Tastenkombination Strg+Alt+Entf erreichen). Die empfohlene Größe des Hauptspeichers hängt von der Größe und Last des Verkehrs auf Ihrem Plone-Server ab, aber 512 Mbyte sind ein guter Anfang. Wenn Sie sich mehr leisten können, empfiehlt sich auch mehr.

An den Hauptspeicherparametern können Sie in Plone etwas drehen, indem Sie die Anzahl der Objekte im Cache erhöhen. Per Voreinstellung hat Plone 400 Objekte im Cache. Das könnten Sie bei einer Site auf 5000 erhöhen, wie in Abbildung 14.2 gezeigt ist. Obwohl sich der Hauptspeicherverbrauch erhöht, steigt dabei aber *auch* die Performance. Die so gewonnene Performance kann man jedoch unmöglich vorhersagen, weil Objekte naturgemäß unterschiedlich groß sind. Der beste Ansatz ist vermutlich der, diesen Wert heraufzusetzen und dann einige Anfragen zu testen, um zu sehen, ob sich dieser Kompromiss lohnt.

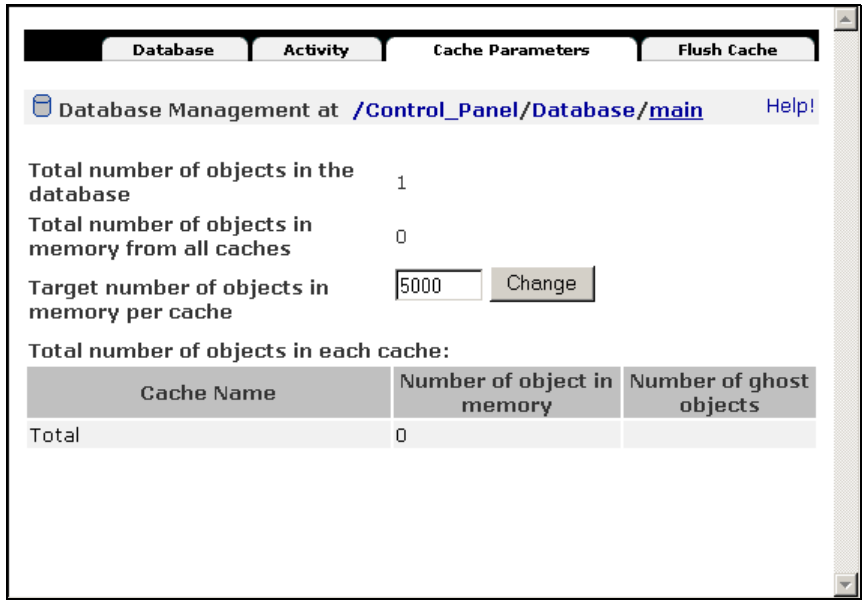


Abbildung 14.2: Die Cache-Parameter im Control Panel ändern

Je weniger Threads Zope außerdem benutzt, desto weniger Hauptspeicher wird verbraucht. Zwar ist Zope multithread-fähig, aber meistens wird nur ein Zope-Thread wirklich benutzt. Eine Reduktion der Thread-Anzahl auf drei ergibt einen speichereffizienteren Server. Anstatt zu versuchen, eine große Zahl von Threads laufen zu lassen, wird empfohlen, mit Hilfe von ZEO-Clients mehr Anfragen zu bedienen. Der Abschnitt »Zope Enterprise Objects verwenden« weiter unten behandelt das etwas detaillierter.

Netzwerkverbindung

Die Netzwerkverbindung kann von entscheidender Bedeutung für die Performance einer Anwendung sein. Sie können nicht schneller sein als die langsamste Verbindung zwischen Ihnen und dem Client. Beachten Sie beim Optimieren einer Plone-Site daher auch die Verbindungszeit. Wenn eine Verbindung zwei Sekunden braucht, ist die Optimierung des Codes ziemlich sinnlos.

Auch hierbei kann das Werkzeug *ab* helfen. Wenn Sie einen Benchmark-Test von Plone.org von Britisch-Kolumbien, Kanada, aus durchführen (der Server befindet sich in Texas), können Sie in der folgenden Ausgabe sehen, dass der Median der Wartezeit für Verbindungen über das Netzwerk 125 Millisekunden betrug:

```
            Connection Times (ms)
            min  mean[+/-sd] median   max
Connect:       90  133  40.2    125     211
Processing:   511 1103 400.2   1113    1846
Waiting:      202  310 110.3    293     565
Total:        601 1236 411.2   1211    2043
```

Eventuell hat der Server auch eine Beschränkung bezüglich der Anzahl der Verbindungen oder bei der Reise durch interne Firewalls. Wenn ein Prozess derart durch die Zeit beschränkt ist, die er für eine Input/Output-Operation benötigt (I/O), heißt er auch *I/O-beschränkt*. Was Sie an dieser Stelle machen können, hängt davon ab, über welche Möglichkeiten Sie im Netzwerk verfügen. Wenn Ihr Client sehr weit von Ihrem Server entfernt ist und die Verbindung sehr langsam ist, dann könnten Caches in der Nähe des Clients eine geeignete Möglichkeit sein.

Ihre Anwendung

Es könnte natürlich auch sein, dass Ihre Anwendung Ursache für die geringe Performance ist. Es gibt zahlreiche (und vermutlich übertriebene) Beispiele von Dienstleistungsfirmen für Kunden mit Problemen. Zu den bekannteren Beispielen gehören die folgenden:

- Kopierter Code von einer Website, der einen tief im System verborgenen *sleep*-Aufruf enthielt, wodurch das Script einige Sekunden pausiert. Eine von jemandem durchgeführte Code-Inspektion hat das zu Tage gebracht, und die Zeile wurde entfernt.

- Mehrfache Abfragen in relationalen Datenbanken, z.B. über ein Dutzend auf einer Seite. Mit einem intelligenteren Design wurden die Abfragen kombiniert, was auch ein Caching erlaubte.
- Ein Script, das Informationen aus der ZODB herausholte, indem alle Objekte in der Datenbank aufgeweckt wurden. Durch die Benutzung des Katalogs (siehe Kapitel 11) wurde die Performance viel besser.
- Eine Abfrage, die alle Datensätze einer Datenbank holt, dann aber nur jeweils 100 auf einer Seite anzeigt und die anderen 99900 wegwirft. Das wurde mit effizienteren SQL-Anweisungen gelöst.

Bevor Sie voreilige Schlussfolgerungen über die Ursache des Problems ziehen, lohnt es sich, Laufzeitmessungen auf der Site vorzunehmen, um festzustellen, wo das Nadelöhr liegt.

14.2.2 Laufzeitmessungen mit Plone

Da Sie die Zeit, die zum Erzeugen von Seiten benötigt wird, quantitativ bestimmen können, können Sie nun versuchen zu optimieren. Das erste Problem besteht allerdings darin, die zu optimierende Stelle zu finden. Wie in früheren Kapiteln gezeigt wurde, ist die Hauptseite eine Sammlung von Templates, Scripten und anderem Code, der zusammengesetzt wird, um eine Seite zu erstellen. Wo sollte ein Entwickler in diesem Berg von Code nachschauen? Um bei der Suche nach Performance-Schwachstellen zu helfen, können Sie so genannte Profiler für Laufzeitmessungen von Aufrufen, Page Templates und Python selbst benutzen. Jeder dieser Profiler kümmert sich um ein Element der Site und berichtet, wie viel Zeit beim Erstellen der Seite und bei jeder dieser drei Komponenten verbraucht wurde.

Beachten Sie bitte, dass Sie beim Aktivieren aller drei Werkzeuge feststellen werden, dass Ihre Plone-Site wirklich langsam wird (und zwar um eine signifikante Größenordnung). Jedes dieser Werkzeuge bringt einen Verlust an Performance mit sich. Sie sollten deren Installation immer rückgängig machen oder sie deaktivieren, nachdem Sie sie benutzt haben, um sicherzugehen, dass Ihre Site mit maximaler Effizienz läuft. Außerdem gilt beim gleichzeitigen Einsatz aller Profiler, dass Sie dann auch die Laufzeit dieser Profiler mitmessen, was dann wirklich verwirrend wird. Ich empfehle Ihnen, mit dem Aufruf-Profiler zu beginnen. Schalten Sie dann so lange jeweils einen der anderen Profiler ein, nachdem Sie den vorherigen deaktiviert haben, bis Sie genügend Informationen haben.

Call Profiler

Dieses Zope-Produkt nimmt eine Anfrage, z.B. nach der Hauptseite, und gibt an, welche Objekte bei ihrer Abarbeitung benutzt wurden und wie viel Zeit jedes

davon benötigt hat. Den Call Profiler finden Sie unter *http://zope.org/Members/richard/CallProfiler*. Trotz der Kommentare auf der Seite ist das Produkt nicht in Zope 2.6 integriert. Installieren Sie das Produkt auf die übliche Weise, und starten Sie dann Ihr Zope neu.

Um den Call Profiler zu aktivieren, gehen Sie im ZMI zum Control Panel und wählen CALL PROFILER. Das Produkt funktioniert derart, dass es Hooks in ein Objekt installiert, damit beim Zugriff auf das Objekt die Zeit für die Wiedergabe des Objekts gemessen werden kann. Das heißt, dass Call Profiler nur bei den Objekten aktiviert wird, die Sie überwachen möchten, wie in Abbildung 14.3 zu sehen ist. Bei einer normalen Plone-Installation müssen Sie das Filesystem Script (Python)- und das Filesystem-Page Template überwachen. Der Call Profiler merkt sich diese Einstellungen nicht nach einem Neustart von Zope, d.h., dass bei einem einfachen Neustart die Hooks deaktiviert werden und Sie bereit für die Auslieferung sind.

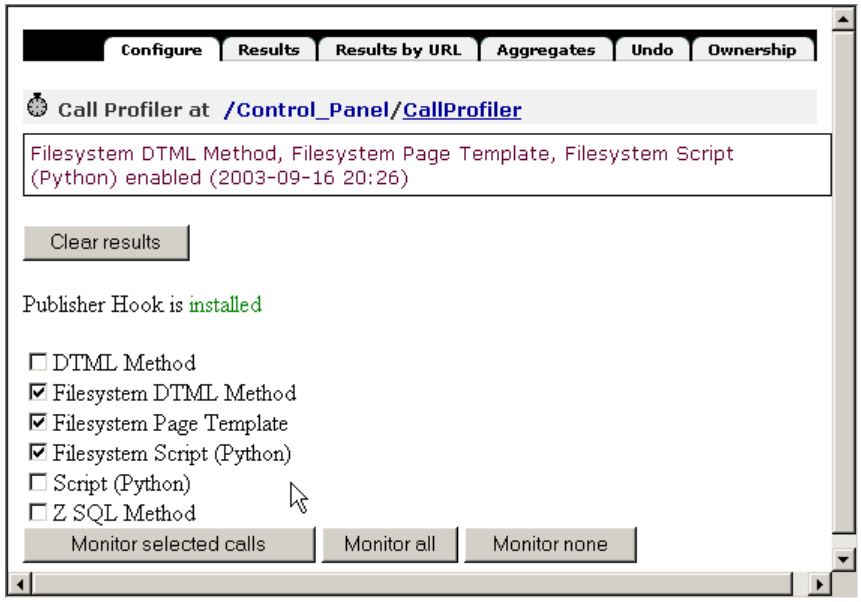
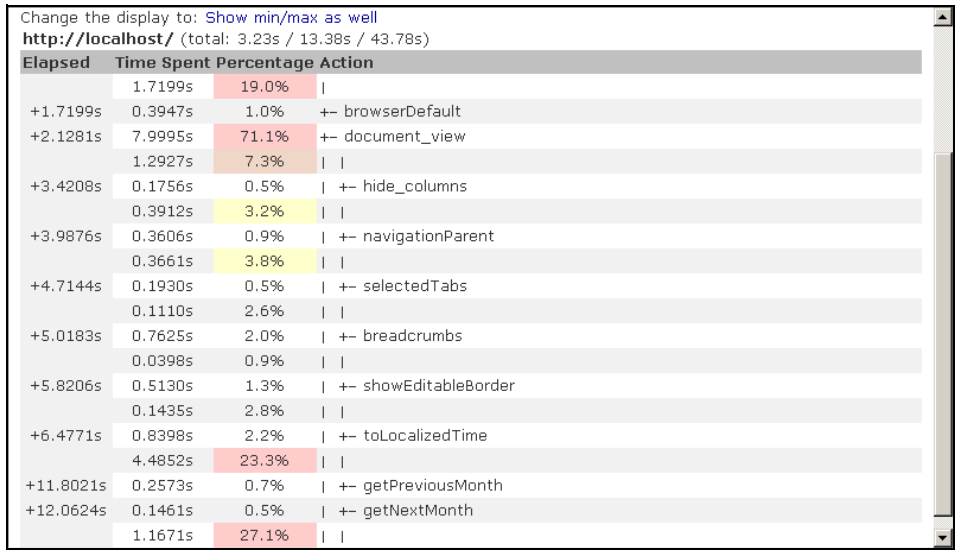


Abbildung 14.3: Call Profiler mit ausgewählten Dateisystem-Hooks

Nachdem die zu überwachenden Objekte ausgewählt wurden, greifen Sie auf die gewünschte URL zu. Am einfachsten greift man auf die zu testende URL zu, indem man das zuvor erwähnte Werkzeug *ab* ausführt, aber einen Webbrowser zu benutzen funktioniert genauso gut. Wenn Sie in diesem Fall die Hauptseite auf *localhost* messen, starten Sie Folgendes:

```
ab -n 20 -c 4 http://localhost/
```

Hierdurch werden 20 Anfragen an Plone gesendet. Sobald sie abgearbeitet sind, können Sie auf die Messungen dieser Anfragen zugreifen. Zurück in der Schnittstelle von Call Profiler sehen Sie oben drei Reiter: RESULTS, RESULTS BY URL und AGGREGATES. Da mehrere Anfragen ausgeführt wurden, wählen Sie den AGGREGATES-Reiter, der am leichtesten zu verstehen ist. In der Liste der aufgerufenen Seiten wird die getestete URL sein. Klicken Sie auf diesen Link, um die Ergebnisse zu dieser URL zu sehen. Nun sollten Sie etwas wie in Abbildung 14.4 sehen.



Change the display to: Show min/max as well

http://localhost/ (total: 3.23s / 13.38s / 43.78s)

Elapsed	Time Spent	Percentage	Action
	1.7199s	19.0%	|
+1.7199s	0.3947s	1.0%	+- browserDefault
+2.1281s	7.9995s	71.1%	+- document_view
	1.2927s	7.3%	| |
+3.4208s	0.1756s	0.5%	| +- hide_columns
	0.3912s	3.2%	| |
+3.9876s	0.3606s	0.9%	| +- navigationParent
	0.3661s	3.8%	| |
+4.7144s	0.1930s	0.5%	| +- selectedTabs
	0.1110s	2.6%	| |
+5.0183s	0.7625s	2.0%	| +- breadcrumbs
	0.0398s	0.9%	| |
+5.8206s	0.5130s	1.3%	| +- showEditableBorder
	0.1435s	2.8%	| |
+6.4771s	0.8398s	2.2%	| +- toLocalizedTime
	4.4852s	23.3%	| |
+11.8021s	0.2573s	0.7%	| +- getPreviousMonth
+12.0624s	0.1461s	0.5%	| +- getNextMonth
	1.1671s	27.1%	| |

Abbildung 14.4: Ergebnisse der Laufzeitmessung

In diesem Beispiel sehen Sie die Elemente, die der Call Profiler erkennen kann. Leider sind die Ergebnisse manchmal etwas kompliziert zu entziffern. Auf den ersten Blick addieren sich die Ergebnisse zu mehr als 100 %. In diesem Fall nimmt `document_view` 71,1 % der verbrauchten Zeit ein. Das ist allerdings irreführend, weil sich die Werte unter dieser Abbildung auf `document_view` und nicht auf die ganze Seite beziehen. In diesem Beispiel für die gesamte Seite nimmt alles vor `browserDefault` 19,9 % der Anfrage in Anspruch. Dann geht sie zu `document_view` über, wofür Sie auch den Prozentanteil sehen können. In diesem Fall macht der Übergang von `toLocalizedTime` nach `getPreviousMonth` 23,3 % der Zeit zur Darstellung von `document_view` aus.

Page Template Profiler

Der Page Template Profiler funktioniert nur mit dem Page Templates-System von Zope. Ähnlich wie der Call Profiler gibt er an, wie viel Zeit bei jedem Aufruf in einem Page Template verbraucht wird. Da Sie im vorigen Beispiel gesehen haben, dass die meiste Zeit in einem Page Template verbraucht wird

(`document_view`), finden Sie es vermutlich interessant zu wissen, wo die Zeit in diesem Template genau verbraucht wird.

Den Page Template Profiler finden Sie unter *http://zope.org/Members/guido_w/PTProfiler*. Installieren Sie das Produkt, und starten Sie dann Zope neu. Um die Installation rückgängig zu machen, müssen Sie den Page Template Profiler aus Ihrem `Products`-Verzeichnis entfernen, wenn Sie mit der Messung fertig sind.

Gehen Sie nach der Installation im ZMI zur Zope-Wurzel, und wählen Sie PT PROFILE VIEWER im Dropdown-Menü ADD. Füllen Sie das ERSTELLEN-Formular aus, indem Sie als ID einen eindeutigen Wert eingeben (geben Sie z.B. *PTProfiler* ein), und klicken Sie dann auf ADD. Wiederholen Sie nun den Aufruf der zu messenden Seite, indem Sie das Werkzeug *ab* starten oder in einem Browser auf die Seite zugreifen. Greifen Sie auf das gerade hinzugefügte Page Template Profiler-Objekt zu, und Sie werden ein Ergebnis für die gerade ausgeführte Anfrage sehen. Klicken Sie darauf, um weitere Details zu sehen (vergleiche Abbildung 14.5).

Expression	Total time	Number of calls	Time per call
Total rendering time	**44.618**	**23**	**1.93991**
python: path(pathexpr)	18.3256	184	0.0996
path: here/calendar_slot/macros/calendarBox	16.8383	23	0.7321
path: here/global_elements/macros/skin_tabs	3.9568	23	0.17203
python: request.get('news', here.portal_catalog.searchResults (meta_type='News Item' , sort_on='Date' , sort_order='reverse' , review_state='published')[:5])	2.1732	23	0.09449
python: here.portal_actions.listFilteredActionsFor(here)	1.8614	23	0.08093
python: here.breadcrumbs(here)	0.8098	23	0.03521
python: here.toLocalizedTime(here.ModificationDate (),long_format=1)	0.6545	23	0.02846
path: options/slots_mapping|here/prepare_slots|nothing	0.6096	23	0.02651
python: here.showEditableBorder(template_id=template_id , actions=actions)	0.5987	23	0.02603
path: here/navigationLocalRelated	0.5436	23	0.02363
python: here.plone_utils.createNavigationTreeBuilder (portalObject,navBatchStart)	0.5375	23	0.02337

Abbildung 14.5: Ergebnisse des Page Template Profilers

In diesem Fall sehen Sie, dass auf meiner Site `calendarBox` jedes Mal 0,7321 Sekunden benötigt, wenn es aufgerufen wird. Da die gesamte Seite 1,9 Sekunden benötigt, können Sie davon ausgehen, dass dies ein Bereich ist, den ich optimieren könnte.

Python Profiler

Der Python Profiler bietet sehr fein aufgelöste Messwerte und wird normalerweise bei der komplexeren Fehlersuche im darunter liegenden Code verwendet. Er liefert einen detaillierten Bericht der Zeiten, die in verschiedenen Bereichen des Python-Code verbracht werden. Bei den Messungen einer Site würden Sie so etwas normalerweise nicht benutzen. Aber der Vollständigkeit halber möchte ich es in diesem Abschnitt beschreiben.

Um den Python Profiler zu aktivieren, müssen Sie eine Variable zur Konfigurationsdatei hinzufügen. Schalten Sie in der Datei `zope.conf` Ihres `etc`-Verzeichnisses den Befehl `publisher-profile-file` ein. Dazu definieren Sie eine Datei, in die er schreiben wird. Unter Windows könnte das `C:\zope.output` sein, unter Linux ist es `/tmp/zope.output`. Fügen Sie die folgende Zeile unter Linux hinzu:

```
publisher-profile-file /tmp/zope.output
```

Dann starten Sie Plone neu, das dann sehr langsam laufen wird. Wenn Sie eine große Anzahl von Anfragen abarbeiten und die Ergebnisse untersuchen möchten, dann enthält die in der Umgebungsvariable angegebene Datei die Ausgabe der Daten. Rufen Sie, wie in den vorherigen Beispielen, die zu messende Seite mit dem Werkzeug *ab* oder einem Webbrowser auf. Dann gehen Sie im ZMI zum Control Panel, wählen DEBUG INFO und wählen dann den PROFILING-Reiter. Sie erhalten dann eine Ausgabe des Python Profilers, wie sie in Abbildung 14.6 zu sehen ist.

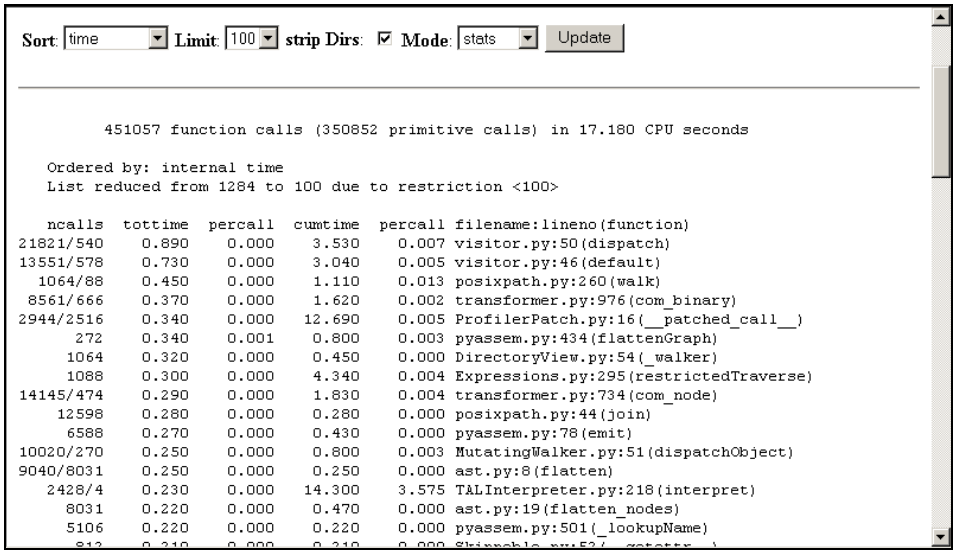


Abbildung 14.6: Ergebnisse des Python Profilers

Wie Sie in Abbildung 14.6 sehen können, zeigt die Ausgabe Details von allem, was Zeit verbraucht. Ich musste das nur selten benutzen.

14.2.3 Einfache Optimierungstricks

Nach einem sehr detaillierten Blick auf Plone hat das Plone-Entwicklerteam die folgenden Optimierungstricks gefunden.

Die Namenssuche einschränken

Die übertriebene Suche nach Namen ist ein häufiger Fehler, dessen Lösung in der Definition einer lokalen Variable besteht. Im folgenden Beispiel muss Plone bei jeder Wiederholung der Schleife eine Suche nach `portal_url` durchführen:

```
<tal:block
 tal:repeat="result here/portal_catalog">
   <a href=""
      tal:attributes="href here/portal_url/getPortalUrl">Home</a>
   ...
</tal:block>
```

Dabei wäre es schneller, ein `tal:define` wie folgt zu benutzen:

```
<tal:block
 tal:repeat="result here/portal_catalog"
 tal:define="url here/portal_url/getPortalUrl>
    <a href=""
       tal:attributes="href url">Home</a>
       ...
</tal:block>
```

Wie schon gesagt, Plone bietet eine große Zahl von globalen Defines. Durch die Verwendung dieser Definitionen kann ein Entwickler die Anzahl der Traversierungen reduzieren. Eine vollständige Liste all dieser Defines finden Sie in Anhang A.

Sicherheitsüberprüfungen und Traversierung

Immer, wenn auf ein Objekt, ein Objektattribut oder eine Methode eines Objekts zugegriffen wird, wird eine Sicherheitsüberprüfung durchgeführt. Auch, wenn jede einzelne nicht so kostspielig ist, können sich viele Sicherheitsüberprüfungen schnell aufaddieren.

Das gilt besonders dann, wenn Sie ein Objekt traversieren, z.B. in `here/ordnerA/ordnerB/objekt`. In diesem Fall führt Zope Sicherheitsüberprüfungen auf allen einzelnen Ordnern und dann auf dem Objekt durch. Wenn auf die Information jedes Mal ohne diese Traversierung zugegriffen werden kann, werden Sie

einen Performance-Gewinn feststellen. Eine andere Methode, Sicherheitsüberprüfungen zu vermeiden, besteht darin, Code im Dateisystem unter `Products` zu schreiben. Der Code unter `Products` wird als vertrauenswürdiger Code betrachtet, wird weniger geprüft und daher schneller ausgeführt.

Der ZCatalog

Der ZCatalog ist ein effizienter Binärbaum mit Daten über Objekte. Sie sollten ihn (in den meisten Situationen) dann benutzen, wenn Sie eine Liste von Objekten erstellen, z.B. Suchergebnisse, bei Zusammenfassungen, beim Finden von Objekten usw. Wenn der Katalog eine Reihe von Ergebnissen zurückgibt, die auf eine Liste von leichtgewichtigen Objekten (so genannten Brains) zugreifen, findet beim Zugriff auf diese Brains weder eine Traversierung noch eine Sicherheitsüberprüfung auf dem Objekt statt.

Zu viele Features

Das mag offensichtlich erscheinen, aber Plone wird mit einer Menge von Features ausgeliefert, die Sie nicht unbedingt benötigen. So benötigen z.B. sowohl das Kalender- als auch das Navigations-Portlet beide große Ressourcen, haben aber einen allgemein eher beschränkten Nutzen. Wenn Sie diese Features nicht benötigen und sie abschalten, steigt dadurch die Performance.

Lohnt sich die Optimierung?

Bevor Sie irgendeine Optimierung durchführen, sollten Sie eine einfache Kosten/Nutzen-Analyse erstellen, um zu sehen, ob es sich lohnt, die Optimierung vorzunehmen.

Angenommen, Sie haben z.B. eine Seite, die in 0,5 Sekunden erstellt wird. Auf dieser Seite verbraucht ein Script 10 % dieser Zeit. Wenn Sie dieses eine Script doppelt so schnell ausführen können, gewinnen Sie dadurch 0,025 Sekunden bei dieser Seite. In diesem Fall ist der Nutzen einer solchen Optimierung gering, weil es einige fixe Kosten gibt, z.B. die eines Entwicklers, der die Analyse durchführt, die Kosten beim Testen dieser Optimierung und möglicherweise die einer Änderung der Dokumentation.

Diese Arbeiten durchzuführen bedeutet auch ein substanzielles Risiko. Beim Ändern von Code können neue Fehler in der Anwendung entstehen. Mit Hilfe von Methoden wie agiler Programmierung könnten diese aber minimiert werden. Außerdem schafft es ein Programmierer vielleicht nicht, die Anwendung schneller zu machen, oder macht sie sogar langsamer.

Es gibt auch Alternativen zur Optimierung des Codes, z.B. könnten Sie mehr Hauptspeicher oder Hardware installieren, falls die Anwendung durch einen dieser Faktoren beschränkt wird. Auch wenn viele Programmierer der Meinung sind, dass es faul ist, Probleme durch mehr Hardware zu lösen, kann das eine

sehr kostengünstige Lösung sein. Neue Hardware stellt ein geringes Risiko dar, kann einen hohen Geschwindigkeitszuwachs bringen und kostet oftmals weniger als ein Programmierer.

Außerdem können Sie Ihren Server wirklich durch Caching oder das Hinzufügen weiterer Rechner skalieren, auf die Sie die Last verteilen. Diese Techniken werden im Rest dieses Kapitels behandelt.

14.2.4 Inhalte cachen

Nun, da Sie die langsamen Teile Ihrer Anwendung gefunden haben, werden Sie das wichtigste Werkzeug zur Steigerung der Performance kennen lernen: das Caching. Unter Caching versteht man die Zwischenspeicherung von Daten, die wieder benutzt werden sollen, ohne sie neu zu berechnen. Es gibt viele, viele Arten von Cache-Speichern und sie können auf vielerlei Weisen benutzt werden.

Mit dem Begriff Caching meine ich zwei Dinge, die man cachen kann: Inhalte und Skins. Inhalte sind Daten, die der Benutzer in Inhaltstypen eingegeben hat. Skins beziehen sich auf alles in `portal_skins` und können Templates, Scripten, Bilder oder Dateien sein. Diese zwei Dinge werden im Cache unterschiedlich gespeichert.

Ich denke beim Caching gern in Begriffen, die mit dem Grad an Einflussmöglichkeit zu tun haben, die ich über den Caching-Mechanismus habe. Mit anderen Worten: Je näher das Caching beim Client durchgeführt wird, desto schneller wird die Antwort sein, aber desto weniger Einfluss habe ich auch auf diesen Cache. Das beinhaltet auch die Möglichkeit, dass es überhaupt keinen Cache-Speicher gibt. Abbildung 14.7 illustriert das Caching zwischen einem Client und einem Server.

Der Browser-Cache des Benutzers ist der schnellste Platz, an dem man etwas cachen kann, aber Sie können nicht wissen, ob ein Benutzer in seinem Browser das Caching überhaupt eingeschaltet hat. Danach kommen die zwischengelagerten Cache-Speicher von Proxy-Servern. Denken Sie daran, dass das Ihr Proxy-Server sein könnte (der unter Ihrer Kontrolle sein sollte) oder der Proxy eines Internet Service Providers (ISP). Und schließlich gibt es die Möglichkeit von Cache-Speichern auf dem Server.

In den folgenden Abschnitten erörtere ich die folgenden Caching-Mechanismen:

- Caching von Skin-Elementen mit dem *Accelerated HTTP Cache Manager*
- Caching von Code mit dem *RAM Cache Manager*
- Caching von benutzererstellten Inhalten mit dem *Caching Policy Manager*

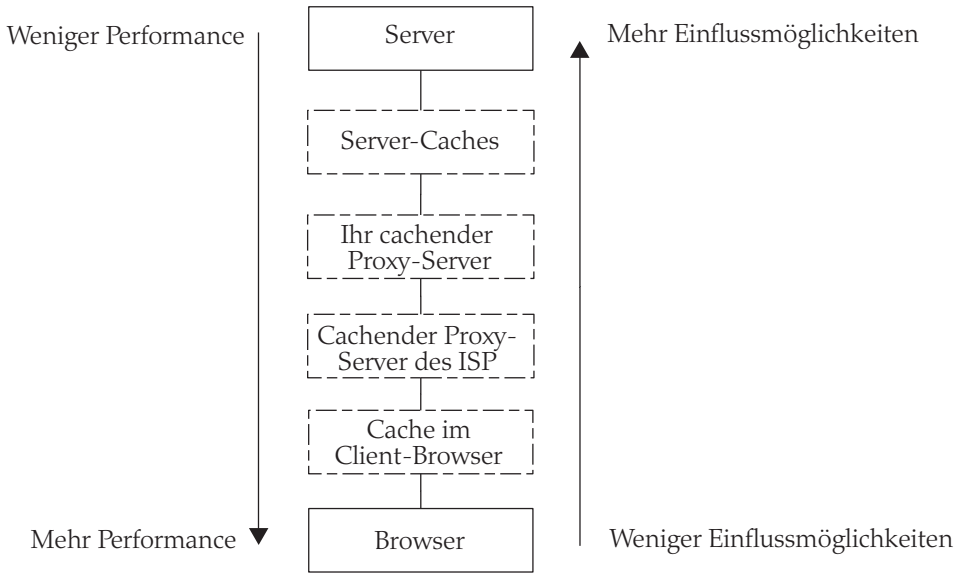


Abbildung 14.7: Cache-Speicher zwischen einem Client und einem Server

Dann werde ich erläutern, wie man Apache und Squid benutzt, zwei gern verwendete externe Server, die über eine Menge an Konfigurationsoptionen für eine hohe Performance verfügen.

14.2.5 Skins cachen

Im Hypertext Transfer Protocol (HTTP) können Sie HTTP-Header für das Caching setzen. Wenn eine Antwort mit solchen Headern zurückgegeben wird, ist es die Aufgabe der Proxies zwischen dem Client und dem Server, das Objekt gemäß dieser Header zu cachen. In Abbildung 14.7 könnte das ein beliebiger Cache-Speicher ab dem Server-Cache oder darunter sein. Dieser Proxy kann ein Webserver sein, über den Sie auf dem Server verfügen, z.B. Apache, oder ein Proxy im Einflussbereich des ISPs. Wie ich noch zeigen werde, ergibt das ein mächtiges Werkzeug, wenn es mit Apache oder Squid kombiniert wird.

Zu diesem Caching kann auch der Browser beitragen, wenn er so eingestellt ist, dass er Caching verwendet (beim Internet Explorer ist das standardmäßig der Fall). Wenn ein Browser allerdings eine Seite aktualisiert, sendet er den Header `Pragma: no-cache`, was auch die Proxies zwingt, ihre Kopie erneut zu laden.

Diese Art von Caching betrifft die gesamte Antwort, d.h., es kann riskant sein, wenn Sie versuchen, das auf eine ganze Seite anzuwenden. Am meisten wird das mit Bildern, Stylesheets, JavaScript oder Seiten benutzt, die sich nicht oft verän-

dern. Bilder, die auf Ihren Seiten oft wiederholt werden, um hübsche Elemente zu erzeugen, z.B. abgerundete Ecken oder Hintergrundbilder, sind hierfür ideal.

Plone erzeugt standardmäßig einen Accelerated HTTP Cache Manager namens *HTTPCache* in der Wurzel Ihrer Plone-Site. Beim Zugriff auf dieses Objekt über das ZMI erscheinen die Management-Optionen für den Cache-Speicher. Das Folgende sind alles vernünftige Voreinstellungen, und zu Beginn muss nichts verändert werden:

- **Title**: Der Titel des Cache Managers, optional.
- **Interval**: Die Zeit (in Sekunden), die das Objekt im Cache bleibt.
- **Cache anonymous connections only**: Hiermit wird nur bei anonymen Benutzern gecacht.
- **Notify URLs**: Das sind URLs von Proxies zum Herunterladen, die geleert werden müssen, wenn sich das Objekt verändert. Weitere Details dazu finden Sie im Abschnitt »Squid verwenden«.

Um zu sehen, wie der Accelerated HTTP Cache Manager funktioniert, basiert das folgende Beispiel auf einem Testobjekt, einem Bild namens `test.gif`. Um zu sehen, welche Header zurückgegeben werden, müssen Sie diese Header testen. Dazu können Sie ein einfaches Python-Script namens `header.py` benutzen. Dieses Script finden Sie in Anhang B. Unter Linux macht das Programm *wget* das Gleiche, wenn Sie die Option `-S` angeben, obwohl es die Datei immer noch für Sie herunterlädt. Beispiel:

```
wget -S http://www.agmweb.ca
```

Es folgen zuerst einmal die Header, die für `test.gif` zurückgegeben werden, *bevor* Sie das zum Cache-Manager hinzugefügt haben:

```
[andy@basil scripts]$ ./header.py http://localhost:8080/test.gif GET
Accept-Ranges: bytes
Connection: close
Content-Length: 2541
Content-Type: image/gif
Date: Wed, 03 Sep 2003 23:55:38 GMT
Etag:
Last-Modified: Wed, 03 Sep 2003 23:54:27 GMT
Server: Zope/(unreleased version, python 2.2.2, linux2) ZServer/1.1
```

Nach dem Hinzufügen des Bildes zum Cache prüfen Sie die HTTP-Header mit dem Script erneut. Sie werden zwei neue Header finden. Beispiel:

```
[andy@basil scripts]$ ./header.py http://localhost:8080/test.gif GET
...
```

```
Cache-Control: max-age=3600
Expires: Thu, 04 Sep 2003 00:56:03 GMT on 2.2.2, linux2) Zserver/1.1
```

Hinweis

Leider hält sich Zope 2 nicht an das RFC (Request for Comments) für `HEAD`-Anfragen. Anstatt bei einer `HEAD`-Anfrage den vollen Satz an Headern zu schicken, fehlen die Werte vom Cache Manager. Beim Testen sollten Sie immer `GET`-Anfragen schicken.

Weitere Informationen zu den HTTP-Headern und den Zusammenhang mit dem Caching finden Sie im RFC 2616 unter *http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html*.

Der Accelerated HTTP Cache Manager speichert eine ganze Antwort im Cache, was mit statischen Elementen gut funktioniert. Eine normale Plone-Seite besteht aber aus personalisierten Elementen, z.B. dem Kalender, der persönlichen Navigationsleiste usw. In dieser Situation müssen Sie nur einen Teil einer Seite cachen können, und hier kommt der RAM Cache Manager ins Spiel.

Der RAM Cache Manager speichert die Ausgabe eines Objekts im RAM, d.h., dass es beim nächsten Auftreten dieses Scripts aus dem Cache geholt wird. Wiederholte Aufrufe dieses Objekts führen dazu, dass die Ausgabe so lange aus dem Cache geholt wird, bis der Cache abläuft. Der springende Punkt bei diesem Manager ist der, dass Sie es vermeiden, jedes Mal komplizierte oder umfangreiche Berechnungen durchzuführen. Stattdessen speichern Sie das Ergebnis und benutzen es erneut. Dieser Cache Manager speichert keine Bilder oder Dateien. Er hält die Benutzer nicht vom Versuch ab, den Cache dahingehend zu konfigurieren, aber er hat keinen Effekt auf diese Objekte.

Plone erstellt standardmäßig einen RAM Manager namens *RAMCache* in der Wurzel Ihrer Plone-Site. Beim Zugriff auf dieses Objekt über das ZMI werden die Management-Optionen für den Cache aufgerufen. Die folgenden Standardwerte sind alle vernünftig, und zu Beginn muss nichts geändert werden:

- **Title**: Der Titel des Cache Managers, optional.
- **REQUEST variables**: Variablen, die die Cache-Bedingung ausmachen. Das ist eine mächtige Option, die die Bedingungen für das Zwischenspeichern von Benutzervariablen darstellt. Wenn z.B. ein zu cachendes Element verlangt, dass es für jeden Benutzer anders oder in verschiedenen Sprachen gecacht werden soll, können Sie hier die REQUEST-Variablen angeben, die Sie cachen möchten.

- **Threshold entries**: Die maximale Anzahl von Einträgen, die im Cache gespeichert werden kann. Verringern Sie diesen Wert, wenn der Cache zu viel RAM verbraucht.
- **Maximum age of a cache entry**: Die Zeit (in Sekunden), die das Objekt im Cache bleibt.
- **Cleanup interval**: Gibt an, wie oft der Cache gesäubert wird.

Da die Anfragen nach dem Objekt bis zu Zope durchkommen, reduziert sich der Netzwerkverkehr dadurch keineswegs. Zope stellt das Ergebnis lediglich schneller dar. Nach der Wahl des STATISTICS-Reiters im ZMI wird eine Statistik über die genaue vom Cache zurückgegebene Trefferanzahl ausgegeben und darüber, wie viele Zugriffe an das Objekt weitergegeben wurden. Wenn zu viele Zugriffe ans Objekt weitergereicht werden, möchten Sie eventuell die Cache-Konfiguration ändern und weniger `REQUEST`-Variablen angeben oder die Verweilzeit im Cache erhöhen.

Caches zuweisen

Um ein Objekt im Dateisystem zum Cache hinzuzufügen, geben Sie einfach für dieses Objekt den Namen des Caches in der `.metadata`-Datei an (`.metadata`-Dateien wurden in Kapitel 11 erläutert). Plone macht das schon für eine Reihe von Bildern, bei CSS und JavaScript. Die Datei `plone_skins /plone_images/pdf_icon.gif.metadata` lautet z.B. wie folgt:

```
[default]
title=Pdf icon
cache=HTTPCache
```

Das bedeutet, dass das Bild mit *HTTPCache* gecacht wird. Die meisten Objekte im Dateisystem eignen sich mehr für den *HTTPCache* als für den *RAMCache*.

14.2.6 Inhaltstypen cachen

Das Cachen von Inhaltstypen ist ein wenig schwieriger und verlangt den Einsatz des *Caching Policy Managers*. Plone installiert dieses Werkzeug standardmäßig, und Sie finden es in der Wurzel der Plone-Instanz unter der ID `caching_policy_manager`.

Bevor Sie irgendeinen Inhalt cachen können, müssen Sie die Cache-Einstellungen für die Templates in Plone ändern. Per Voreinstellung gibt Plone Header für Inhalte aus, die das Caching völlig ausschalten. Wenn Sie Folgendes nicht machen, funktioniert der Rest dieses Abschnitts nicht. Wenn Sie PORTAL_SKINS und dann PLONE_TEMPLATES anklicken, finden Sie das Page Template

`global_cache_settings`. Es wird auf allen Seiten benutzt, die das Plone-Haupt-Template benutzen. Dieses Template sieht zurzeit wie folgt aus:

```
<metal:cacheheaders define-macro="cacheheaders">
    <metal:block tal:define="dummy python:request.RESPONSE.setHeader ~↵
'Content-Type', 'text/html;;charset=%s' % charset)" />
    <metal:block tal:define="dummy python:request.RESPONSE.setHeader ~↵
('Content-Language', lang)" />
    <metal:block tal:define="dummy python:request.RESPONSE.setHeader ~↵
('Expires', 'Sat, 1 Jan 2000 00:00:00 GMT')" />
    <metal:block tal:define="dummy python:request.RESPONSE.setHeader ~↵
('Pragma', 'no-cache')" />
</metal:cacheheaders>
```

Das bedeutet, das nichts gecacht wird, weil die HTTP-Header `Pragma: no-cache` und `Expires` gesetzt wurden. Um das auszuschalten und um sicherzustellen, dass Sie etwas Bestimmtes cachen können, passen Sie dieses Template an und entfernen die Direktiven `Pragma` und `Expires`. Ihr Template sollte nun wie folgt aussehen:

```
<metal:cacheheaders define-macro="cacheheaders">
    <metal:block tal:define="dummy python:request.RESPONSE.setHeader ~↵
'Content-Type', 'text/html;;charset=%s' % charset)" />
    <metal:block tal:define="dummy python:request.RESPONSE.setHeader ~↵
('Content-Language', lang)" />
</metal:cacheheaders>
```

Nach diesem Schritt können Sie weiterhin bestimmte Dinge mit dem `caching_policy_manager` gezielt cachen. Gehen Sie im ZMI zu diesem Werkzeug, und Sie werden die folgenden Optionen sehen:

- **Policy ID**: Eine eindeutige ID für eine Policy, wird nur intern benutzt.
- **Predicate**: Ein TALES-Ausdruck zum Filtern der Inhalte. Die Variable `content` enthält das gerade dargestellte Objekt.
- **Mod. Time**: Ein TALES-Ausdruck, der ausgewertet wird und einen Wert aus dem Objekt für die Berechnung des Änderungszeitpunkts zurückgibt. Die Variable `content` ist das gerade dargestellte Objekt.
- **Max age (secs)**: Gibt an, wie lange der Cache-Header dafür gesetzt werden soll.
- **Vary**: Variiert den zu sendenden Header (mehr darüber erfahren Sie später im Abschnitt »Squid verwenden«).
- **No-cache**: Sendet den HTTP-Header `no-cache`.
- **No-store**: Sendet den HTTP-Header `no-store`.
- **Must-revalidate**: Sendet den HTTP-Header `must-revalidate`.

Das Folgende ist eine Beispiel-Policy, die alle Bilder auf der Site cachen würde:

- **Policy ID**: `Images`
- **Predicate**: `python:content.portal_type=='Image'`
- **Max age (secs)**: `3600`

Lassen Sie alle anderen Felder leer, und wählen Sie ADD, um diese Policy hinzuzufügen. Der `caching_policy_manager` sieht nun ungefähr wie in Abbildung 14.8 aus.

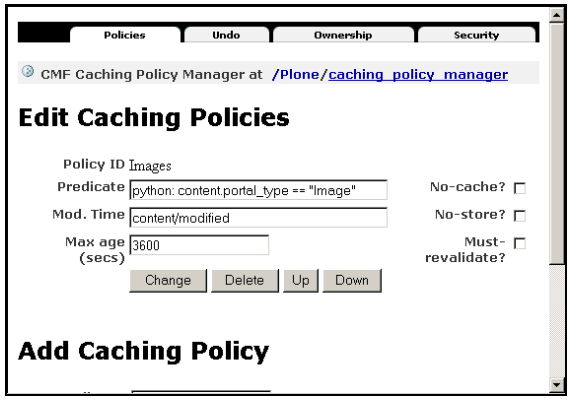


Abbildung 14.8: Der caching_policy_manager mit hinzugefügter Bilder-Policy

Um das richtig zu testen, müssen Sie über die Plone-Schnittstelle ein Bild zu Ihrer Site hinzufügen. Die Bilder werden mit den passenden Headern gesendet, wenn Sie die Aktion ANZEIGEN DES OBJEKTS aufrufen. Ich hoffe, dass man diese Aktion in späteren Versionen dieses Werkzeugs konfigurieren kann. Sie werden das mit `test.gif` testen, einem Bild, das von einem Site-Mitglied über die Plone-Schnittstelle hinzugefügt wurde, und zwar wie folgt:

```
~/header.py http://localhost/test.gif/view GET
Cache-Control: max-age=3600
Connection: close
Content-Language:
Content-Length: 19810
Content-Type: text/html;charset=utf-8
Date: Fri, 05 Sep 2003 18:42:44 GMT
Etag:
Expires: Fri, 05 Sep 2003 19:42:44 GMT
Last-Modified: Fri, 05 Sep 2003 18:33:41 GMT
Pragma: no-cache
Server: Zope/(unreleased version, python 2.2.2, linux2) ZServer/1.1
```

Wie erwartet werden nun die Header `Last-Modified` und `Expires` gesendet. Durch die Änderung von Prädikaten und das Hinzufügen mehrerer Policies können Sie ein recht ausgefeiltes Caching-System erstellen. Kompliziertere Regeln können Sie natürlich an ein Script (Python)-Objekt abgeben, wenn Sie das wünschen. Wenn das Prädikat z.B. wie folgt lautet

```
python: here.myCachingRules(content)
```

dann fügen Sie ein Script (Python)-Objekt namens `myCachingRules` hinzu, um diese Regeln zu berechnen. Beispiel:

```
##parameters=content
# cache all files, images and anything
# thats published
if content.portal_type in ['File', 'Image']:
    return 1
if content.review_state in ['published',]:
    return 1
```

In diesem Script cachen Sie alle Dateien und Bilder sowie alles, was im Zustand *Veröffentlicht* ist, indem Sie die HTTP-Header über den Caching Policy Manager setzen.

14.2.7 Beispiel: Caching auf ZopeZen.org

Bei der Entwicklung der Site *http://www.zopezen.org* gab es ein Hauptproblem. Die Erstellung der Hauptseite von ZopeZen, auf der die Nachrichten und die Anzahl der Antworten aufgelistet sind, ist sehr aufwendig. In Plone gibt es keinen einfachen Weg, um aus dem Katalog die Anzahl der Diskussionsbeiträge zu einem Element zu berechnen.

Das ist eine ideale Situation für den RAM Cache Manager. Da der Verkehr durch neue Elemente oder Nachrichten ziemlich gering ist (vielleicht ein oder zwei pro Tag), erscheint die Annahme vernünftig, dass sich die Hauptseite während einer Zeitdauer von 30 Minuten nicht stark verändert. Die Funktion, die die Nachrichten und Antworten holt, heißt `getNewsAndReplies`, und sie erledigt die Aufgabe, alle für das Template `index_html` benötigte Daten zu holen.

Das Template `index_html` verfügt über benutzerspezifische Elemente. Der Anmeldekasten links zeigt z.B. den Benutzern, welche Optionen sie haben. Das heißt, dass es nicht funktionieren würde, einen Accelerated HTTP Cache Manager zu benutzen oder das ganze Template mit dem RAM Cache Manager zu cachen. Das würde dazu führen, dass die Benutzer die Optionen anderer Benutzer sehen könnten.

Die ZopeZen-Skin cacht stattdessen das Script (Python)-Objekt `getNewsAndReplies`, indem sie dieses zum RAM Cache Manager hinzufügt. Dadurch wird garantiert, dass der Großteil der aufwendigen Arbeit der Seitendarstellung gecacht wird. Da die einzelnen Nachrichten für alle Benutzer gleich sind, macht ein Cachen auf Basis der `REQUEST`-Variablen keinen Sinn. Daher wurde `AUTHENTICATED_USER` aus der Liste der `REQUEST`-Variablen für den Cache entfernt. Eine Laufzeitmessung auf der Hauptseite ergibt ohne Cache 1,06 Anfragen pro Sekunde, während mit Cache 4,96 Anfragen pro Sekunde bearbeitet werden, das ist ein signifikanter Unterschied für eine kleine Änderung.

14.2.8 Cache-Server verwenden

Nun, da Sie Cache-Header nach ausgefeilten Regeln senden können, können Sie jetzt einen anderen Server benutzen, um Anfragen an Plone zu cachen. So schnell Zope auch ist, es kann nie schneller als Apache, Squid oder IIS sein, wenn es um die Ausgabe von Inhalten geht. Diese Server können statischen und gecachten Inhalt einfach und schnell ausgeben. Zum Teil liegt das daran, dass diese Server in C geschrieben sind, aber auch daran, dass sie bei jeder Anfrage weniger Arbeit haben. So gibt es keine Sicherheitsüberprüfungen, Datenbankabfragen oder Verhandlungen über die zu verwendende Sprache. Und als Sie Kapitel 10 gelesen haben, haben Sie außerdem schon einen Proxy-Server installiert.

Apache verwenden

Apache ist der Open-Source-Standard-Webserver. Die folgenden Abschnitte beschreiben Techniken für Apache 2.0 Server unter Linux. Mit nur geringen Änderungen an der Syntax funktionieren die meisten dieser Tipps auch unter Apache 1.3. Weitere Informationen zu verschiedenen Apache-Servern und -Plattformen finden Sie in der exzellenten Apache-Dokumentation unter *http://www.apache.org*.

Inhalte komprimieren

Die Möglichkeit, Ihre Seiten zu komprimieren, spart Bandbreite. Bevor eine Seite vom Server gesendet wird, wird sie schnell noch komprimiert, um vom Client wieder dekomprimiert zu werden. Damit lassen sich Seiten schneller herunterladen, und es fallen beim Besitzer der Site weniger Kosten wegen der verwendeten Bandbreite an, da die Dateien kleiner sind. Dazu aktivieren Sie zuerst das Modul *mod_deflate*. Das hängt von Ihren genauen Einstellungen ab. Unter Linux machen Sie z.B. Folgendes:

```
LoadModule cache_module modules/mod_deflate
```

Als Zweites fügen Sie Folgendes zu Ihrer Server-Konfiguration hinzu, um Texte in Hypertext Markup Language (HTML), Extensible Markup Language (XML) und einfache Texte zu dekomprimieren:

```
AddOutputFilterByType DEFLATE text/html text/xml text/plain
```

Manche Clients behandeln die Dekomprimierung etwas anders, d.h., es lohnt sich, die Dokumentation zu *mod_deflate* zu lesen, um detailliertere Beispiele zu sehen (*http://httpd.apache.org/docs-2.0/mod/mod_deflate.html*).

Ablauf-Header setzen

In den vorangegangenen Abschnitten haben Sie gesehen, wie Sie Ablauf-Header senden können, indem Sie Plone-Werkzeuge manipulieren. Mit Apache kann man diese Header ebenfalls leicht mit der Direktive `ExpiresActive` senden. Das ist eine Alternative zur Verwendung der verschiedenen Plone-Werkzeuge. Um z.B. für alle Bilder die Ablauf-Header auf 24 Stunden ab jetzt zu setzen, können Sie Folgendes zu Ihrer Apache-Site-Konfiguration hinzufügen:

```
ExpiresActive On
ExpiresByType image/gif "access plus 1 day"
ExpiresByType image/png "access plus 1 day"
ExpiresByType image/jpeg "access plus 1 day"
```

Weitere Informationen zu *mod_expires* finden Sie unter *http://httpd.apache.org/docs-2.1/mod/mod_expires.html*.

Caching in Apache

Apache enthält mehrere Systeme, die Caching-Aufgaben für Sie erledigen können. Das Apache-Standardmodul *mod_cache* hat zwei Caching-Modi: *Memory* und *Disk*. Dabei werden alle Seitenanfragen gemäß eines Satzes von Parametern für eine bestimmte Zeit gecacht. Um einen Cache auf der Platte im Ordner `/tmp/apache_cache` einzurichten, fügen Sie Folgendes zur Site-Konfiguration hinzu:

```
CacheRoot /tmp/apache_cache
CacheEnable disk /
CacheSize 256
CacheDirLevels 5
CacheDirLength 3
```

Leider kann es etwas schwierig sein, den Beweis zu erbringen, dass Apache den Inhalt wirklich im Cache speichert. Der einfachste Ansatz ist vielleicht der, es so zu testen, indem `z2.log` in Plone überwacht wird, um zu sehen, ob es getroffen wird. Weitere Informationen zu *mod_cache* finden Sie unter *http://httpd.apache.org/docs-2.0/mod/mod_cache.html*.

Squid verwenden

Squid ist ein Open-Source-Proxy-Server, der sehr oft zusammen mit Zope benutzt wird. Damit können Sie Zope beschleunigen, indem Sie Inhalte cachen, die in Squid erstellt werden, damit mehrere Anfragen von Squid und nicht von Zope bearbeitet werden. Auch hier gilt, dass Squid keine dynamischen Inhalte darstellt und in C geschrieben ist, d.h., es antwortet wesentlich schneller. In Kapitel 10 habe ich die Installation von Squid und dessen Einsatz als Proxy beschrieben. Wenn Sie Squid zur Beschleunigung von Plone benutzen möchten, lesen Sie bitte in diesem Kapitel die Angaben zur Einrichtung von Squid als Proxy-Server nach.

Wie Sie weiter oben in diesem Kapitel gesehen haben, können Sie mit dem Caching Policy Manager und dem Accelerated HTTP Cache Manager fast beliebige Informationen in HTTP-Headern unterbringen. Squid fungiert nun in ähnlicher Weise als Browser-Cache. Wenn eine Anfrage nach einer Seite kommt und diese Cache-Header vorhanden sind, wird Squid die Seite cachen. Wiederholte Treffer führen dazu, dass Squid und nicht Plone die Seite zurückgibt.

Ob eine Seite gecacht wurde, kann man relativ einfach sagen, da Squid den Header `X-Cache` zur Antwort hinzufügt. Mit dem Script `header.py` können Sie sehen, ob die Seite erfolgreich gecacht wurde. Ein `HIT` bedeutet, dass Squid eine Kopie im Cache gefunden und zurückgegeben hat. Wenn keine Kopie gefunden und Plone gefragt wurde, wird ein `MISS` zurückgegeben. Beispiel:

```
X-Cache: HIT from www.agmweb.ca
```

Beim Testen in der Entwicklungsumgebung zeigt Squid beeindruckende Zahlen und beschleunigt die Anzeige einer Plone-Seite im Cache von etwa zwei Anfragen pro Sekunde auf mehr als 25 pro Sekunde. Benutzer haben berichtet, dass auf schnellen Servern relativ leicht Werte von 200 Anfragen pro Sekunde erreicht werden.

Squid-Caches säubern

Wenn ein Benutzer ein Objekt bearbeitet, ändert es sich in Plone. Da dieses Objekt aber in einem früheren Zustand gecacht ist, enthält der Cache eine alte Version. Benutzer, die auf die Site zugreifen, erhalten dann eine alte Version und nicht die neue. Wenn Sie die Caches beeinflussen können (wie bei Squid), können Sie `PURGE`-Befehle an den Caching-Server schicken, damit er Objekte aus dem Cache entfernt.

Beim *Accelerated HTTP Cache Manager* fügen Sie die URLs der Caches zu `Notify URLs (via PURGE)` hinzu. Hier ein Beispiel dafür:

```
http://192.168.1.1:80/example.org
```

In diesem Beispiel ist die IP-Adresse die Adresse des Caches, und die Domain ist die zu löschende Site. Damit Squid die `PURGE`-Direktive ausführt, müssen Sie sicherstellen, dass Squid konfiguriert ist. Falls Squid auf *localhost* läuft, sähe das wie folgt aus:

```
acl PURGE method purge
http_access allow localhost
http_access allow purge localhost
http_access deny purge
http_access deny all
```

Der Caching Policy Manager verfügt im Moment über keinen *PURGE*-Mechanismus, obwohl Sie ein Script (Python)-Objekt zum Workflow hinzufügen könnten, um das zu erreichen. Sie könnten den Python-Code in Listing 14.2 als externe Methode speichern und sie bei Bedarf im Workflow ausführen.

Listing 14.2: Ein Script zum Löschen des Squid-Caches

```
import urllib
import urlparse
import httplib

URLs = [
    # enter the URLs you would like
    # to purge here
    'http://localhost:8080',
]

def purge(objectURL):
    for url in URLs:
        if not url:
            continue
        assert url[:4] == 'http', "No protocol specified"

        url = urlparse.urljoin(url, objectURL)
        parsed = urlparse.urlparse(url)
        host = parsed[1]
        path = parsed[2]

        h = httplib.HTTP(host)
        h.putrequest('PURGE', path)
        h.endheaders()
        errcode, errmsg, headers = h.getreply()
        h.getfile.read()

if __name__ == '__main__':
    print purge('/')
```

Das Collective-Projekt enthält ein neues Werkzeug namens `CMFSquidTool`, das Ihnen diese Arbeit abnimmt. Es beobachtet Änderungen am Inhalt, und wenn es eine entdeckt, sendet es für Sie ein Purge an den Squid-Cache. Ich habe es noch nicht ausprobiert, aber Sie sollten sich dieses Werkzeug definitiv anschauen, wenn Sie mit Squid arbeiten.

Säuberungen in Squid-Caches vermeiden

Säuberungen in Caches vermeidet man am besten durch selektiveres Caching. Sowohl der Caching Policy Manager als auch der RAM Cache Manager bieten Möglichkeiten zur selektiven Angabe dessen, was vom Cache zurückgegeben werden soll.

Vary

Der Caching Policy Manager und Squid unterstützen beide den `Vary`-Tag. Falls ein `Vary`-Tag angegeben wird, extrahiert Squid aus der Anfrage die im `Vary`-Tag angegebenen Header. Diese werden dann mit dem Cache verglichen. Wenn sie übereinstimmen, wird die Seite aus dem Cache zurückgegeben. Ansonsten wird die Anfrage an Plone weitergegeben.

Wenn der `Vary`-Tag im Caching Policy Manager z.B. den Wert `Accept-Language` hat und eine Anfrage an Squid kommt, wird die Seite gemäß der Einstellung `Accept-Language` in diesem Anfrage-Header gecacht. Wenn ein Benutzer eine Seite mit einer anderen Einstellung verlangt, wird eine neue Seite zurückgegeben. Das heißt, Sie können die Seiten nach Sprachen cachen.

Der am wenigsten aggressive Wert von `Vary` ist `*`, bei dem alle Anfragen gecacht werden, die mit anderen Anfragen identisch sind. Unterschiedliche Anfragen werden direkt an Plone weitergegeben. Auch wenn dies das am wenigsten aggressive Caching-System ist, garantiert es doch, dass der Benutzer nur aktuelle Inhalte sieht.

REQUEST-Methoden

Die `REQUEST`-Methoden des RAM Cache Managers verfolgen das gleiche Konzept wie `Vary`, außer dass das Werkzeug eine Liste von Zope-Anfragevariablen akzeptiert. Das Ergebnis einer Suche im Cache basiert dann auf diesen Variablen. Der voreingestellte Wert lautet `AUTHENTICATED_USER`, d.h., dass alle authentifizierten Benutzer ihre eigene Version des Caches sehen. Nicht registrierte (anonyme) Benutzer sehen alle den gleichen Inhalt.

14.3 Zope Enterprise Objects verwenden

Aber der eigentliche Schlüssel zur Skalierung und Administration von Plone besteht darin, ZEO (Zope Enterprise Objects) zu benutzen. Das ist ein wesentliches Werkzeug in sehr vielen Bereichen von Plone, sowohl für die Entwicklung als auch für die Produktion. Viele Leute denken, es sollte zu den Standardeinrichtungen von Plone gehören, und vielleicht wird es das bald. Im Moment wird unter Linux ZEO mit Plone ausgeliefert, aber nicht installiert. Es funktioniert auch unter Windows, wird dort aber nicht vollständig mit Diensten oder einer einfachen Installation unterstützt.

In einer normalen Plone-Installation gibt es eine Instanz von Plone, die mit einer Instanz der ZODB kommuniziert. Während diese eine Plone-Instanz auf die ZODB zugreift, wird diese gesperrt und kein anderer Prozess kann darauf zugreifen. Das beschränkt die Skalierbarkeit der Site und führt eine isolierte Schwachstelle ein. In der Welt relationaler Datenbanken wäre das gleichbedeutend damit, dass nur ein Prozess auf Ihre Datenbank zugreifen könnte.

ZEO bricht die Abhängigkeit auf und isoliert den Zugriff auf die ZODB (auch *ZEO-Server* genannt), damit mehrere Prozesse (auch *ZEO-Clients* genannt) sich mit der ZODB verbinden können, wie in Abbildung 14.9 zu sehen ist. In der Terminologie relationaler Datenbanken ist das äquivalent mit einem Prozess, der die Verbindung zur Datenbank herstellt.

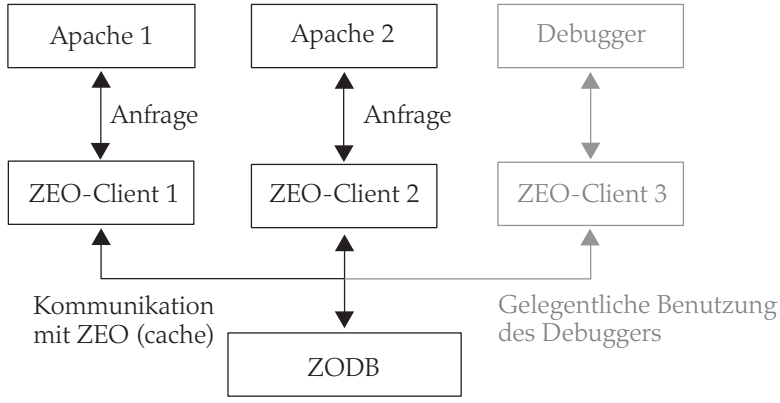


Abbildung 14.9: Eine Standardkonfiguration von ZEO

Da sich mehrere Prozesse mit einer ZODB verbinden können, haben Sie nun die Möglichkeit, mit mehreren Kopien von Plone zu arbeiten. Im Wesentlichen könnten Sie nun zwei oder drei Plone-Instanzen mit dem gleichen Inhalt haben. Das heißt, Sie können nicht nur die Last Ihrer Site auf viele Computer verteilen, sondern können nun mit einem Programm eine Verbindung zu Ihrer Site herstellen

und aufwendige Aufgaben auf anderen Computern ausführen oder Ihre Site zur Laufzeit auf Python-Ebene untersuchen.

Und schließlich ist ein auch nicht ganz unwesentlicher Punkt der, dass die Zeiten für einen Neustart eines ZEO-Clients sehr kurz sind. Der Aufwand beim Laden der Datenbanken entfällt, d.h., Sie können Plone-Sites schnell neu starten.

14.3.1 Installation von ZEO

ZEO ist in Zope 2.7 enthalten, der Zope-Version, die von diesem Buch unterstützt wird. In früheren Zope-Versionen war es separat verfügbar. Im Moment gibt es keine einfache Möglichkeit, ZEO unter Windows zu installieren. Das Script `mkzeoinstance` funktioniert nicht. ZEO selbst funktioniert, wie es soll, aber Sie müssen die ZEO-Quellen lesen, um zu sehen, wie man das anstellt. Außerdem funktioniert `zopectl` nicht unter Windows, d.h., die folgenden Beispiele funktionieren dort ebenso wenig.

Linux

Um einen ZEO-Server zu erstellen, benutzen Sie das Script `mkzeoinstance` im Verzeichnis `/opt/Zope-2.7/bin`. Das setzt voraus, dass Zope bereits wie in Kapitel 2 beschrieben installiert ist. Das Script erwartet folgende Parameter:

- **Directory**: Das Verzeichnis, in dem die ZEO-Server-Instanz erzeugt werden soll.
- **Host**: Der Host und Port, an dem der Server reagieren sollte, im Format `host:port`. Der Port ist jener Port, mit dem sich ZEO-Clients verbinden, und sollte durch eine Firewall geschützt sein, da ZEO keine Sicherheit gegen unautorisierten Zugriff bietet. Parameter sind optional. Der voreingestellte Port ist 9999.
- **User und Password**: Der vorgegebene Standardbenutzer und sein Passwort für den Server im Format `benutzer:passwort`. Ist optional.

Folgendes z.B. installiert ZEO unter `/var/zeo` auf dem Standard-Port:

```
cd /opt/Zope-2.7/bin
./mkzeoinstance /var/zeo
```

Hierdurch wird eine neue Datenbank mit der entsprechenden Konfiguration erstellt. Diese Datenbank ist an einem neuen Ort, aber das ist in Ordnung. Wenn Sie eine existierende Zope-Installation mit ZEO aufrüsten wollen, dann müssen Sie das laufende Zope anhalten und dann die Datenbank von Ihrer alten Installation in das neue ZEO-Verzeichnis verschieben. In meinem Fall heißt das, die Datei `Data.fs` von `/var/zope/var` nach `/var/zeo/var` zu verschieben.

Als Nächstes müssen Sie die Konfiguration Ihrer Zope-Instanz ändern. Öffnen Sie dazu `zope.conf` in `etc`, und geben Sie folgende Information ein:

```
# ZEO client storage:
#
<zodb_db main>
   mount-point /
   <zeoclient>
     server localhost:9999
     storage 1
     name zeostorage
     var $INSTANCE/var
   </zeoclient>
</zodb_db>
```

Im obigen Code geben Sie den Port und den Server an, auf dem sich der ZEO-Server befindet. Außerdem müssen Sie die vorhandene Abbildung auf die lokale Datenbank auskommentieren. Das sollte etwa wie folgt aussehen:

```
#<zodb_db main>
#    # Main FileStorage database
#    <filestorage>
#      path $INSTANCE/var/Data.fs
#    </filestorage>
#    mount-point /
#</zodb_db>
```

Um zu testen, ob das funktioniert, starten Sie zuerst den ZEO-Server. Dazu sind eventuell mehr Rechte notwendig, als der Benutzer hat, unter dem Sie ihn installiert haben:

```
$ cd /var/zeo/bin
$ ./zeoctl start
daemon process started, pid=29316
```

Der ZEO-Daemon wurde erfolgreich gestartet. Starten Sie nun einen Zope-Client, und versuchen Sie, sich z.B. wie folgt damit zu verbinden:

```
$ cd /var/zope/bin
$ ./zopectl start
daemon process started, pid=29338
```

Das macht alles einen guten Eindruck, und Sie können nun wie gewohnt auf Ihr Plone zugreifen.

14.3.2 ZEO-Clients benutzen

In dieser Konfiguration wird über den ZEO-Server auf die ZODB zugegriffen, und jede Zope-Instanz ist ein ZEO-Client. Mehrere ZEO-Clients können mit dem Server eine Verbindung aufbauen. Client und Server müssen keineswegs auf dem gleichen Rechner laufen, solange der Client eine Verbindung zum Server herstellen kann. Falls die Clients auf dem gleichen Computer sind, muss sich jeder Client an einen anderen HTTP- und FTP-Port binden, um Konflikte untereinander zu vermeiden.

Wenn Ihr Client startet, verbindet er sich mit dem Speicher, der in Ihrer Konfiguration angegeben ist, statt mit dem lokalen Standardspeicher. Eine häufige Anforderung ist die, einen zusätzlichen Rechner zu erlauben, auf dem rechenintensive Aufgaben laufen, z.B. eine Aktualisierung des Katalogs, eine Komprimierung der Datenbank oder eine Durchführung komplexer Abfragen, ohne dass die Performance der anderen Clients sinkt. Mit der Funktion `zopectl` lässt sich das recht einfach machen:

```
$ cd /var/zope/bin
$ ./zopectl debug
Starting debugger (the name "app" is bound to the top-level Zope object)
```

Um die Datenbank zu komprimieren, würden Sie dann Folgendes machen:

```
>>> app.Control_Panel.Database.manage_pack(days=0)
```

Weil Sie einen ZEO-Client ausführen, müssen Sie dem Server sagen, dass eine Änderung vorgenommen wurde und die Caches aktualisiert wurden. Schließen Sie die Transaktion wie folgt ab:

```
>>> get_transaction().commit()
>>> app._p_jar.close()
```

Das empfiehlt sich besonders dann, wenn Sie eine hochperformante Site betreiben und die Datenbank komprimieren müssen. Die Site wird ein wenig langsamer laufen, wenn die Transaktion beendet wird, aber der größte Teil der schweren Arbeit erfolgt auf dem Client, der die Komprimierung durchführt. Das könnte ein von Ihrer Site völlig verschiedener Rechner sein. Daher ist das eine hervorragende Methode, die Last zu verteilen.

Bei der Fehlersuche ist es extrem hilfreich, zu diesem Prompt zu gelangen, da Sie dort die Objekte in diesem `app`-Objekt untersuchen können. Sie werden feststellen, dass sie mit den Objekten übereinstimmen, die Sie im ZMI sehen. Beispiel:

```
>>> app.objectIds()
['acl_users', 'Control_Panel', 'temp_folder',...
```

Wie lautet die API für dieses `app`-Objekt? Sie können die eingebaute Python-Funktion `dir` benutzen, um das Objekt zu untersuchen und sogar die Methode `__doc__`, um die darin enthaltenen Kommentar-Strings wie folgt zu sehen:

```
$dir(app)
>>> dir(app)
['COPY', 'COPY__roles__', 'Control_Panel', 'DELETE',...
>>> app.valid_roles.__doc__
'Return list of valid roles'
```

Ein gutes Beispiel einer ZEO-basierten Anwendung ist *CMFNewsFeed* (*http://sf.net/projects/collective*). Sie stellt eine Verbindung zu Plone mit einem ZEO-Client her. Dieser separate Client legt dann los und sammelt alle Nachrichten, die er finden kann, und fügt die Daten in die Site ein. Dadurch, dass alles Sammeln und Katalogisieren in einem separaten Prozess läuft, wird garantiert, dass die Performance der Haupt-Site hoch bleibt.

Für Entwickler ist ZEO ein unverzichtbares Werkzeug. Damit können Sie durch ein Programm mit Ihrem Server interagieren, während er läuft. Wenn Sie als erfahrener Programmierer an dieser Stelle immer noch verwirrt sein sollten, was Plone und die Objektdatenbank angeht, dann öffnet Ihnen ZEO normalerweise die Augen.

14.3.3 Lastverteilung und Ausfallsicherung

Obwohl ZEO die Möglichkeit bietet, Plone auf vielen Servern zu betreiben, hat es für den Benutzer nichts zu bieten, womit er eine Lastverteilung vornehmen könnte. Unter *Lastverteilung* (engl. *load balancing*) versteht man, dass hereinkommende Anfragen auf verschiedene Server gesendet werden, um die Last der Seitengenerierung zu verteilen. Ausgefeilte Werkzeuge testen, ob der Server überhaupt läuft, bevor sie ihm eine Anfrage schicken.

Bei der Verteilung der Last haben Sie Hardware- und Software-Möglichkeiten. Squid z.B. beherrscht die dynamische Ausfallsicherung (engl. *failover*). *Pound* ist ein Beispiel für ein Lastverteilungssystem, das Sie unter *http://www.apsis.ch/pound/index.html* finden.

Squid kennt das *Internet Cache Protocol* (ICP), ein Protokoll, mit dem es prüfen kann, ob eine Plone-Site läuft, bevor Anfragen an sie weitergereicht werden. Auf hochgradig dynamischen Sites ist das eventuell ein Muss. Weitere Informationen zu ICP und Zope finden Sie unter *http://www.zope.org/Members/htrd/icp/intro*.

A Wichtige Konfigurationen und einige APIs

Dieser Anhang enthält einige der wichtigsten Konfigurationsmöglichkeiten bei der Entwicklung mit Zope, Plone und Python. Er enthält Informationen für Site-Entwickler und listet auch einige der nützlichsten APIs (Application Programming Interfaces) auf.

A.1 Einrichten Ihrer Umgebung

In den folgenden Abschnitten finden Sie Informationen über die Konfiguration Ihrer Entwicklungs- oder Produktionsumgebung, damit Sie diese optimal einrichten können. Wenn Sie viel mit Plone entwickeln, möchte ich Ihnen diese Einstellungen sehr empfehlen.

A.1.1 PYTHONPATH einrichten

Die Einrichtung von `PYTHONPATH` ist sehr nützlich, weil Sie dadurch sehr leicht auf die gesamte Zope-Funktionalität vom Python-Prompt aus zugreifen können. Sie können ganz leicht herausfinden, ob Sie diese Einstellung bereits verwenden. Versuchen Sie einfach, das Modul `PageTemplate` aus `Products` zu importieren. Wenn das noch nicht eingerichtet ist, werden Sie folgenden Fehler sehen:

```
$ python -c "import Products.PageTemplate"
Traceback (most recent call last):
  File "<string>", line 1, in ?
ImportError: No module named Products.PageTemplate
```

Unix, Linux und Mac OS X

Finden Sie zuerst das Produktverzeichnis Ihrer Zope-Installation (nicht die Instanzwurzel). Bei einer Standardinstallation liegt es unter `/opt/Zope-2.7/lib/python`. Unter Windows liegt es unter `C:\Programme\Plone\Zope\lib\python`. Wenn Python gestartet wird, liest es eine Umgebungsvariable namens `PYTHONPATH`,

aus der es alle dort angegebenen Verzeichnisse in seinen Suchpfad für neue Module übernimmt. Das heißt, Sie müssen Ihr Verzeichnis zu dieser Variablen hinzufügen.

Das machen Sie mit dem Befehl `export`. Um also zu sehen, ob zu Beginn irgendetwas in `PYTHONPATH` enthalten ist, führen Sie Folgendes aus:

```
$ export | grep PYTHONPATH
declare -x PYTHONPATH="/home/andy/modules"
```

In meinem Fall habe ich bereits eine Umgebungsvariable namens `PYTHONPATH`, aber auf Ihrem Rechner haben Sie diese Einstellung möglicherweise nicht. Bei mir enthält sie einen Pfad zu einigen internen Modulen. Nun müssen Sie also Zope 2 zu diesem Pfad hinzufügen, z.B. so:

```
$ export PYTHONPATH="/opt/Zope-2.7/lib/python:$PYTHONPATH"
```

Ob das funktioniert hat, können Sie testen, indem Sie den folgenden Befehl wiederholen, wobei Sie darauf achten sollten, dass kein Fehler auftritt.

```
$ python -c "import Products.PageTemplate"
```

Windows

Unter Windows befindet sich das Zope-Produktverzeichnis in `C:\Programme\Plone 2\Zope\lib\python`. Wenn Python gestartet wird, liest es eine Umgebungsvariable namens `PYTHONPATH`, aus der es alle dort angegebenen Verzeichnisse in seinen Suchpfad für neue Module übernimmt. Das heißt, Sie müssen Ihr Verzeichnis zu dieser Variablen hinzufügen. Unter Windows fügen Sie eine Umgebungsvariable hinzu, indem Sie auf das Icon »ARBEITSPLATZ« rechtsklicken und EIGENSCHAFTEN auswählen. Klicken Sie im Dialogfeld SYSTEMEIGENSCHAFTEN auf den Reiter ERWEITERT, und klicken Sie dann unter SYSTEMVARIABLEN auf den Button NEU, wie in Abbildung A.1 zu sehen ist.

Anschließend wird das Dialogfeld SYSTEMVARIABLEN geöffnet. Um eine Variable zu bearbeiten, wählen Sie eine aus der Liste und bearbeiten den Wert, wie in Abbildung A.2 gezeigt wird. Die Variable sollte `PYTHONPATH` heißen, und ihr Wert sollte der Ort sein, wo sich Ihr Plone befindet, z.B. `C:\Programme\Plone2\Zope\lib\python`.

Nun sollten Sie die Import-Anweisung ganz normal ausführen können. Das können Sie von der Kommandozeile aus tun, oder Sie starten PythonWin und probieren dort den Import aus, wie in Abbildung A.3 demonstriert wird.

Ab jetzt können Sie nicht nur Produkte importieren, sondern Sie können auf der Kommandozeile auch `import Zope` ausführen, was Voraussetzung für viele Scripten und Werkzeuge in den fortgeschritteneren Kapiteln dieses Buchs ist.

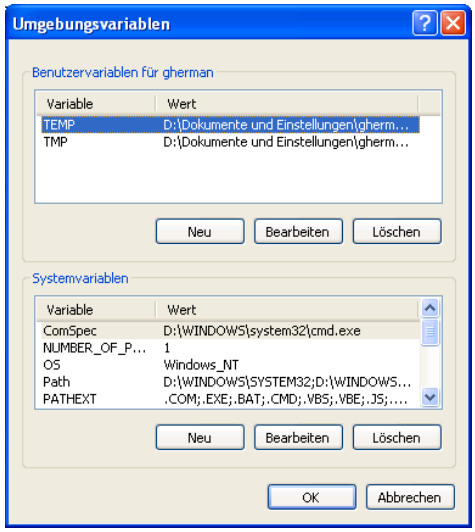


Abbildung A.1: Das Dialogfeld für die Umgebungsvariablen

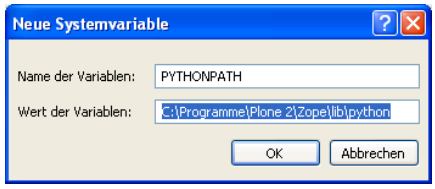


Abbildung A.2: Eine Variable hinzufügen

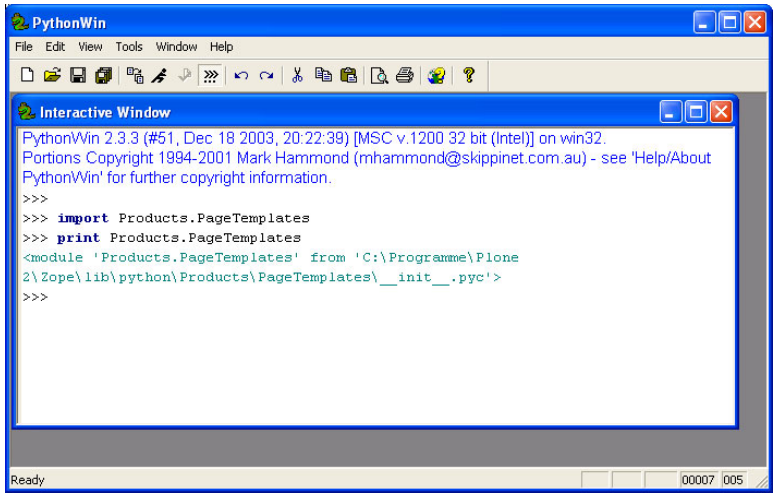


Abbildung A.3: Importieren in PythonWin

A.1.2 Ausführung von Unittests einrichten

Plone enthält einige hundert Unittests, die eine hervorragende Methode darstellen, um zu überprüfen, ob Ihr Plone korrekt funktioniert. Die Plone-Unittests finden Sie in `CMFPlone/tests`. Wenn Sie mit Plone entwickeln, dann wäre es eine gute Idee, diese auf Ihrem Rechner einzurichten, damit Sie eigene Tests hinzufügen können. Um diese Tests auszuführen, benötigen Sie `ZopeTestCase`. In Zukunft wird das vermutlich ein Teil von Plone werden, aber im Moment müssen Sie das noch unter *http://zope.org/Members/shh/ZopeTestCase* herunterladen.

Die Installation ist etwas ungewöhnlich, Sie müssen das Paket erst auspacken und dann in das Verzeichnis `lib/Python/Testing` des Zope-Wurzelverzeichnisses kopieren. Unter Unix liegt es normalerweise in `opt/Zope-2.7`, während es unter Windows in `C:\Programme\Plone 2\Zope` liegt.

Als nächstes benötigen Sie ein Script, das die Tests ausführt. Dazu schreibe ich normalerweise ein kleines Shell-Script und kopiere es ins `bin`-Verzeichnis meiner Plone-Instanz. Dann kann ich einfach dieses Shell-Script starten, um die Tests auszuführen. Unter Unix, wo sich meine Plone-Instanz unter `/var/test` befindet, sieht mein Script wie folgt aus:

```
export SOFTWARE_HOME=/opt/Zope-2.7/lib/python
export INSTANCE_HOME=/var/test123

echo Testing CMFPlone...
cd $INSTANCE_HOME/Products/CMFPlone/tests
python2.3 runalltests.py
```

Unter Windows sieht das Äquivalent dazu so aus:

```
set SOFTWARE_HOME=C:\Programme\Plone 2\Zope
set INSTANCE_HOME=C:\Programme\Plone 2\Data

cd "C:\Programme\Plone 2\Data\Products\CMFPlone\tests"
"C:\Programme\Plone 2\Python\python.exe" runalltests.py
```

Die Ausgabe sollte ungefähr wie folgt aussehen:

```
[root@basil bin]# ./testAll.sh
Testing CMFPlone...
SOFTWARE_HOME: /opt/Zope-2.7/lib/python
INSTANCE_HOME: /opt/Zope-2.7/lib/python/Testing
Loading Zope, please stand by ... done (7.899s)
Installing CMFCore ... done (1.363s)
Installing CMFDefault ... done (0.713s)
Und so weiter...
```

Das Unittest-Framework ist eine hervorragende Grundlage für die Durchführung von Unittests bei eigenen Produkten. Wenn Sie Produkte oder allgemein Software für Plone entwickeln, dann sollten Sie idealerweise Unittests dafür schreiben. Die in Plone enthaltenen Unittests sind ein exzellentes Beispiel dafür, wie Sie das machen können.

A.1.3 Die Zope-Konfigurationsdatei

Die Zope-Konfigurationsdatei ist in einem Format namens ZConfig geschrieben. Weitere Informationen über ZConfig finden Sie unter *http://www.zope.org/Members/fdrake/zconfig*. Die Konfigurationsdatei hat den Namen `zope.conf` und befindet sich im `etc`-Verzeichnis Ihrer Zope-Installation. Manche Installationsprogramme erzeugen auch eine Datei namens `plone.conf` mit Plone-spezifischen Angaben.

Als Plattform für alle Plone 2-Installationen wird Zope 2.7 empfohlen. In dieser Version von Zope kam diese Datei zum ersten Mal vor. Davor wurden Parameter über die Kommandozeile an Zope übergeben.

In dieser Datei fangen alle Kommentare mit einem # an. In ZConfig definieren Sie Variablen, die später in dieser Konfigurationsdatei benutzt werden können. Dazu verwenden Sie eine Zeile im folgenden Format:

```
%define variablen-name variablen-wert
```

Folgende Zeilen sind Beispieldefinitionen aus meiner Konfigurationsdatei:

```
%define INSTANCE /var/test
%define ZOPE /opt/Zope-2.7
```

Diese Variablen benutzen Sie dann, indem Sie ihren Namen ein `$` voranstellen, wie z.B. in `$INSTANCE`.

Tabelle A.1 enthält die gesamte Zope-Konfiguration für Version 2.0.1. Die Spalte *Direktive* enthält den Variablennamen in der Konfigurationsdatei. Die Spalte *Beschreibung* erklärt, was die Direktive bewirkt, und die Spalte *Vorgabe* gibt an, was passiert, wenn kein Wert für diese Variable gesetzt ist. In der Spalte *Beispiel* habe ich ein paar Beispiele als zusätzliche Hilfe zur Erklärung angegeben.

Direktive	Beschreibung	Vorgabe	Beispiel
`instancehome`	Der Pfad zu den Dateien mit Daten, Produkten, dem Import-Verzeichnis und dem `Extensions`-Verzeichnis. Jede Zope-Instanz sollte eine solche Direktive haben.	Keine	`/var/zope` oder `$INSTANCE`
`client home`	Das Verzeichnis, in dem die Dateien zur Prozessidentifikation von Zope liegen, z.B. die Prozess-ID-Datei `Z2.pid`. Von seinem Gebrauch wird abgeraten. Siehe stattdessen `pid-filename`.	`INSTANCE/var`	`$INSTANCE_HOME/var`
`Path`	Name eines Verzeichnisses, das am Anfang von Pythons Modulsuchpfad eingefügt werden soll. Diese Direktive kann mehrfach verwendet werden. Da das zu spät sein kann, sollte lieber, wie vorher beschrieben, `PYTHONPATH` verändert werden.	Keine	`path /home/pythonModules`
`Products`	Name eines Verzeichnisses, das weitere Produkte enthält. Kann mehrfach verwendet werden. Jedes identifizierte Verzeichnis wird zum `__path__` des `Products`-Pakets hinzugefügt. Ich muss vom Gebrauch abraten, da normalerweise ein `instance home` genügt. Es wurden auch schon Fehler in Plone wegen dieser Direktive gemeldet.	Keine	`products /home/chrism/projects/myproducts`
`environment`	Ein Abschnitt, in dem beliebige Schlüssel/Wert-Paare als Umgebungsvariablen definiert werden können, während Zope läuft. Es empfiehlt sich nicht, hier Systemvariablen wie `PYTHONPATH` zu setzen. Die meisten Startup-Scripten definieren für Sie solche Variablen. Deswegen wird vom Gebrauch abgeraten.	Keine	`<environment> MY_PRODUCT_ENVVAR foobar</environment>`

Tabelle A.1: Direktiven in der Zope-Konfigurationsdatei

Direktive	Beschreibung	Vorgabe	Beispiel
`debug-mode`	Ein Schalter, der verschiedene Bereiche im Betrieb von Zope betrifft, die bei der Entwicklung mit Zope nützlich sind. Ich empfehle, alle Entwicklungsserver in diesem Modus zu betreiben. Manche Plone-Installationsprogramme schalten das standardmäßig evtl. aus Gründen der Performance aus.	`on`	`debug-mode off`
`effective-user`	Wenn Sie Zope unter dem Benutzer `root` betreiben möchten, muss diese Direktive mit dem Namen oder der ID eines effektiven Benutzers angegeben werden, zu dem Zope mit `suid` umschaltet, nachdem die Server-Ports belegt sind. Funktioniert nur unter Unix und wenn Zope unter `root` gestartet wird. Sie können einen Benutzer namens `zope` erstellen und den effektiven Benutzer auf `zope` setzen, damit Zope garantiert unter diesem Benutzer läuft. Außerdem dürfen Sie dann Ports zwischen 21 und 80 belegen.	Keine	`effective-user zope`
`enable-product-installation`	Wenn diese Direktive eingeschaltet ist, führt Zope eine Produkt-Installation beim Hochfahren durch (eine Registrierung von Python-Modulen in verschiedenen Produktverzeichnissen). Wenn man das ausschaltet, erfolgt das Hochfahren von Zope/ZEO eventuell schneller, kann aber auch dazu führen, dass Ihre Produktliste im Control Panel nicht mehr synchron mit dem Inhalt der Produktverzeichnisse ist.	`on`	`enable-product-installation off`

Tabelle A.1: Direktiven in der Zope-Konfigurationsdatei (Forts.)

Direktive	Beschreibung	Vorgabe	Beispiel
`locale`	Das unterstützt die Internationalisierung, durch die Angabe eines Namens für ein Locale. Schauen Sie in die Dokumentation Ihres Betriebssystems für spezifische Informationen.	Keine	`locale fr_FR`
`port-base`	Der auf Port-Nummern angewendete Offset bei der ZServer-Konfiguration. Wenn der `http-server`-Port gleich 8080 und `port-base` gleich 1000 ist, hört der HTTP-Server den Port 9080 ab. Das ist dann gut, wenn Sie mehrere Plone-Instanzen auf einem Rechner betreiben und schnell alle Port-Nummern ändern müssen.	`0`	`1000`
`datetime-format`	Setzt diese Variable entweder auf `us` oder `international`, um das `DateTime`-Modul jeweils dazu zu zwingen, Datumsstrings mit den Formaten MM-DD-JJ bzw. DD-MM-JJ zu parsen. Der Vorgabewert lautet `us`.	`us`	`datetime-format international`
`zserver-threads`	Gibt die Anzahl der Threads an, mit denen Zopes Webserver ZServer Anfragen bedient. Auf den meisten Sites führt ein hoher Wert an dieser Stelle nicht zu besserer Performance. Dazu verwende man ZEO und Caching.	`4`	`zserver-threads 10`
`python-check-interval`	Ein Integer für das »Check-Intervall« des Python-Interpreters, das bestimmt, wie oft der Interpreter periodische Aufgaben durchführt, z.B. das Umschalten von Threads und die Signalbehandler. Auf den meisten Sites führt ein hoher Wert an dieser Stelle nicht zu besserer Performance.	`500`	`python-check-interval 1000`

Tabelle A.1: Direktiven in der Zope-Konfigurationsdatei (Forts.)

Direktive	Beschreibung	Vorgabe	Beispiel
`zserver-read-only-mode`	Wenn eingeschaltet, erzeugt Zope keine Log- und Prozess-ID-(PID-)Dateien. Zugriffs- und Fehler-Logdateien werden auf die Standardausgabe ausgegeben.	`off`	`zserver-read-only-mode on`
`pid-filename`	Der Pfad der Datei, in der die PIDs von Zope geschrieben werden. Der Vorgabewert lautet `client-home/Z2.pid`.	`CLIENT_HOME /Z2.pid`	`pid-filename /home/chrism/projects/sessions/var/Z2.pid`
`lock-filename`	Der Pfad der »Lock-Datei«, die von Zope im laufenden Betrieb gesperrt wird.	`CLIENT_HOME /Z2.lock`	`lock-filename /home/chrism/projects/sessions/var/Z2.lock`
`mime-types`	Hiermit erfährt Zope von weiteren `mime-types`-Dateien, die es laden soll. Die Dateien haben das gleiche Format wie die von Apache. Diese Einstellung darf mehrmals in einer Konfigurationsdatei vorkommen.	Keine	`mime-types $INSTANCE/etc/mime.types`
`structured-text-header-level`	Setzt den Vorgabewert des HTML-Header- Levels in Dokumenten in strukturiertem Text. Standardwert ist `3`, d.h., die obersten Header werden mit dem Tag `<h3>` erzeugt. Leider ignoriert CMF diesen Wert noch, was irgendwann noch geändert werden muss.	`3`	`structured-text-header-level 1`
`rest-input-encoding`	Gibt die Eingabekodierung von Dokumenten in restrukturiertem Text an, z.B. als `utf-8`, `iso-8859-15` oder eine andere gültige und von Python erkannte Kodierung. Der Vorgabewert ist der Ihrer Python-Version.	Systemvorgabewert	`rest-input-encoding iso-8859-15`

Tabelle A.1: Direktiven in der Zope-Konfigurationsdatei (Forts.)

Direktive	Beschreibung	Vorgabe	Beispiel
`rest-output-encoding`	Gibt die Ausgabekodierung von Dokumenten in restrukturiertem Text an, z.B. `utf-8`, `iso-8859-15` oder eine andere gültige und von Python erkannte Kodierung. Der Vorgabewert ist der Ihrer Python-Version.	Systemvorgabewert	`rest-output-encoding iso-8859-15`
`cgi-environment`	Ein Abschnitt, in dem Benutzer beliebige Schlüssel/Wert-Paare für CGI-Umgebungsvariablen angeben können.	Keine	*<cgi-environment>* *HTTPS_SERVER Foobar Server 1.0* *HTTPS_PORT 443* *</cgi-environment>*
`dns-server`	Gibt die IP-Adresse Ihres Domain Name System-(DNS-) Servers an, durch den in Zopes Zugriffslogdateien aufgelöste Hostnamen geschrieben werden. Wenn Sie das einschalten, werden Verzögerungen durch DNS-Anfragen Ihre Site verlangsamen.	Keine	`dns-server 127.0.0.1`
`ip-address`	Gibt die IP-Adresse vor, auf denen die verschiedenen Server-Protokolle ihre Anfragen erwarten. Ohne einen Wert an dieser Stelle hört Zope alle verfügbaren IP-Adressen auf dem Rechner ab.	Keine	`ip-address 127.0.0.1`
`http-realm`	Der HTTP-Header-Wert für `Realm`, den Zope in dieser Instanz ausgibt. Dieser Wert taucht oftmals in Dialogfeldern zur einfachen Authentifizierung auf.	`Zope`	`Plone`
`automatically-quote-dtml-request-data`	Setzen Sie diese Direktive auf `off`, um das »Autoquoting« von implizit erhaltenen `REQUEST`-Daten in DTML-Code zu unterbinden, der ein in <`dtml-var`>-Konstrukten enthält.	`On`	`automatically-quote-dtml-request-data on`

Tabelle A.1: Direktiven in der Zope-Konfigurationsdatei (Forts.)

Direktive	Beschreibung	Vorgabe	Beispiel
	Sonst werden alle implizit von `REQUEST` in DTML erhaltenen Daten (im Gegensatz zum direkten Zugriff mit `REQUEST.einVarName`), die ein < enthalten, mit HTML-Quotes versehen, wenn sie mit <`dtml-var`> oder `&dtml-` interpoliert werden. Das verringert die Wahrscheinlichkeit, dass Programmierer ihre Sites für clientseitige Trojaner-Attacken offen lassen.		
`trusted-proxy`	Gibt einen oder mehrere Hostnamen oder IP-Adressen an.	Keine	`trusted-proxy www.example.com`
`publisher-profile-file`	Gibt eine Datei im Dateisystem an, durch die Zopes Profiling-Fähigkeiten aktiviert werden. Weitere Informationen dazu finden Sie unter dem PROFILING-Reiter im ZMI. Sollte in Produktion nicht gesetzt werden, da dadurch der Code wesentlich langsamer ausgeführt wird als normal.	Keine	`publisher-profile-file $INSTANCE/var/profile.dat`
`security-policy-implementation`	Die normale Sicherheitsmaschinerie von Zope ist in C implementiert. Die Python-Version können Sie verwenden, wenn Sie hier `python` setzen. Diese ist langsamer, bietet mit Verbose-Security aber wichtige Informationen.	`C`	`security-policy-implementation python`
`skip-authentication-checking`	Setzen Sie hier `on`, wenn Zope Prüfungen bei der Authentifizierung von Servern auslassen soll, die nur anonyme Inhalte anbieten.	`off`	`skip-authentication-checking on`

Tabelle A.1: Direktiven in der Zope-Konfigurationsdatei (Forts.)

Direktive	Beschreibung	Vorgabe	Beispiel
`skip-ownership-checking`	Setzen Sie hier `on`, wenn Zope Besitz-Prüfungen auslassen soll, wenn Code »über das Web« ausgeführt werden soll. Das ist standardmäßig eingeschaltet, um Sicherheitsprobleme mit Trojanern zu verhindern, wobei Benutzer mit niedrigen Privilegien solche mit höheren Privilegien dazu bringen können, gefährlichen Code auszuführen.	`on`	`skip-ownership-checking off`
`maximum-number-of-session-objects`	Ein Integer-Wert als »maximale Anzahl von Unterobjekten« des transienten Objekt-Containers in `/temp/folder/session_data`	`1000`	`maximum-number-of-session-objects 10000`
`session-add-notify-script-path`	Ein optionaler Zope-Pfadname eines aufrufbaren Objekts, das als »Script beim Addieren von Objekten« des transienten Objekt-Containers aufgerufen werden soll, der beim Hochfahren im Ordner `/temp_folder` erzeugt wird. Für den Einsatz mit Sessions.	ungesetzt	`session-add-notify-script-path /scripts/add_notifier`
`session-delete-notify-script-path`	Ein optionaler Zope-Pfadname eines aufrufbaren Objekts, das als »Script beim Löschen von Objekten« des transienten Objekt-Containers aufgerufen werden soll, der beim Hochfahren im Ordner `/temp_folder` erzeugt wird. Für den Einsatz mit Sessions.	ungesetzt	`session-delete-notify-script-path /scripts/del_notifier`
`session-timeout-minutes`	Ein Integer-Wert für die Anzahl der Minuten, die als »Datenobjekt-Timeout« des transienten Objekt-Containers `/temp/folder/session_data` benutzt werden soll.	`20`	`session-timeout-minutes 30`

Tabelle A.1: Direktiven in der Zope-Konfigurationsdatei (Forts.)

Direktive	Beschreibung	Vorgabe	Beispiel
`suppress-allaccess-rules-`	Wenn auf `on` gesetzt, werden in Ihrer Zope-Site keine Zugriffsregeln ausgeführt. Nützlich, wenn Sie sich selbst aus einem bestimmten Teil Ihrer Site aussperren, indem Sie eine falsche Zugriffsregel einstellen.	`off`	`suppress-all-access-rules on`
`suppress-all-site-roots`	Wenn auf `on` gesetzt, sind in Ihrer Zope-Site keine Site-Roots aktiviert. Nützlich, wenn Sie sich selbst aus einem bestimmten Teil Ihrer Site aussperren, indem Sie eine falsche Site-Root schreiben.	`off`	`suppress-all-site-roots on`
`database-quota-size`	Die obere Grenze als Anzahl von Bytes für die Größe der FileStorage-basierten Zope-Datenbank. Nachdem diese Zahl erreicht wird, können keine weiteren Objekte zur Datenbank hinzugefügt werden.	Keine	`database-quota-size 1000000`
`read-only-database`	Bewirkt, dass die FileStorage-basierte Zope-Datenbank ZODB nur lesend geöffnet wird. Andere Dateien, z.B. Logdateien, können geschrieben werden.	`off`	`read-only-database on`
`zeo-client-name`	Wenn Sie einen persistenten ZEO-Client-Cache haben möchten, der den Cache-Inhalt über Neustarts von ClientStorage hinweg aufbewahrt, müssen Sie einen `zeo-client-name` definieren. Wenn Sie sonst ZEO benutzen, wird der Client-Cache in temporären Dateien gespeichert, die gelöscht werden, wenn ClientStorage beendet wird. Der Wert von `zeo-client-name` ist eindeutig für die erzeugten Cache-Dateien, falls diese Zope-Instanz ein ZEO-Client ist.	`off`	`Zeo-client-name on`

Tabelle A.1: Direktiven in der Zope-Konfigurationsdatei (Forts.)

Direktive	Beschreibung	Vorgabe	Beispiel

`logger`	Dieser Bereich sollte einen oder mehrere `logger`-Abschnitte mit den Namen `access`, `event` und `trace` definieren. Der `access`-Logger loggt Zugriffe auf den Zope- Server fest, der `event`-Logger loggt Informationen zu Zope-Events, und der `trace`-Logger loggt detaillierte Informationen zu Serveranfragen (nur für die Fehlersuche). Jeder Logger-Abschnitt darf ein Name/Wert-Paar eines Levels enthalten, mit dem der Grad an Logging-Details angegeben wird. Der Vorgabe-Level lautet `INFO`. Erlaubte Werte hierfür sind `CRITICAL`, `ERROR`, `WARN`, `INFO`, `DEBUG` und `ALL`. Jeder Logger-Abschnitt darf weiterhin ein oder mehrere `handler`-Abschnitte enthalten, die den Typ des Log-Handlers angeben.
	
	Es gibt fünf Handler-Typen: `logfile`, `syslog`, `win32-eventlog`, `http-handler` und `email-notifier`. Jeder hat seine eigenen erlaubten Unterschlüssel, die gewisse Aspekte des Handlers bestimmen. In allen Handler-Abschnitten kann auch `format` (der Format-String der Log-Einträge), `dateformat` (der Format-String der Datums-Strings) und `level` angegeben werden. Letzterer hat die gleiche Semantik wie wie der übergeordnete Logger-Level, überschreibt aber den Level-Logger des Handlers, in dem er definiert ist.
		Der `access`-Log schreibt auf dem Level `INFO` in die Datei `<instance home/log /Z2.log>`, der `event`- Log auf dem Level `INFO` in das Datei-Log, und der `trace`-Log wird nirgendwo geschrieben.	`<eventlog>`
			`  level ALL`
			`  <logfile>`
			`  path $INSTANCE/log/event.log`
			`  level INFO`
			`  </logfile>`
			`</eventlog>`
			`<logger access>`
			`  level WARN`
			`  <logfile>`
			`  path $INSTANCE/log/Z2.log`
			`  format %(message)s`
			`  </logfile>`
			`</logger>`

Tabelle A.1: Direktiven in der Zope-Konfigurationsdatei (Forts.)

Direktive	Beschreibung	Vorgabe	Beispiel
warnfilter	In diesem Abschnitt können Sie Warnungsfilter angeben. Folgende Schlüssel gelten in einem warnfilter-Abschnitt, action: einer der Strings error, ignore, always, default, module und once; message: ein String mit einem regulären Ausdruck, mit dem die Warnung übereinstimmen muss (unabhängig von Groß-/Kleinschreibung); category: ein Python-Klassenname mit Punkten (muss eine Unterklasse von Warning sein), von der die Warnungskategorie eine Unterklasse sein muss, damit die Übereinstimmung erfolgen kann; module: ein String mit einem regulären Ausdruck, der mit dem Modulnamen übereinstimmen muss (unabhängig von der Schreibweise); lineno: ein Integer, der mit der Nummer der Zeile übereinstimmen muss, in der die Warnung aufgetreten ist, oder 0, was mit allen Zeilennummern übereinstimmt.	Keine	<warnfilter> action ignore category exceptions.DeprecationWarning </warnfilter>
max-listen-sockets	Die maximale Anzahl von Sockets, die ZServer versucht zu öffnen, um einkommende Verbindungen zu bedienen.	1000	max-listen-sockets 500
servers	Eine Reihe von Abschnitten, mit denen die verschiedenen ZServer von Zope angegeben werden können. Es können sieben Servertypen definiert werden: http-server, ftp-server, webdav-source-server, persistent-cgi, fast-cgi, monitor-server und icp-server. Wenn keine Server definiert werden, werden die vorgegebenen Server benutzt.	HTTP-Server auf Port 8080 und FTP auf 8021	<http-server> # valid key is "address" and "force-connection- close" address 8080 # force-connection-close on </http-server> <ftp-server> # valid key is "address" address 8021 </ftp-server>

Tabelle A.1: Direktiven in der Zope-Konfigurationsdatei (Forts.)

Direktive	Beschreibung	Vorgabe	Beispiel
	Ports können entweder in einer einfachen Form angegeben werden (80) oder in komplexer Form inklusive Hostname (127.0.0.1:80). Wenn der Hostname weggelassen wird, wird als Hostname `default-ip-address` benutzt. Port-Nummern werden durch die Einstellung in `port-base` mit dem Standardwert 8000 verschoben. In Ihrer Plone-Installation kann das verändert worden sein, damit eine erste Installation einfacher wird.		

`database`	Der `database`-Abschnitt erlaubt die Angabe eigener Datenbank- und Speichertypen. Es kann mehr als ein `zodb_db`-Abschnitt definiert werden. Die Werte im Speicher werden von den Werten in dem jeweiligen Datenbank-Client gesetzt.	Siehe Beispiel.	`<zodb_db main>`
			`  # Main FileStorage database`
			`  <filestorage>`
			`  path $INSTANCE/var/Data.fs`
			`  </filestorage>`
			`  mount-point /`
			`</zodb_db>`
			`<zodb_db temporary>`
			`  # Temporary database database (for sessions)`
			`  <temporarystorage>`
			`  name temporary storage for sessioning`
			`  </temporarystorage>`
			`  mount-point /temp_folder`
			`  container-class Products.TemporaryFolder.TemporaryContainer`
			`</zodb_db>`

Tabelle A.1: Direktiven in der Zope-Konfigurationsdatei (Forts.)

Von allen Konfigurationen sind die letzten paar (Datenbank, Server und Logger) Direktiven in einer einfachen XML-artigen Syntax. Wenn Sie mit der Apache-Protokolldatei arbeiten, wird Ihnen dieses Format bekannt vorkommen. Diese Direktiven können Sie auch mehrfach anwenden. So besteht eine häufige Konfiguration von Installationsprogrammen auf dem Mac oder unter Windows darin,

HTTP-Verbindungen auf zwei Ports, 80 und 8080, durchzuführen. Das machen Sie wie folgt:

```
<http-server>
  address 8080
</http-server>

<http-server>
  address 80
</http-server>
```

A.2 Regeln zur Textformatierung

Plone verfügt mit External Editor und Epoz über zwei exzellente Werkzeuge zur Bearbeitung von HTML. In Zope und Python werden allerdings zwei Formate von einfachem Text häufig verwendet: strukturierter Text und restrukturierter Text. Da die beiden genannten Editoren qualitativ sehr gut sind, glaube ich, dass Sie diese Formate nicht benötigen werden. Aber wenn Sie mit Zope oder Plone entwickeln werden, werden Ihnen diese Formate sehr wahrscheinlich irgendwann begegnen.

Beide Formate versuchen, ähnliche Dinge zu machen: Sie bieten ein System zur Auszeichnung von einfachem Text an, aus dem sie HTML generieren. Das zielt auf Entwickler ab, die gern mit einfachem Text arbeiten und daran gewöhnt sind, die aber den Aufwand scheuen, selbst HTML zu produzieren.

A.2.1 Strukturierter Text

Es folgt eine Darstellung, wie sie bereits in einem älteren Artikel erschienen ist. Sie finden ihn online unter *http://plone.org/documentation-old/howto/UsingStructuredText*. Im folgenden Abschnitt ist die Einrückung des Textes im Code von entscheidender Bedeutung dafür, wie der Text dargestellt wird.

Einfache Formatierung

Das Grundkonzept von strukturiertem Text baut auf einem Absatz auf. Das Beispiel

```
Das ist der erste Absatz.

Das ist der zweite Absatz.
```

wird zu folgendem HTML umgewandelt:

```
<p>Das ist der erste Absatz.</p>
<p>Das ist der zweite Absatz.</p>
```

Die Whitespaces zwischen Elementen in strukturiertem Text sind von Bedeutung. In diesem Fall bewirkt eine Leerzeile zwischen den beiden anderen, dass ein neuer Absatz anfängt. Das ist intuitiv leicht zu verstehen. In E-Mails z.B. werden Absätze auch mit Leerzeilen voneinander getrennt. Für Hervorhebungen im Text benutzt strukturierter Text eine andere Konvention, nämlich Sternchen. Beispiel:

```
Das ist der *erste* Absatz.

Das ist der **zweite** Absatz.
```

In HTML entstehen daraus die Tags `em` und `strong`:

```
<p>Das ist der <em>erste</em> Absatz.</p>
<p>Das ist der <strong>zweite</strong> Absatz.</p>
```

Auch dieses Muster kann man häufig in E-Mails beobachten. Einige andere häufig benutzte Muster werden ebenfalls unterstützt, z.B. Verweise auf einen Jargon-Begriff:

```
Wenn Sie 'STX' lesen, wissen Sie, dass es eine
Abkürzung für 'strukturierter Text' ist.
```

Die HTML-Ausgabe hierzu sieht wie folgt aus:

```
<p>Wenn Sie <code>STX</code> lesen, wissen Sie, dass es eine
Abkürzung für <code>strukturierter Text</code> ist.</p>
```

Einrückung verwenden

Der vorangegangene Abschnitt handelte von Textkonventionen, mit denen eine Semantik verbunden ist. Bei der Verarbeitung des strukturierten Textes werden aus dieser Semantik bestimmte HTML-Tags erzeugt. Die Einrückung von strukturiertem Text hat ebenfalls eine Semantik. Die grundlegendste hat mit dem Konzept von Überschriften in HTML zu tun. Im folgenden Beispiel steht die Einrückung für eine outline-artige Struktur.

```
Einrückung verwenden
  Der vorangegangene Abschnitt handelte von Textkonventionen, mit denen eine
Semantik verbunden ist. Bei der Verarbeitung des strukturierten Textes werden
aus dieser Semantik bestimmte HTML-Tags erzeugt.
```

Daraus wird folgender HTML-Code produziert:

```
<h1>Einrückung verwenden</h1>

<p>Der vorangegangene Abschnitt handelte von Textkonventionen, mit denen eine
```

```
Semantik verbunden ist. Bei der Verarbeitung des strukturierten Textes werden
aus dieser Semantik bestimmte HTML-Tags erzeugt.</p>
```

In der Einrückung war also eine Bedeutung enthalten, der Absatz war nämlich der Überschrift untergeordnet, und diese Beziehung wird in HTML ausgedrückt. Tatsächlich kann eine solche Outline-Beziehung wie folgt fortgesetzt werden:

```
Einrückung verwenden

  Der vorangegangene Abschnitt handelte von Textkonventionen, mit denen eine
Semantik verbunden ist. Bei der Verarbeitung des strukturierten Textes werden
aus dieser Semantik bestimmte HTML-Tags erzeugt.

  Grundlagen der Einrückung

    In diesem Abschnitt werden wir die Grundlagen der Einrückung untersuchen...
```

Daraus wird folgender HTML-Code produziert:

```
<h1>Einrückung verwenden</h1>

<p>Der vorangegangene Abschnitt handelte von Textkonventionen, mit denen eine
Semantik verbunden ist. Bei der Verarbeitung des strukturierten Textes werden
aus dieser Semantik bestimmte HTML-Tags erzeugt.</p>

<h2>Grundlagen der Einrückung</h2>

<p>In diesem Abschnitt werden wir die Grundlagen der Einrückung
untersuchen...</p>
```

Listen und Listeneinträge

Listen werden in strukturiertem Text ebenfalls unterstützt, darunter ungeordnete, geordnete und beschreibende Listen. Die Konvention zu ungeordneten Listen ist ein häufig benutztes Muster in der Kommunikation mit Texten.

```
In HTML gibt es drei verschiedene Arten von Listen:
- ungeordnete Listen
- geordnete Listen
- beschreibende Listen
```

In strukturiertem Text dürfen Sie Listenelemente mit den vorangestellten Symbolen `*`, `o` und `-` auszeichnen. Das obige Beispiel produziert folgenden HTML-Code:

```
<p>In HTML gibt es drei verschiedene Arten von Listen:</p>
<ul>
<li><p>ungeordnete Listen</p></li>
<li><p>geordnete Listen</p></li>
```

```
<li><p>beschreibende Listen</p></li>
</ul>
```

Die Konvention hinter geordneten Listen sieht in strukturiertem Text wie folgt aus:

```
In HTML gibt es drei verschiedene Arten von Listen:

1. ungeordnete Listen
2. geordnete Listen
3. beschreibende Listen
```

Dabei wird Folgendes produziert:

```
<p>In HTML gibt es drei verschiedene Arten von Listen:</p>
<ol>
<li><p>ungeordnete Listen</p></li>
<li><p>geordnete Listen</p></li>
<li><p>beschreibende Listen</p></li>
</ol>
```

Beschreibende Listen lassen sich genauso leicht mit doppelten Spiegelstrichen angeben. Beispiel:

```
geordnete Listen -- HTML-Viewer wandeln die Listenelemente in eine nummerierte
Folge um

beschreibende Listen -- werden normalerweise für Definitionslisten wie in
Glossaren verwendet
```

Daraus wird folgender HTML-Code:

```
<dl>
<dt>geordnete Listen</dt>
<dd><p>HTML-Viewer wandeln die Listenelemente in eine nummerierte Folge
um</p></dd>
<dt>beschreibende Listen</dt>
<dd><p>werden normalerweise für Definitionslisten wie in Glossaren
verwendet</p></dd>
</dl>
```

Beispiel-Code

Autoren von strukturiertem Text können eine einfache Konvention benutzen, um die mit der Semantik eines `code`-Tags in HTML verbundene unproportionale Darstellung zu erreichen. Der Code

```
Sobald Sie das Dialogfeld sehen, klicken Sie auf den Button 'OK'.
```

wird zum Beispiel mit folgendem HTML-Code dargestellt:

```
<p>Sobald Sie das Dialogfeld sehen, klicken Sie
auf den Button <code>OK</code>.</p>
```

Manchmal benötigen Sie aber auch längere Code-Passagen. Was, wenn Sie z.B. eine Python-Funktion mitten in einem Artikel über Python dokumentieren möchten? Einen solchen Code-Block geben Sie an, indem Sie einen Absatz mit zwei Doppelpunkten beenden (::) und den oder die folgenden Absätze einrücken. Aus

```
In diesem Beispiel wandeln wir Menschenjahre in Hundejahre um:

def hundeJahre(alter):
    """Wandle Menschenjahre in Hundejahre um."""
    return alter * 7
```

wird dieser HTML-Code erzeugt:

```
<p>In diesem Beispiel wandeln wir Menschenjahre in Hundejahre um:</p>

<pre>
def hundeJahre(alter):
    """Wandle Menschenjahre in Hundejahre um."""
    return alter * 7
</pre>
```

Bei dieser Konvention, ein :: am Ende eines Absatzes mit einem eingerückten Block zu kombinieren, wird nicht nur eine Code-Semantik angewendet, sondern der eingerückte Block wird auch *geschützt*. Das heißt, die Schnipsel mit strukturiertem Text bzw. HTML in diesem Artikel werden nicht angetastet und bleiben in ihrer rohen Darstellung. Im folgenden Beispiel werden die Zeichen für kleiner-als, größer-als und das kaufmännische Und geschützt:

```
Hier ist ein HTML-Beispiel:

<html>
<p>Dies ist eine Seite zu Hunden & Katzen.</p>
<html>
```

was das folgenden HTML-Code produziert:

```
<p>Hier ist ein HTML-Beispiel:</p>

<pre>
  <html>
    <p>Dies ist eine Seite zu Hunden & Katzen.</p>
```

```
  <html>
</pre>
```

Hyperlinks

Die vorangegangenen Abschnitte konzentrierten sich auf verschiedene Arten, mit üblichen Textkonventionen eine Präsentationssemantik in HTML zu erzielen. Aber das Web besteht nicht nur aus Text. Verbindungen von Wörtern und Sätzen mit weiteren Informationen sowie die Einbindung von Bildern sind ebenso wichtig. Daher werden Hyperlinks und Bild-Tags auch im strukturierten Textformat unterstützt.

Beginnen wir z.B. mit einem einfachen Hyperlink. Wenn Sie z.B den folgenden Absatz über Python in strukturiertem Text haben:

```
Weitere Informationen zu Python finden Sie auf der
"Python-Website":http://www.python.org.
```

wird daraus Folgendes in HTML:

```
<p>Weitere Informationen zu Python finden Sie auf der
<a href="http://www.python.org">Python-Website</a>.
```

Die Konvention dahinter ist ziemlich einfach:

- Der Verweistext wird in Anführungszeichen gesetzt.
- Dem zweiten Anführungszeichen folgen ein Doppelpunkt und eine URL.
- Nach der URL dürfen Interpunktionszeichen stehen.

Zu dieser Basiskonvention gibt es eine Reihe von Varianten, damit auch relative, mailto- und Bild-URLs verwendet werden können.

A.2.2 Restrukturierter Text

Restrukturierter Text (im Englischen oft als reStructured Text geschrieben) ist eine neuere Version von strukturiertem Text, die einige Probleme beheben soll, die manche Leute mit älteren Versionen hatten. Strukturierter Text versagt nicht nur beim Thema Internationalisierung, sondern produziert ungültige Auszeichnungen. Außerdem ist seine Syntax manchmal etwas schwer verständlich.

Das neue restrukturierte Textformat ist zu einem der Standards für die Dokumentation von Python-Code geworden und wird auch im Docutils-Projekt verwendet. Die Online-Dokumentation ist so gut, dass ich Ihnen ohne Vorbehalte empfehlen möchte, sie unter *http://docutils.sourceforge.net/rst.html* zu lesen.

Es folgt einiges an Material, das von Richard Jones geschrieben wurde.

Einleitung

Als Grundeinheit wird ein Absatz erkannt, also ein Textbrocken, der mit Leerzeilen abgetrennt ist, wobei eine Leerzeile ausreicht. Absätze müssen alle die gleiche Einrückungstiefe haben, d.h., sie müssen links gleich weit weg vom Rand sein. Eingerückte Absätze erscheinen als eingerückte Zitate. Aus dem Code

```
Dies ist ein Absatz, ein recht kurzer noch dazu.

  Aus diesem Absatz wird ein eingerückter Textblock,
  in dem oftmals ein anderer Text zitiert wird.

Und noch ein Absatz.
```

wird folgende Ausgabe:

```
<blockquote>
  <p>Dies ist ein Absatz, ein recht kurzer noch dazu.</p>
    <blockquote>
      Aus diesem Absatz wird ein eingerückter Textblock,
      in dem oftmals ein anderer Text zitiert wird.
    </blockquote>
  <p>Und noch ein Absatz.</p>
</blockquote>
```

Textstile

In Absätzen und anderen Textteilen können Sie Text zusätzlich mit `*kursiv*` als *kursiv* oder mit `**fett**` als **fett** auszeichnen.

Wenn Sie möchten, dass etwas nichtproportional erscheint, verwenden Sie doppelte Rückanführungszeichen (Backklicks ■■■?): ``. Beachten Sie, dass innerhalb dieser Anführungszeichen nichts angetastet wird, d.h. Sternchen usw. bleiben unverändert.

Falls Sie ein »besonderes« Zeichen im Text verwenden möchten, können Sie das normalerweise tun, denn restrukturierter Text ist ziemlich clever. Das Sternchen z.B. funktioniert prima. Wenn Sie wirklich einmal Text mit Sternchen drumherum haben möchten, der *nicht* kursiv gesetzt werden soll, müssen Sie angeben, dass das Sternchen keine Bedeutung hat. Das machen Sie, indem Sie wie folgt einen Rückschrägstrich davor setzen:

```
\*
```

Oder Sie umschließen ihn mit doppelten Rückanführungszeichen wie folgt:

```
``\*``
```

Listen

Es gibt drei verschiedene Sorten von Listen: geordnete, ungeordnete und beschreibende Listen. Bei allen dreien dürfen Sie beliebig viele Absätze, Unterlisten usw. verwenden, solange der linke Absatzrand genauso weit eingerückt ist wie die erste Textzeile im Listenelement.

Listen stehen immer am Anfang eines neuen Absatzes, d.h., sie müssen nach einer Leerzeile stehen. Beginnen Sie eine Zeile mit einer Zahl oder einem Buchstaben, gefolgt von einem Punkt, mit einer runden Klammer danach oder insgesamt in runden Klammern, was immer Ihnen lieber ist. Alle Beispiele in Listing A.1 werden erkannt.

Listing A.1: Beispiele für Punkte

```
1. Zahlen

A. Großbuchstaben
   auch über mehrere Zeilen

   sogar mit zwei Absätzen darin!

a. Kleinbuchstaben

   3. mit einer Unterliste, die mit
      einer anderen Zahl beginnt
   4. aber die Zahlen sollten schon
      die richtige Reihenfolge haben!

I. Große römische Zahlen

i. Kleine römische Zahlen

(1) wieder Zahlen

1) und nochmal
```

Listing A.2 zeigt das Ergebnis, aber beachten Sie, dass die Stile für die verschiedenen Aufzählungslisten nicht von allen Browsern unterstützt werden, d.h., Sie können eventuell nicht den ganzen Effekt sehen.

Listing A.2: Beispielliste

```
<ol class="arabic simple">
<li>Zahlen</li>
</ol>
```

```
<ol class="upperalpha">
<li><p class="first">Großbuchstaben
auch über mehrere Zeilen</p>
<p>sogar mit zwei Absätzen darin!</p>
</li>
</ol>
<ol class="loweralpha simple">
<li>Kleinbuchstaben<ol class="arabic" start="3">
<li>mit einer Unterliste, die mit
einer anderen Zahl beginnt</li>
<li>aber die Zahlen sollten schon
die richtige Reihenfolge haben!</li>
</ol>
</li>
</ol>
<ol class="upperroman simple">
<li>Große römische Zahlen</li>
</ol>
<ol class="lowerroman simple">
<li>Kleine römische Zahlen</li>
</ol>
<ol class="arabic simple">
<li>wieder Zahlen</li>
</ol>
<ol class="arabic simple">
<li>und nochmal</li>
</ol>
```

Statt wie bei geordneten Listen können Sie Listeneinträge auch mit einem Punkt davor haben, indem Sie entweder `-`, `+` oder `*` wie folgt verwenden:

```
* ein Punkt mit "*"

  - eine Unterliste mit "-"

    + eine andere Unterliste

  - ein weiterer Eintrag
```

was folgenden Code erzeugt:

```
<ul class="simple">
<li>ein Punkt mit &amp;quot;*&amp;quot;<ul>
<li>eine Unterliste mit &amp;quot;-&amp;quot;<ul>
<li>eine andere Unterliste</li>
</ul>
</li>
```

```
<li>ein weiterer Eintrag</li>
</ul>
</li>
</ul>
```

Anders als die beiden vorausgegangenen Listenarten bestehen beschreibende Listen aus einem Begriff und seiner Definition. Das Format einer solchen Liste lautet wie folgt:

```
*Was*
  Beschreibende Listen verbinden einen Begriff mit einer Definition.

*Wie*
  Der Begriff besteht aus einem einzeiligen Satz, und die Definition
  besteht aus einem oder mehreren Absätzen, die relativ zum Begriff eingerückt
  sind. Zwischen Begriff und Definition sind keine Leerzeilen erlaubt.
```

Dadurch wird der folgende Code erzeugt:

```
<blockquote>
<dl class="docutils">
<dt><em>Was</em></dt>
<dd>Beschreibende Listen verbinden einen Begriff mit einer Definition.</dd>
<dt><em>Wie</em></dt>
<dd>Der Begriff besteht aus einem einzeiligen Satz, und die Definition besteht
aus einem oder mehreren Absätzen, die relativ zum Begriff eingerückt sind.
Zwischen Begriff und Definition sind keine Leerzeilen erlaubt.</dd>
</dl>
</blockquote>
```

Vorformatierung (Code-Beispiele)

Um einfach einen Brocken Text einzufügen, der unter keinen Umständen jemals angetastet werden soll, beenden Sie den vorhergehenden Absatz mit ::. Der vorformatierte Block gilt als beendet, wenn der Text wieder die Einrückungsebene vor dem vorformatierten Block erreicht. Beispiel:

```
Beispiel::

      Leerräume, Zeilenenden, Leerzeilen und alle Arten
      von Auszeichnungen, z.B. *das* oder \das, werden
      in solchen literalen Blöcken erhalten.
    Schau mal, eine Ebene rauf
    (aber nicht weit genug).

Nicht mehr im Beispiel.
```

Dadurch wird der folgende Code erzeugt:

```
<div class="document">
<p>Beispiel:</p>
<pre class="literal-block">
  Leerräume, Zeilenenden, Leerzeilen und alle Arten
  von Auszeichnungen, z.B. *das* oder \das, werden
  in solchen literalen Blöcken erhalten.
Schau mal, eine Ebene rauf
(aber nicht weit genug).
</pre>
<p>Nicht mehr im Beispiel.</p>
</div>
```

Beachten Sie, dass ein Absatz, der nur aus :: besteht, in der Ausgabe nicht erscheint. Beispiel:

```
::

  Das ist vorformatierter Text, und der
  letzte Absatz mit "*::*" wird entfernt.
```

Das erzeugt den folgenden Code:

```
<pre class="literal-block">
  Das ist vorformatierter Text, und der
  letzte Absatz mit "::" wird entfernt.
</pre>
```

Abschnitte

Mit Überschriften können Sie längere Texte in Abschnitte untergliedern. Sie bestehen aus einer einzelnen Textzeile mit nur einer Schmuckzeile danach oder einer davor und danach. Diese Zeilen bestehen aus Minuszeichen (-----), Gleichheitszeichen (=====), Tilden (~~~~~) oder einem beliebigen nichtalphanumerischen Zeichen, das Ihnen gefällt:

```
- = ` : ' " ~ ^ _ * + # < >
```

Eine Schmuckzeile nur unter einer Überschrift ist verschieden von einer, die vor und nach einer Überschrift verwendet wird, aber sonst aus dem gleichen Zeichen besteht. Diese Zeile davor bzw. danach muss mindestens so lang sein wie die Textzeile. Diese Zeilen sollten Sie konsistent verwenden, da alle Überschriften mit der gleichen Schmuckzeile sich auf der gleichen Ebene befinden. Beispiel:

```
Kapitel 1 Titel
===============

Abschnitt 1.1 Titel
-------------------

Unterabschnitt 1.1.1 Titel
~~~~~~~~~~~~~~~~~~~~~~~~~~

Abschnitt 1.2 Titel
-------------------

Kapitel 2 Titel
===============
```

Daraus wird der folgende Code, der hier in vereinfachtem Pseudo-XML angegeben ist:

```
<section>
  <title>
    Kapitel 1 Titel
  <section>
    <title>
      Abschnitt 1.1 Titel
    <section>
      <title>
        Unterabschnitt 1.1.1 Titel
  <section>
    <title>
      Abschnitt 1.2 Titel
<section>
  <title>
    Kapitel 2 Titel
```

Pseudo-XML verwendet eine Einrückung zur Darstellung der Schachtelung und enthält keine schließenden Tags. Hierbei kann man nicht wie bei den anderen Beispielen eine echte Ausgabe zeigen, weil Abschnitte in Blöcken nicht vorkommen dürfen. Vergleichen Sie als konkretes Beispiel den Quelltext zum englischen Original dieses Textes unter *http://docutils.sourceforge.net/docs/rst/quickstart.html* mit der erzeugten Ausgabe.

Beachten Sie, dass dabei die Abschnittsüberschriften als Ziele von Links fungieren, wenn man ihren Namen benutzt. Um einen Link zur Überschrift *Listen* zu erstellen, schreiben Sie `Listen_`. Wenn in der Überschrift ein Leerzeichen vorkommt, wie in *Restrukturierter Text*, müssen Sie die Überschrift wie folgt in Anführungszeichen setzen: `` `Restrukturierter Text`_ ``.

Bilder

Um ein Bild in Ihrem Dokument einzubinden, verwenden Sie die Direktive `image`. Zum Beispiel wird der folgende Code:

```
.. image:: images/biohazard.png
```

wie folgt zu HTML umgewandelt:

```
<div class="image">
<img alt="images/biohazard.png" src="images/biohazard.png" />
</div>
```

Der Teil `images/biohazard.png` bezeichnet den Dateinamen des Bildes, das an dieser Stelle im Dokument erscheinen soll. Beim Bildformat (Format, Größe usw.) gibt es keine Einschränkungen. Wenn das Bild im HTML-Code erscheinen soll und Sie noch weitere Angaben dazu machen möchten, können Sie das wie folgt tun:

```
.. image:: images/biohazard.png
     :height: 100
     :width: 200
     :scale: 50
     :alt: Alternativtext
```

A.3 Verschiedenes

Es folgen ein paar Tipps, die für Plone-Entwickler hilfreich sein könnten.

A.3.1 Alle globalen Definitionen im Haupt-Template

Tabelle A.2 listet alle globalen Definitionen im Haupt-Template auf, zusammen mit dem Code, der sie definiert, und einer Beschreibung. Wie immer gilt, dass dies keine starre Liste ist, sondern eine, die sich im Laufe der Zeit vermutlich verändern wird. Daher empfehle ich, einen Blick in den Quellcode zu werfen. Diese Zusammenstellung stammt aus `CMFPlone/skins/main_template/globale_defines.pt`.

Diese Definitionen werden vorwiegend in Page Templates benutzt und stellen nützliche Abkürzungen dar. Beispiel:

```
<a href="" tal:attributes="portal_url">Url to the portal</a>
```

Name	Code	Beschreibung
Utool	nocall:here/portal_url;	Das Werkzeug portal_url.
portal	utool/getPortalObject;	Das eigentliche Portal-Objekt.
portal_object	nocall:portal;	Ein weiterer Name für das Portal.
portal_url	utool;	Ein weiterer Name für das Werkzeug portal_url.
mtool	nocall:portal/portal_membership;	Das Werkzeug membership.
gtool	nocall:portal/portal_groups | nothing;	Das Werkzeug groups, falls vorhanden.
dtool	nocall:portal/portal_groupdata | nothing;	Das Werkzeug groups data tool, falls vorhanden.
atool	nocall:portal/portal_actions;	Das Werkzeug portal_actions.
aitool	nocall:portal/portal_actionicons | nothing;	Das Werkzeug portal_actionicons.
putils	nocall:portal/plone_utils;	Das Werkzeug utils.
wtool	nocall:portal/portal_workflow;	Das Werkzeug portal_workflow.
ifacetool	nocall:portal/plone_interface | nothing;	Das Werkzeug portal_interface, falls vorhanden.
portal_title	portal_object/Title;	Der Portaltitel.
object_title	here/Title;	Der Titel des aktuellen Objekts.
member	mtool/getAuthenticatedMember;	Das aktuelle Mitglied.
checkPermission	Nocall:mtool/checkPermission;	Die Funktion checkPermission des membership-Werkzeugs.
membersfolder	mtool/getMembersFolder;	Der Members-Ordner des aktuellen Mitglieds, falls vorhanden.
isAnon	mtool/isAnonymousUser;	Boolescher Wert, falls der Benutzer anonym ist.
actions	python:portal.portal_actions.listFilteredActionsFor(here);	Die Aktionen am aktuellen Ort.
keyed_actions	python:portal.keyFilteredActions(actions);	Die Liste aller Aktionen mit einer ID.
user_actions	Actions/user;	Aktionen für den Benutzer.
workflow_actions	actions/workflow;	Workflow-Aktionen.
folder_actions	actions/folder;	Ordner-Aktionen.
global_actions	actions/global;	Globale Aktionen.

Tabelle A.2: Globale Definitionen im Haupt-Template

Name	Code	Beschreibung
portal_tabs	actions/portal_tabs|nothing;	Portalreiter-Aktionen.
wf_state	python:wtool.getInfo-For(here,'review_state',None);	Workflow-Zustand des aktuellen Objekts.
portal_properties	portal/portal_properties;	Das Objekt portal_properties.
site_properties	portal_properties/site_properties;	Das Objekt site_properties.
ztu	modules/ZTUtils;	Das Modul ZTUtils, ein nützliches Hilfsmodul.
actions	options/actions|actions;	Die über das Template explizit übergebenen Aktionen.
wf_actions	workflow_actions;	Ein weiterer Name für Workflow-Aktionen.
isFolder	python:here.getType-Info().getId() in site_properties.use_folder_tabs;	Boolescher Wert, falls der Kontext ein Ordner ist.
template_id	options/template_id | template/getId | nothing;	Die ID des aktuellen Templates.
slots_mapping	options/slots_mapping| here/prepare_slots| nothing;	Die Slots-Abbildung.
Iterator	python:modules['Products.CMF-Plone'].IndexIterator;	Ein Iterator, verwendet in Templates.
tabindex	python:Iterator(pos=30000);	Der Tabindex-Iterator für Formulare.
here_url	here/absolute_url;	Aktuelle absolute_url.
sl	slots_mapping/left;	Abbildung für linke Slots.
sr	slots_mapping/right;	Abbildung für rechte Slots.
default_language	site_properties/default_language| nothing;	Standardsprache in site_properties.
allowed_types	here/getAllowedTypes;	Erlaubte Inhaltstypen in diesem Ordner.
is_editable	python:here.showEditableBor-der(template_id= template_id, allowed_types= allowed_types, actions=actions);	Gibt an, ob das aktuelle Objekt bearbeitet werden kann und eine grüne Umrandung haben soll.

Tabelle A.2: Globale Definitionen im Haupt-Template (Forts.)

A.3.2 API zu DateTime

Ich habe im ganzen Buch schon `DateTime`-Objekte verwendet, ohne ihre API zu erklären. In Plone werden solche Objekte sehr häufig verwendet, z.B. bei der Suche mit Datumsangaben oder bei deren Darstellung.

Um ein `DateTime`-Objekt zu erstellen, übergeben Sie einen String, der auch als Datumsangabe interpretiert werden kann:

```
>>> from DateTime import DateTime
>>> d = DateTime('2004/12/01')
>>> d.month()
12
```

Auf diesen Datumsobjekten können Sie dann gewisse Operationen ausführen. Um z.B. die Differenz zwischen zwei Datumsobjekten zu bilden, machen Sie Folgendes:

```
>>> x = DateTime('2004/11/02')
>>> z = DateTime('2004/11/30')
>>> z - x
28.0
```

Dieser Abschnitt enthält nur die Kurzfassung des APIs. Die vollständige Dokumentation finden Sie im `DateTime`-Verzeichnis Ihrer Zope-Installation.

`DateTime`-Objekte repräsentieren Zeitpunkte und bieten eine Schnittstelle zu ihrer Darstellung, ohne dass dabei der absolute Wert des Objekts verändert wird.

Sie können `DateTime`-Objekte aus einer Vielzahl von Strings oder aus numerischen Daten erzeugen, oder Sie erzeugen sie aus anderen `DateTime`-Objekten. Sie können ihre Darstellung an viele Zeitzonen anpassen, und Sie können `DateTime`-Objekte für eine gegebene Zeitzone erstellen.

Außerdem verfügen `DateTime`-Objekte zum Teil über ein zahlenähnliches Verhalten:

- Zwei `DateTime`-Objekte können voneinander subtrahiert werden, um die Zeitdauer in Tagen dazwischen zu beschreiben.
- Ein `DateTime`-Objekt kann zu einer positiven oder negativen Zahl addiert werden, um ein neues `DateTime`-Objekt plus der angegebenen Zahl von Tagen zu erzeugen.
- Eine positive oder negative Zahl kann zu einem `DateTime`-Objekt addiert werden, um ein neues `DateTime`-Objekt plus der angegebenen Zahl von Tagen zu erzeugen.

- Eine positive oder negative Zahl kann von einem `DateTime`-Objekt subtrahiert werden, um ein neues `DateTime`-Objekt minus der angegebenen Zahl von Tagen zu erzeugen.

Sie können `DateTime`-Objekte zu der Anzahl von Tagen seit dem 1. Januar 1901 im Integer-, Long- oder Float-Typ konvertieren, indem Sie die normalen Funktionen `int`, `long` und `float` verwenden. Beachten Sie aber, dass diese Funktionen die Anzahl der Tage in der Greenwich Mean Time (GMT) angeben und nicht in der lokalen Zeitzone Ihres Rechners. Außerdem bieten `DateTime`-Objekte auch einen Zugriff auf ihren Wert in einem Float-Format, der mit dem `time`-Modul in Python verwendet werden kann, vorausgesetzt, dass dieser Wert im Zeitraum des auf der Epoche basierenden `time`-Moduls liegt.

Ein `DateTime`-Objekt sollte als unveränderlich betrachtet werden. Alle Umwandlungs- und andere Operationen geben ein neues `DateTime`-Objekt zurück und verändern nicht das gegebene Objekt.

Um ein `DateTime`-Objekt zu erzeugen, übergeben Sie einen String, der ein gültiges Datum beschreibt. Wenn Sie kein Argument übergeben, wird ein Objekt für den aktuellen Zeitpunkt erzeugt. Der Wert eines `DateTime`-Objekts wird immer in absoluter UTC-Zeit verwaltet und wird im Kontext einer Zeitzone dargestellt, die auf den Argumenten basiert, die bei der Erzeugung des Objekts angegeben wurden.

Tabelle A.3 beschreibt alle Methoden, die bei einem `DateTime`-Objekt aufgerufen werden können.

Methode	Beschreibung
`aMonth()`	Gibt den abgekürzten Monatsnamen zurück.
`pCommon()`	Gibt eine String-Darstellung des Objektwerts im Format Jan. 13, 2005 1:41 pm zurück.
`minute()`	Gibt die Minute zurück.
`isLeapYear()`	Gibt zurück, ob das aktuelle Jahr (im Kontext der Zeitzone des Objekts) ein Schaltjahr ist.
`pMonth()`	Gibt den (mit Punkt) abgekürzten Monatsnamen zurück.
`DayOfWeek()`	Kompatibilität: siehe `Day()`.
`Day_()`	Kompatibilität: siehe `pDay()`.
`isCurrentDay()`	Gibt im Kontext der Zeitzone dieses Objekts zurück, ob dieses Objekt einen Zeitpunkt am heutigen Tag darstellt.
`Mon()`	Kompatibilität: siehe `aMonth()`.
`hour()`	Gibt die Stunde in 24-Stunden-Darstellung zurück.

Tabelle A.3: Verfügbare Methoden auf einem DateTime-Objekt

Methode	Beschreibung
`Date()`	Gibt den Datums-String zu dem Objekt zurück.
`aCommonZ()`	Gibt eine String-Darstellung des Objektwerts im Format Jan 13, 2005 1:40 pm GMT+1 zurück.
`fCommonZ()`	Gibt eine String-Darstellung des Objektwerts im Format January 13, 2005 1:42 pm GMT+1 zurück.
`isCurrent-Year()`	Gibt im Kontext der Zeitzone dieses Objekts zurück, ob dieses Objekt einen Zeitpunkt im laufenden Jahr darstellt.
`AMPMMinutes()`	Gibt den Zeit-String für ein Objekt ohne Sekunden mit Angabe von am/pm zurück.
`dd()`	Gibt den Tag als String mit zwei Ziffern zurück.
`TimeMinutes()`	Gibt den Zeit-String für ein Objekt ohne Sekunden in 24-Stunden-Darstellung zurück.
`h_24()`	Gibt die Stunde in 24-Stunden-Darstellung zurück.
`isPast()`	Gibt zurück, ob dieses Objekt einen Zeitpunkt vor dem Zeitpunkt dieses Aufrufs darstellt.
`dow()`	Gibt den Wochentag als Integer zurück, wobei Sonntag gleich 0 ist.
`isFuture()`	Gibt zurück, ob dieses Objekt einen Zeitpunkt nach dem Zeitpunkt dieses Aufrufs darstellt.
`pCommonZ()`	Gibt eine String-Darstellung des Objektwerts im Format Jan. 13, 2005 2:01 pm GMT+1 zurück.
`timezone()`	Gibt die Zeitzone des Objekts zurück.
`h_12()`	Gibt die Stunde in 12-Stunden-Darstellung zurück.
`PreciseTime()`	Gibt eine genaue String-Darstellung des Objekts zurück.
`isCurrent-Minute()`	Gibt im Kontext der Zeitzone dieses Objekts zurück, ob dieses Objekt einen Zeitpunkt in der laufenden Minute darstellt.
`rfc822()`	Gibt den Zeitpunkt im Format RFC 822 zurück.
`equalTo(t)`	Vergleicht dieses `DateTime`-Objekt mit einem anderen `DateTime`-Objekt oder einer Fließkommazahl, wie sie Pythons `time`-Modul zurückgibt. Gibt zurück, ob das Objekt einen Zeitpunkt beschreibt, der identisch mit dem angegebenen Objekt ist.
`yy()`	Gibt das Kalenderjahr als String mit zwei Ziffern zurück.
`mm()`	Gibt den Monat als String mit zwei Ziffern zurück.
`Mon_()`	Kompatibilität: siehe `pMonth()`.
`toZone(z)`	Gibt ein `DateTime`-Objekt mit dem aktuellen Wert in der angegebenen Zeitzone zurück.

Tabelle A.3: Verfügbare Methoden auf einem DateTime-Objekt (Forts.)

Methode	Beschreibung
`earliestTime()`	Gibt im Kontext der Zeitzone des Objekts ein neues `DateTime`-Objekt zurück, das den frühestmöglichen Zeitpunkt darstellt (in ganzen Sekunden), der immer noch auf den Tag des aktuellen Objekts fällt.
`aDay()`	Gibt den abgekürzten Wochentag zurück.
`dayOfYear()`	Gibt Kontext der Zeitzone des Objekts den Tag des Jahres zurück.
`latestTime()`	Gibt im Kontext der Zeitzone des Objekts ein neues `DateTime`-Objekt zurück, das den spätestmöglichen Zeitpunkt darstellt (in ganzen Sekunden), der immer noch auf den Tag des aktuellen Objekts fällt.
`notEqualTo(t)`	Vergleicht dieses `DateTime`-Objekt mit einem anderen `DateTime`-Objekt oder einer Fließkommazahl, wie sie Pythons `time`-Modul zurückgibt. Gibt zurück, ob das Objekt einen Zeitpunkt beschreibt, der verschieden vom angegebenen Objekt ist.
`PreciseAMPM()`	Gibt eine genaue String-Darstellung des Objekts mit Angabe von am/pm zurück.
`day()`	Gibt den Tag als ganze Zahl zurück.
`timeTime()`	Gibt den Zeitpunkt als Fließkommazahl in UTC zurück, dem Format, das Python im `time`-Modul verwendet. Beachten Sie, dass man mit `DateTime` Zeitpunkte erstellen kann, die keine Entsprechung im Modul `time` haben, und in solchen Fällen wird der Fehler `DateTimeError` ausgelöst. Im Allgemeinen muss der Wert eines `DateTime`-Objekts zwischen dem 1. Januar 1970 (oder der Epoche auf Ihrem lokalen Rechner) und dem 1. Januar 2038 liegen, um einen gültigen Wert im Stil von `time.time()` zu erzeugen.
`ampm()`	Gibt den passenden Zeitzusatz (am bzw. pm) zurück.
`greaterThan(t)`	Vergleicht dieses `DateTime`-Objekt mit einem anderen `DateTime`-Objekt oder einer Fließkommazahl, wie sie z.B. von Pythons `time`-Modul erzeugt wird. Gibt zurück, ob das Objekt einen Zeitpunkt darstellt, der größer als das angegebene Objekt ist.
`month()`	Gibt den Monat des Objekts als ganze Zahl zurück.
`AMPM()`	Gibt den Zeit-String zu einem Objekt auf die nächste Sekunde genau zurück.
`second()`	Gibt die Sekunde zurück.
`parts()`	Gibt ein Tupel mit dem Kalenderjahr, Monat, Tag, der Stunde, Minute, Sekunde und Zeitzone des Objekts zurück.
`greaterThanE-qualTo(t)`	Vergleicht dieses `DateTime`-Objekt mit einem anderen `DateTime`-Objekt oder einer Fließkommazahl, wie sie z.B. von Pythons `time`-Modul erzeugt wird. Gibt zurück, ob das Objekt einen Zeitpunkt darstellt, der größer oder gleich dem angegebenen Objekt ist.

Tabelle A.3: Verfügbare Methoden auf einem DateTime-Objekt (Forts.)

Methode	Beschreibung
`lessThan-EqualTo(t)`	Vergleicht dieses `DateTime`-Objekt mit einem anderen `DateTime`-Objekt oder einer Fließkommazahl, wie sie z.B. von Pythons `time`-Modul erzeugt wird. Gibt zurück, ob das Objekt einen Zeitpunkt darstellt, der kleiner oder gleich dem angegebenen Objekt ist.
`isCurrent-Hour()`	Gibt im Kontext der Zeitzone des Objekts zurück, ob dieses Objekt einen Zeitpunkt darstellt, der in die laufende Stunde fällt.
`aCommon()`	Gibt eine String-Darstellung des Objektwertes im Format Jan 13, 2005 4:22 pm zurück.
`dow_1()`	Gibt den Wochentag als ganze Zahl zurück, wobei, anders als bei der Methode `dow`, Sonntag gleich 1 ist.
`Day()`	Gibt den vollen Namen des Wochentages zurück.
`fCommon()`	Gibt eine String-Darstellung des Objektwertes im Format January 13, 2005 4:29 pm zurück.
`Month()`	Gibt den vollen Monatsnamen zurück.
`isCurrent-Month()`	Gibt im Kontext der Zeitzone des Objekts zurück, ob dieses Objekt einen Zeitpunkt darstellt, der in den laufenden Monat fällt.
`year()`	Gibt das Kalenderjahr des Objekts zurück.
`lessThan(t)`	Vergleicht dieses `DateTime`-Objekt mit einem anderen `DateTime`-Objekt oder einer Fließkommazahl, wie sie z.B. von Pythons `time`-Modul erzeugt wird. Gibt zurück, ob das Objekt einen Zeitpunkt darstellt, der kleiner als das angegebene Objekt ist.
`Time()`	Gibt den Zeit-String für ein Objekt bis auf die nächste Sekunde genau zurück.
`pDay()`	Gibt den (mit Punkt) abgekürzten Namen des Wochentages zurück.

Tabelle A.3: Verfügbare Methoden auf einem DateTime-Objekt (Forts.)

Tabelle A.4 beschreibt alle eingebauten Methoden von `DateTime`-Objekten.

Methode	Beschreibung
`` `dateTime` ``	Wandelt ein `DateTime`-Objekt in einen String um, der wie ein Python-Ausdruck aussieht.
`str(dateTime)`	Wandelt ein `DateTime`-Objekt in einen String um.
`cmp(dateTime, other)`	Vergleicht ein `DateTime`-Objekt mit einem anderen oder mit einer Fließkommazahl, wie `time.time()` sie zurückgibt.
`hash(dateTime)`	Berechnet den Hash-Wert eines `DateTime`-Objekts.

Tabelle A.4: Eingebaute Methoden von DateTime-Objekten

Tabelle A.5. beschreibt generische Dienste, die `DateTime` unterstützt.

Methode	Beschreibung
`dateTime + anderes`	Ein `DateTime`-Objekt darf zu einer Zahl addiert werden und umgekehrt. Zwei `DateTime`-Objekte können aber nicht addiert werden.
`dateTime - anderes`	Von einem `DateTime`-Objekt kann man entweder ein anderes `DateTime`-Objekt oder eine Zahl subtrahieren, aber ein `DateTime`-Objekt kann nicht von einer Zahl subtrahiert werden.
`anderes + dateTime`	Addiert `dateTime` zu `anderes`. Ein `DateTime`-Objekt kann zu einer Zahl addiert werden und umgekehrt, aber zwei `DateTime`-Objekte können nicht addiert werden.
`int(dateTime)`	Wandelt in eine Integer-Anzahl von Tagen seit dem 1. Januar 1901 (GMT) um.
`long(dateTime)`	Wandelt in eine Long-Anzahl von Tagen seit dem 1. Januar 1901 (GMT) um.
`float(dateTime)`	Wandelt in eine Float-Anzahl von Tagen seit dem 1. Januar 1901 (GMT) um.

Tabelle A.5: Generische DateTime-Funktionen

A.3.3 Workflows in Python schreiben

Dieser Abschnitt behandelt die API zum Schreiben eines Workflows in Python, damit Sie nicht über das Web entwickeln müssen. Die folgenden Abschnitte enthalten eine verkürzte Liste der Methoden, die Sie benötigen, um einen Workflow zu erstellen. Um eine umfassende Liste zu erhalten, müssen Sie den Quellcode lesen, den Sie im Verzeichnis `Products/DCWorkflow` Ihrer Plone-Instanz finden.

Das Schreiben eines Workflows beginnt mit dem Erstellen einer Definition. Diese kann man wie folgt importieren:

```
from Products.DCWorkflow.DCWorkflow import DCWorkflowDefinition
```

Erzeugen Sie eine neue Instanz dieser Definition, und geben Sie ihr eine ID:

```
wf = DCWorkflowDefinition(id)
```

Nun haben Sie einen Workflow, den Sie manipulieren können. Im Folgenden ist `wf` die Workflow-Definition, die ich gerade erzeugt habe.

Zustände

Zustände verfügen über die folgenden Methoden:

- **addState(Id)**: Fügt einen Zustand mit der gegebenen ID hinzu. Zustände müssen Sie erst hinzufügen, bevor Sie sie manipulieren können.

- **setInitialState(Id)**: Setzt den Ausgangszustand für diesen Workflow.
- **setProperties([key=value,...])**: Eine Reihe von Schlüssel/Wert-Paaren, die auf Eigenschaften eines Zustands abgebildet werden. Um z.B. die Übergänge auf Veröffentlichen und Zurückweisen zu setzen, verwenden Sie `setProperties(transitions=('publish', 'reject'))`..
- **setPermission(recht, akquirieren, (rolle, [rolle...]))**: Der Name des Rechts, ob `akquirieren` aktiviert ist oder nicht (ein boolescher Wert) und dann ein Tupel aller Rollen, für die das gilt.

Um auf diese Methoden zuzugreifen, verwenden Sie das `states`-Objekt im Workflow-Objekt. Verwenden Sie z.B. `wf.states.addState('new')`.

Übergänge

Übergänge verfügen über die folgenden Methoden:

- **addTransition(id)**: Fügt einen Übergang mit der gegebenen ID hinzu. Übergänge müssen Sie erst hinzufügen, bevor Sie sie manipulieren können.
- **setProperties([key=value,...])**: Eine Reihe von Schlüssel/Wert-Paaren, die auf Eigenschaften eines Übergangs abgebildet werden.

Um auf diese Methoden zuzugreifen, verwenden Sie das `transitions`-Objekt im Workflow-Objekt. Verwenden Sie z.B. `wf.transitions.addTransition('reject')`.

Variablen

Variablen verfügen über die folgenden Methoden:

- **addVariable(id)**: Fügt eine Variable mit der gegebenen ID hinzu. Variablen müssen Sie erst hinzufügen, bevor Sie sie manipulieren können.
- **setStateVar(id)**: Setzt den Variablennamen, der normalerweise `review_state` ist.
- **setProperties([key=value,...])**: Eine Reihe von Schlüssel/Wert-Paaren, die auf Eigenschaften einer Variablen abgebildet werden.

Um auf diese Methoden zuzugreifen verwenden Sie das `variables`-Objekt im Workflow-Objekt. Verwenden Sie z.B. `wf.transitions.addVariable('action')`.

Sonstige

Worklists verhalten sich hier wie Übergänge, Variablen und Zustände. Um eine Worklist hinzuzufügen, rufen Sie `addWorklist` auf. Eine weitere nützliche Methode besteht darin, Rechte zu der Liste der verwalteten Rechte für den Workflow hinzuzufügen. Das bewerkstelligt man mit der Methode `addManagedPermission(permission name)`.

B Code-Listings

Dieser Anhang enthält Listings von Code, wie er im restlichen Buch verwendet wird. Manche dieser Scripten, die extern ausgeführt werden, verlangen, dass bei Ihnen das Verzeichnis `/Zope/lib/python` auf Ihrem Python-Pfad liegt. In Anhang A wird das näher ausgeführt. All diese Scripten werden im Buch detailliert beschrieben.

B.1 Kapitel 5

Die folgenden Code-Listings stammen aus Kapitel 5.

B.1.1 Page Template: test_context

Listing B.1 zeigt alle Variablen, die in Page Templates verfügbar sind. Es kann verwendet werden, um nützliche Informationen bei der Fehlersuche auszugeben.

Listing B.1: test_context

```
<html>
  <head />
  <body>
    <h1>Debug information</h1>
  <h2>CONTEXTS</h2>
  <ul>
    <tal:block
        tal:repeat="item CONTEXTS">
    <li
        tal:condition="python: item != 'request'"
        tal:define="context CONTEXTS;">
            <b tal:content="item" />
            <span tal:replace="python: context[item]" />
    </li>
    </tal:block>
  </ul>
  <h2>REQUEST</h2>
```

```
  <p tal:replace="structure request" />
  </body>
</html>
```

B.1.2 Page Template: user_info (1)

`user_info (1)` ist ein rohes Page Template, das nicht dem Plone-Stil folgt und Benutzerinformationen ausgibt, wie in Listing B.2 zu sehen ist. Um dieses Template zu benutzen, müssen Sie die Benutzer-ID, die Sie untersuchen möchten, an den Abfrage-String übergeben. Beispiel: `user_info?userName=andy`.

Listing B.2: user_info (1)

```
<html>

<body>
<div
  tal:omit-tag=""
  tal:define="
    userName request/userName|nothing;
    userObj python: here.portal_membership.getMemberById(userName);
    getPortrait nocall: here/portal_membership/getPersonalPortrait;
    getFolder nocall: here/portal_membership/getHomeFolder
    ">
     <p tal:condition="not: userName">
        No username selected.
     </p>
     <p tal:condition="not: userObj">
       That username does not exist.
     </p>

     <table tal:condition="userObj">
       <tr>
         <td>
          <img src=""
            tal:replace="structure python: getPortrait(userName)" />
         </td>
         <td>
           <ul>
           <li>
             <i>Username:</i>
             <span tal:replace="userName" />
           </li>
           <li>
             <i>Full name:</i>
             <span tal:replace="userObj/fullname" />
```

```
            </li>
            <li
              tal:define="home python: getFolder(userName)"
              tal:condition="home">
              <i>Home folder:</i>
              <a href=""
                 tal:attributes="href home/absolute_url"
                 tal:content="home/absolute_url">Folder</a>
            </li>
            <li>
              <i>Email:</i>
              <a href=""
                  tal:define="email userObj/email"
                  tal:attributes="href string:mailto:$email"
                  tal:content="email">Email</a>
            </li>
            <li>
              <i>Last login time:</i>
              <span tal:replace="userObj/last_login_time" />
            </li>
          </ul>
        </td>
      </tr>
    </table>

</div>
</body>
</html>
```

B.2 Kapitel 6

Die folgenden Code-Listings stammen aus Kapitel 6.

B.2.1 Page Template: user_info (2)

Listing B.3 enthält eine verfeinerte Version des Page Templates `user_info` aus Kapitel 5. Den Benutzernamen müssen Sie nicht übergeben, da es über alle Mitglieder Ihrer Site iteriert.

Listing B.3: user_info (2)

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-US"
      lang="en-US"
      i18n:domain="plone"
      metal:use-macro="here/main_template/macros/master">
```

```
<body>
<div metal:fill-slot="main">
  <tal:block
    tal:define="
     getPortrait nocall: here/portal_membership/getPersonalPortrait;
     getFolder nocall: here/portal_membership/getHomeFolder
     ">
    <table>
     <tal:block
      tal:repeat="userObj here/portal_membership/listMembers">
       <metal:block
         metal:use-macro="here/user_section/macros/userSection" />
     </tal:block>
    </table>
 </tal:block>
</div>
</body>
</html>
```

B.2.2 Page Template: user_section

Listing B.4 enthält ein Page Template-Makro zum vorherigen Page Template `user_info` und zeigt bei jedem Aufruf die einzelnen Benutzerangaben.

Listing B.4: user_section

```
<div metal:define-macro="userSection"
 tal:define="userName userObj/getUserName">
 <tr>
   <td>
     <img src=""
       tal:replace="structure python: getPortrait(userName)" />
   </td>
   <td tal:define="prop nocall: userObj/getProperty">
     <ul>
       <li>
         <i>Username:</i>
         <span tal:replace="userName" />
       </li>
       <li>
         <i>Full name:</i>
         <span tal:replace="python: prop('fullname')" />
       </li>
       <li
         tal:define="home python: getFolder(userName)"
```

```
        tal:condition="home">
        <a href=""
        tal:attributes="href home/absolute_url">Home Folder</a>
      </li>
      <li>
        <i>Email:</i>
        <a href=""
          tal:define="email python: prop('email')"
          tal:attributes="href string:mailto:$email"
          tal:content="email">Email</a>
      </li>
      <li>
        <i>Last login time:</i>
         <span tal:replace="python: prop('last_login_time')" />
      </li>
    </ul>
  </td>
 </tr>
</div>
```

B.2.3 Script (Python): google_ad_portlet

Listing B.5 enthält ein Page Template, das Google-Anzeigen in einem Portlet darstellt.

Listing B.5: google_ad_portlet

```
<div metal:define-macro="portlet">
    <div class="portlet">
<script type="text/javascript"><!--
google_ad_client = "yourUniqueValue";
google_ad_width = 120;
google_ad_height = 600;
google_ad_format = "120x600_as";
//--></script>
<script type="text/javascript"
  src="http://pagead2.googlesyndication.com/pagead/show_ads.js">
</script>

   </div>
</div>
```

B.2.4 Script (Python): recently_changed

Listing B.6 enthält ein Script, das alle ihm übergebenen Objekte untersucht und dann herausfindet, was es Neues gibt. Die Objekte müssen mit dem Parameter `objects` übergeben werden.

Listing B.6: recently_changed

```
##title=recentlyChanged
##parameters=objects
from DateTime import DateTime

now = DateTime()
difference = 5 # in Tagen
result = []

for object in objects:
  diff = now - object.bobobase_modification_time()
  if diff < difference:
    dct = {"object":object,"diff":int(diff)}
    result.append(dct)

return result
```

B.2.5 Externe Methode: readFile

Listing B.7 zeigt ein Beispiel für eine externe Methode, die eine Datei liest.

Listing B.7: readFile

```
def readFile(self):
   fh = open(r'c:\Program Files\Plone\Data\Extensions\README.txt', 'rb')
   data = fh.read()
   return data
```

B.2.6 Python-Script: zpt.py

Listing B.8 enthält ein Script, das völlig unabhängig von Plone ist und das eine Syntax-Prüfung eines Page Templates vornimmt. Das ist hilfreich bei der Fehlersuche von Page Templates, die außerhalb von Plone geschrieben werden.

Listing B.8: zpt.py

```
#!/usr/bin/python
from Products.PageTemplates.PageTemplate import PageTemplate
import sys

def test(file):
    raw_data = open(file, 'r').read()
    pt = PageTemplate()
    pt.write(raw_data)
    if pt._v_errors:
        print "*** Error in:", file
        for error in pt._v_errors[1:]:
            print error

if __name__=='__main__':
    if len(sys.argv) < 2:
        print "python check.py file [files...]"
        sys.exit(1)
    else:
        for arg in sys.argv[1:]:
            test(arg)
```

B.2.7 Page Template: feedbackForm

Listing B.9 zeigt das Formular, in dem Benutzer Feedback eingeben können.

Listing B.9: feedbackForm

```
<!--
$Id: feedbackForm.cpt,v 1.2 2004/01/05 05:07:20 andy Exp $
Copyright: ClearWind Consulting Ltd
License: http://www.clearwind.ca/license
-->
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-US"
lang="en-US" i18n:domain="plone"
metal:use-macro="here/main_template/macros/master">
  <body>
    <div metal:fill-slot="main"
    tal:define="errors options/state/getErrors;">
      <p>Please send us any feedback you might have about the
      site.</p>
      <form method="post" tal:attributes="action template/id;">
        <fieldset>
          <legend class="legend"
          i18n:translate="legend_feedback_form">Website
```

```
        Feedback</legend>
        <div class="field"
        tal:define="error_email_address errors/email_address|nothing;"
             tal:attributes="class python:test(error_email_address, 'field ↩
                         error', 'field')">

          <label i18n:translate="label_email_address">Your email
          address</label>
          <span class="fieldRequired" title="Required"
          i18n:attributes="title"
          i18n:translate="label_required">(Required)</span>
          <div class="formHelp"
          i18n:translate="label_email_address_help">Enter your
          email address.</div>
          <div tal:condition="error_email_address">
            <tal:block i18n:translate=""
            content="error_email_address">Error</tal:block>
          </div>
          <input type="text" name="email_address"
          tal:define="user context/portal_membership/getAuthenticatedMember; ↩
                      email user/email|nothing"
          tal:attributes="tabindex tabindex/next; value ↩
                          request/email_address|email|nothing" />
        </div>
        <div class="field">
          <label i18n:translate="label_feedback_comments">
          Comments</label>
          <div class="formHelp" id="label_feedback_comments_help"
          i18n:translate="label_feedback_comments_help">Enter the
          comments you have about the site.</div>
          <textarea name="comments" rows="10"
          tal:content="request/comments|nothing"
          tal:attributes="tabindex tabindex/next;" />
        </div>
        <div class="formControls">
          <input class="context" type="submit" tabindex=""
          name="form.button.Submit" value="Submit"
          i18n:attributes="value"
          tal:attributes="tabindex tabindex/next;" />
        </div>
      </fieldset>
      <input type="hidden" name="form.submitted" value="1" />
    </form>
  </div>
 </body>
</html>
```

B.2.8 Controller Python-Script: sendEmail

Listing B.10 zeigt das Steuerscript, das dem Benutzer die E-Mail schickt.

Listing B.10: sendEmail

```
#!/usr/bin/python
#$Id: sendEmail.cpy,v 1.2 2004/01/05 05:48:11 andy Exp $
#Copyright: ClearWind Consulting Ltd
#License: http://www.clearwind.ca/license

mhost = context.MailHost
emailAddress = context.REQUEST.get('email_address')
administratorEmailAddress = context.email_from_address
comments = context.REQUEST.get('comments')

# the message format, %s will be filled in from data
message = """
From: %s
To: %s
Subject: Website Feedback

%s

URL: %s
"""

# format the message
message = message % (
    emailAddress,
    administratorEmailAddress,
    comments,
    context.absolute_url())

mhost.send(message)

screenMsg = "Comments sent, thank you."
state.setKwargs( {'portal_status_message':screenMsg} )
return state
```

B.2.9 Controller Python-Script: validEmail

Listing B.11 zeigt das Script, das die E-Mail validiert.

Listing B.11: validEmail

```
#$Id: validEmail.vpy,v 1.1 2003/10/31 16:59:47 andy Exp $
#Copyright: ClearWind Consulting Ltd
#License: http://www.clearwind.ca/license

email = context.REQUEST.get('email_address', None)

if not email:
    state.setError('email_address', 'An email address is required',
new_status='failure')

if state.getErrors():
    state.set(portal_status_message='Please correct the errors shown.')

return state
```

B.3 Kapitel 7

Die folgenden Code-Listings stammen aus Kapitel 7.

B.3.1 Script (Python): setSkin

Falls es in Form einer Zugriffsregel in Ihrer Site zugewiesen wurde, ändert das Script in Listing B.12 alle Anfragen auf `intern.einesite.org` so ab, dass die Skin `Plone Default` verwendet wird. Sonst wird die Skin `Custom Chrome` verwendet.

Listing B.12: setSkin

```
##title=Skin changing script
##parameters=
req = context.REQUEST
if req['SERVER_URL'].find('internal.somesite.org') > -1:
    context.changeSkin("Plone Default")
context.changeSkin("Custom Chrome")
```

B.3.2 CSS: ploneCustom.css

Listing B.13 zeigt das Cascading Stylesheet, das auf der NASA-Mars-Site verwendet wird.

Listing B.13: ploneCustom.css

```
body {
    background: #343434;
}

#visual-portal-wrapper {
    width: 680px;
    margin: 1em auto 0 auto;
}

#portal-top {
    background: url("http://mars.telascience.org/header.jpg") transparent no-
repeat;
    padding: 162px 0 0 0;
    position: relative;
}

#portal-logo {
    background: transparent;
    background-image: none;
    margin: 0;
    position: absolute;
    top: 130px;
    left: 5px;
    z-index: 20;
}

#portal-logo a {
    padding-top: 25px;
    height /**/: 25px;
    width: 375px;
}

#portal-globalnav {
   background: url("http://mars.telascience.org/listspacer.gif") transparent;
    padding: 0;
    height: 21px;
    border: 0;
    margin: 0 0 1px 6px;
    clear: both;
}

#portal-globalnav li {
    display: block;
    float: left;
    height: 21px;
```

```
    background: url("http://mars.telascience.org/liststart.gif") transparent no-
      repeat;
    padding: 0 0 0 33px;
    margin: 0 0.5em 0 0;
}

#portal-globalnav li a {
    display: block;
    float: left;
    height: 21px;
    background: url("http://mars.telascience.org/listitem.gif") transparent
      right top;
    padding: 0 33px 0 0;
    border: 0;
    line-height: 2em;
    color: black;
    font-size: 90%;
    margin: 0;
}

#portal-globalnav li a:hover,
#portal-globalnav li.selected a {
    background-color: transparent;
    border: 0;
    color: #444;
}

#portal-personaltools {
    clear: both;
    margin-left: 6px;
    border-top-color: #776a44;
    border-top-style: solid;
    border-top-width: 1px;
}

#portal-breadcrumbs {
    clear: both;
}

#portal-breadcrumbs,
#portal-columns,
.documentContent {
    background: white;
    margin-left: 6px;
}

.documentContent {
```

```
   margin: 0;
   font-size: 100%;
}

.screenshotThumb {
  float:right;
}

#portal-footer {
    margin: -1px 0 0 6px;
    padding: 0.8em 0;
    border: 1px solid #ddd;
    border-style: solid none none none;
    background: white;
    color: #666;
    font-size: 90%;
}

#portal-footer a {
    color: #333;
    text-decoration: underline;
}

dt {
    color: #ECA200;
}

.documentDescription {
    font-size: 110%;
}

#portal-breadcrumbs img {
    display: none;
}

li.reqlist {
    margin-top: 0;
    margin-bottom: 0;
}
```

B.4 Kapitel 8

Die folgenden Code-Listings stammen aus Kapitel 8.

B.4.1 Script (Python): mail.py

Listing B.14 enthält ein Script, das immer dann eine E-Mail an Benutzer schickt, wenn sich ein Objekt geändert hat.

Listing B.14: mail.py

```
##parameters=state_change
# the objects we need
object = state_change.object
mship = context.portal_membership
mhost = context.MailHost
administratorEmailAddress = context.email_from_address

# the message format, %s will be filled in from data
message = """
From: %s
To: %s
Subject: New item submitted for approval - %s

%s

URL: %s
"""

for user in mship.listMembers():
    if "Reviewer" in mship.getMemberById(user.id).getRoles():
        if user.email:
            msg = message % (
                 administratorEmailAddress,
                 user.email,
                 object.TitleOrId(),
                 object.Description(),
                 object.absolute_url()
            )
        mhost.send(msg)
```

B.5 Kapitel 9

Die folgenden Code-Listings stammen aus Kapitel 9.

B.5.1 Externe Methode: importUsers

Listing B.15 zeigt eine externe Methode zum Importieren von Benutzern aus einer .csv-Datei im Dateisystem.

Listing B.15: importUsers

```
#!/usr/bin/python
#$Id$
#Copyright: ClearWind Consulting Ltd
#License: http://www.clearwind.ca/license
import csv

fileName = "/var/zope.zeo/Extensions/test.csv"

def importUsers(self):
    reader = csv.reader(open(fileName, "r"))
    pm = self.portal_membership
    pr = self.portal_registration
    pg = self.portal_groups
    out = []
    ignoreLine = 1

    for row in reader:
        # ignore blank lines
        if not row: continue

        if ignoreLine:
            continue
            ignoreLine = 0

        # check we have exactly 4 items
        assert len(row) == 4
        id, name, email, groups = row

        password = pr.generatePassword()

        try:
            pr.addMember(id = id,
                password = password,
                roles = ["Member",],
                properties = {
                    'fullname': name,
                    'username': id,
                    'email': email,
                    }
```

```
                )
            for groupId in groups.split(','):
                group = pg.getGroupById(groupId)
                group.addMember(id)

            out.append("Added user %s" % id)

        except ValueError:
            out.append("Skipped %s" % id)

    return "\n".join(out)
```

B.5.2 Externe Methode: fixUsers

Listing B.16 zeigt ein Script, das für alle Benutzer Eigenschaften von Epoz setzt.

Listing B.16: fixUsers

```
def fixUsers(self):
    pm = self.portal_membership
    members = pm.listMemberIds()

    out = []
    for member in members:
        # now get the actual member
        m = pm.getMemberById(member)
        # get the editor property for that member
        p = m.getProperty('wysiwyg_editor', None)

        out.append("%s %s" % (p, member))
        if p is not None and p != 'Epoz':
            m.setMemberProperties({'wysiwyg_editor': 'Epoz',})
            out.append("Changed property for %s" % member)
    return "\n".join(out)
```

B.5.3 Externe Methode: getGroups

Listing B.17 zeigt ein Script, das zuerst den Erzeuger eines Objekts holt. Dann holt es alle Benutzer, die in der gleichen Gruppe sind wie dieser Erzeuger.

Listing B.17: getGroups

```
##parameters=object=None
# object is the object to find all the members of the same group for
users = []
```

```
# get the creator
userName = object.Creator()
user = context.portal_membership.getMemberById(userName)
pg = context.portal_groups

# loop through the groups the user is in
for group in user.getGroups():
  group = pg.getGroupById(group)

   # loop through the users in each of those groups
  for user in group.getGroupUsers():
    if user not in users and user != userName:
      users.append(user)

return users
```

B.6 Kapitel 11

Die folgenden Code-Listings stammen aus Kapitel 11.

B.6.1 Script (Python): scriptObjectCreation

Listing B.18 zeigt ein Script, das einen Ordner mit einem Dokument darin erzeugt.

Listing B.18: scriptObjectCreation

```
##title=Create
##parameters=
# create with a random id
newId = context.generateUniqueId('Folder')

# create a object of type Folder
context.invokeFactory(id=newId, type_name='Folder')
newFolder = getattr(context, newId)

# create a new Document type
newFolder.invokeFactory(id='index.html', type_name='Document')

# get the new page
newPage = getattr(newFolder, 'index.html')
newPage.edit('html', '<p>This is the default page.</p>')

# return something back to the calling script
return "Done"
```

B.6.2 Page Template: getCatalogResults

Listing B.19 zeigt eine Beispielkatalogabfrage, die auf den REQUEST-Parametern eine Abfrage durchführt.

Listing B.19: getCatalogResults

```
##title=Get Catalog Results
##parameters=
return context.portal_catalog.searchResults(REQUEST=context.REQUEST)
```

B.6.3 Page Template: testResults

Listing B.20 zeigt ein Beispielergebnis einer Katalogsuche.

Listing B.20: testResults

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-US"
      lang="en-US"
      metal:use-macro="here/main_template/macros/master"
      i18n:domain="plone">
<body>
<div metal:fill-slot="main">
<ul tal:define="results here/getCatalogResults">
    <li tal:repeat="result results">
        <a href=""
           tal:attributes="href result/getURL"
           tal:content="result/Title" />
        <span tal:replace="result/Description" />
    </li>
</ul>
</div>
</body>
</html>
```

B.6.4 Page Template: testForm

Listing B.21 zeigt ein Beispielformular, das die Seite `testResults` mit einer Dropdown-Liste aufruft, die auf einer Katalogabfrage basiert.

Listing B.21: testForm

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-US"
      lang="en-US"
      metal:use-macro="here/main_template/macros/master"
```

```
      i18n:domain="plone">
<body>
<div metal:fill-slot="main">
    <p>Select a content type to search for</p>
    <form method="post" action="testResults">
        <select name="Type">
            <option
   tal:repeat="value python:here.portal_catalog.uniqueValuesFor('Type')"
   tal: content="value" />
        </select>
        <br />
        <input type="submit" class="context">
    </form>
</div>
</body>
</html>
```

B.7 Kapitel 12

Die folgenden Code-Listings stammen aus Kapitel 12.

B.7.1 Beispielprodukt: PloneSilverCity

Dieses Produkt finden Sie im Kollektiv unter *http://sf.net/projects/collective*. Sie können dieses Produkt auch von der Website zu diesem Buch unter *http://plone-book.agmweb.ca* herunterladen.

B.7.2 Beispielprodukt: PloneStats

Dieses Produkt finden Sie im Kollektiv unter *http://sf.net/projects/collective*. Sie können dieses Produkt auch von der Website zu diesem Buch unter *http://plone-book.agmweb.ca* herunterladen.

B.8 Kapitel 13

Die folgenden Code-Listings stammen aus Kapitel 13.

B.8.1 Beispielprodukt: ArchExample

Dieses Produkt befindet sich im Archetypes-Release, aber Sie können eine Kopie auch von der Website zu diesem Buch unter *http://plone-book.agmweb.ca* herunterladen.

B.8.2 Page Template: email_widget.py

Listing B.22 zeigt ein eigenes Beispiel-Widget für Archetypes.

Listing B.22: email_widget.py

```
<!--
$Id: feedbackForm.cpt,v 1.2 2004/01/05 05:07:20 andy Exp $
Copyright: ClearWind Consulting Ltd
License: http://www.clearwind.ca/license
-->
<html xmlns:tal="http://xml.zope.org/namespaces/tal"
      xmlns:metal="http://xml.zope.org/namespaces/metal"
      i18n:domain="plone">

  <body>
    <div metal:define-macro="edit">
      <div metal:use-macro="here/widgets/string/macros/edit" />
    </div>

    <div metal:define-macro="search">
      <div metal:use-macro="here/widgets/string/macros/search" />
    </div>
    <div class="field" metal:define-macro="view">
        <metal:block define-slot="widget_label" />
        <metal:block use-macro="here/widgets/field/macros/view">
              <metal:block fill-slot="widget_view">
                  <a href="#" tal:attributes="href string:mailto:${accessor}"
                     tal:content="accessor">email</a>
              </metal:block>
        </metal:block>
    </div>
</body>
</html>
```

B.8.3 Beispielprodukt: WorldExample

Das Listing für das Produkt WorldExample können Sie von der Website zu diesem Buch unter *http://plone-book.agmweb.ca* herunterladen.

B.8.4 Python-Modul: PersonSQL.py

Listing B.23 zeigt einen Archetypes-Inhaltstyp für eine Person, der die Daten in einer relationalen Datenbank speichert.

Listing B.23: PersonSQL.py

```
#!/usr/bin/python
#$Id: PersonSQL.py,v 1.1 2004/01/09 21:02:37 andy Exp $
#Copyright: ClearWind Consulting Ltd
#License: http://www.clearwind.ca/license

from Products.Archetypes.public import Schema
from Products.Archetypes.public import IntegerField, StringField
from Products.Archetypes.public import IntegerWidget, StringField
from Products.Archetypes.SQLStorage import PostgreSQLStorage
from config import PROJECTNAME

schema = BaseSchema + Schema((

  IntegerField('age',
      validators=(("isInt",)),
      storage = SQLStorage(),
      widget=IntegerWidget(label="Your age"),

      ),

  StringField('email',
      validators = ('isEmail',),
      index = "TextIndex",
      storage = SQLStorage(),
      widget = StringWidget(label='Email',)
      ),

    ))

class PersonSQL(BaseContent):
    """Our person object"""
    schema = schema

registerType(PersonSQL, PROJECTNAME)
```

B.9 Kapitel 14

Die folgenden Code-Listings stammen aus Kapitel 14.

B.9.1 Python-Modul: header.py

Listing B.24 zeigt ein Script, das die Header für eine URL ausgibt.

Listing B.24: header.py

```
#!/usr/bin/python
import sys

from httplib import HTTP
from urlparse import urlparse

def getHeaders(url, method):
    p = list(urlparse(url))
    if not p[0]:
        url = 'http://' + url
        p = list(urlparse(url))

    h = HTTP(p[1])

    h.putrequest(method, p[2])
    h.putheader('Accept-Encoding', 'gzip, deflate')
    h.endheaders()

    reply = h.getreply()
    print "Status:", reply[0]
    print "Status message:", reply[1]
    hdrs = reply[2].headers
    hdrs.sort()
    for header in hdrs:
        print header[:-1]

def usage():
    print """Usage: headers.py URL [method]

URL - the URL to get headers for, http:// default
method - GET default
"""
    sys.exit()

if __name__=='__main__':
    if len(sys.argv) < 2: usage()
    method = 'GET'
    if len(sys.argv) > 2:
        method = sys.argv[2]
    getHeaders(sys.argv[1], method)
```

B.9.2 Script (Python): myCachingRules

Listing B.25 enthält eine angepasste Caching-Regel für einen Policy-Manager, nur um Ihnen mehr Optionen zu geben.

Listing B.25: myCachingRules

```
##parameters=content
# cache all files, images and anything
# thats published
if content.portal_type in ['File', 'Image']:
    return 1
if content.review_state in ['published',]:
    return 1
```

B.9.3 Externe Methode: Purge Cache

Listing B.26 zeigt ein Beispiel für ein Script, das den Cache-Speicher löscht.

Listing B.26: Purge Cache

```
import urllib
import urlparse
import httplib

URLs = [
    # enter the URLs you would like
    # to purge here
    'http://localhost:8080',
]

def purge(objectURL):
    for url in URLs:
        if not url:
            continue
        assert url[:4] == 'http', "No protocol specified"

        url = urlparse.urljoin(url, objectURL)
        parsed = urlparse.urlparse(url)
        host = parsed[1]
        path = parsed[2]

        h = httplib.HTTP(host)
        h.putrequest('PURGE', path)
        h.endheaders()
```

```
        errcode, errmsg, headers = h.getreply()
        h.getfile.read()

if __name__ == '__main__':
    print purge('/')
```

C Glossar und Werkzeuge

Dieser Anhang beschreibt alle Standardwerkzeuge und -objekte, die in einer Plone-Site erzeugt werden, und verweist auf Stellen, wo sie im Buch vorkommen. Das Glossar enthält eine Liste aller wichtigen Begriffe, die in diesem Buch und in Plone verwendet werden.

C.1 Werkzeuge

Tabelle C.1 beschreibt die Standardwerkzeuge, die Plone erzeugt.

Werkzeugname	Beschreibung
`caching_policy_manager`	Dient dem Cachen von Inhalten (siehe Kapitel 14).
`content_type_registry`	Bietet verschiedene Möglichkeiten, neue Inhalte zu verarbeiten (siehe Kapitel 11).
`plone_utils`	Allgemeine Hilfsfunktionen, auf die normalerweise nicht zugegriffen wird.
`portal_actionicons`	Verbindet ein Bild mit einer Aktion.
`portal_actions`	Die Kerndefinition der Aktionen.
`portal_calendar`	Enthält den calendar-Slot, wird sonst nicht benutzt. In Kapitel 4 finden Sie über Informationen den calendar-Slot.
`portal_catalog`	Katalogwerkzeug für Indexinhalte (siehe Kapitel 11).
`portal_controlpanel`	Bietet eine Schnittstelle zu den in Plone sichtbaren Aktionen im Control Panel.
`portal_discussion`	Dient zur Diskussion von Inhalten (siehe Kapitel 4).
`portal_factory`	Sorgt dafür, dass bei der normalen Erzeugung von Inhalten keine unfertigen Teilobjekte in der Datenbank liegen bleiben (siehe Kapitel 12).
`portal_form`	Veraltet, wegen Rückwärtskompatibilität noch vorhanden.
`portal_form_controller`	Stellt Dienste für Formulare zur Verfügung (siehe Kapitel 7).

Tabelle C.1: Plone-Werkzeuge

Werkzeugname	Beschreibung
portal_groupdata	Speichert Information über Gruppen (siehe Kapitel 9).
portal_groups	Dient zur Erzeugung von Gruppen (siehe Kapitel 9).
portal_interface	Bietet Entwicklern ein API zur Untersuchung von Objektschnittstellen.
portal_memberdata	Speichert Information über Benutzer (siehe Kapitel 9).
portal_membership	Kümmert sich um Optionen von Mitgliedschaften (siehe Kapitel 10).
portal_metadata	Metadaten über Portal-Inhaltstypen. Wird wenig benutzt.
portal_migration	Dient der Migration zu neuen Versionen von Plone (siehe Kapitel 14).
portal_navigation	Veraltet, wegen Rückwärtskompatibilität noch vorhanden.
portal_properties	Eigenschaften und Werte der Site (siehe Kapitel 4.)
portal_quickinstaller	Ein Hilfswerkzeug für die schnelle Installation von Produkten (siehe Kapitel 10).
portal_registration	Kümmert sich um Optionen bei der Registrierung von Benutzern (siehe Kapitel 9).
portal_skins	Enthält Dienste zu Skins und alle Skins selbst (siehe Kapitel 4-7).
portal_syndication	Bietet Zugriff auf RSS-Feeds zu Plone-Inhalten. (RSS steht für *Rich Site Summary* bzw. *Really Simple Syndication*.)
portal_types	Hauptwerkzeug bei der Behandlung von Inhaltstypen in einem Portal (siehe Kapitel 11-13).
portal_undo	Bietet Zugang zu den Undo-Mechanismen von Plone.
portal_url	Bietet Zugang zu nützlichen APIs bei der Bestimmung von URLs.
portal_workflow	Bietet Workflow und dazugehörige Möglichkeiten (siehe Kapitel 7).

Tabelle C.1: Plone-Werkzeuge (Forts.)

C.2 Objekte

Tabelle C.2 beschreibt die Standardobjekte, die Plone in einer Plone-Site erzeugt.

Objekt	Beschreibung
`HTTPCache`	Enthält HTTP-Header für Skins (siehe Kapitel 14).
`MailHost`	Bietet Zugang zum SMTP-Server (Simple Mail Transfer Protocol) zum Versenden von E-Mail.
`Members`	Ein umfangreicher Ordner, in dem die Ordner der Site-Mitglieder erstellt werden (siehe Kapitel 9).
`RAMCache`	Bietet Caching für Skins im RAM (Random Access Memory; siehe Kapitel 14).
`acl_users`	Der Hauptordner für Benutzer (siehe Kapitel 14).
`cookie_authentication`	Bietet Authentifikation von Benutzern mit Hilfe von Cookies (siehe Kapitel 9).
`error_log`	Die Protokolldatei von im System aufgetretenen Fehlern (siehe Kapitel 4).
`index_html`	Das Standard-`index_html`, das auf der Site erscheint (siehe Kapitel 6).

Tabelle C.2: Plone-Objekte

C.3 Glossar

Tabelle C.3 enthält Definitionen aller wichtigen Begriffe in der Plone-Welt.

Objekt	Beschreibung
Akquisition	*Akquisition* ist ein Mechanismus in Zope zur Vererbung von Objekteigenschaften. Akquisition ist die Grundlage der Zope-Objekthierarchie, die davon viel Gebrauch macht.
Aktion	In der Plone-Terminologie sind *Aktionen* eine konfigurierbare Art von Navigationselementen auf einer Site. Einige Beispiele sind Anzeigen, Bearbeiten und Mitglieder. In Kapitel 5 finden Sie weitere Details.
Anfrage (request)	Jede Anfrage eines Clients nach einer Seite erzeugt eine Anfrage an Plone. Diese wird in einem Anfrage-Objekt in Zope gekapselt, das normalerweise `REQUEST` oder `request` heißt.
Anmeldung (login)	Durch diesen Prozess gehen Sie, wenn Sie auf dem Anmeldeschirm Ihren Benutzernamen und Passwort eingeben, was gleichbedeutend mit der Authentifikation ist.
Anonyme Rolle	Eine Standardrolle im Sicherheitsmodell von Zope. Die anonyme Rolle wird Besuchern der Site so lange zugeordnet, bis sie sich mit ihrer Zope-Kennung und ihrem Zope-Passwort anmelden.

Tabelle C.3: Plone-Definitionen

Objekt	Beschreibung
Antwort (response)	Zu jeder Anfrage wird eine Antwort generiert, die in Zope in einem Antwort-Objekt gekapselt wird. Dieses heißt normalerweise `RESPONSE` oder `response`.
Anzeige (view)	Eine *Anzeige* stellt Informationen mit einer vordefinierten Struktur dar. Die Aktionen z.B. in `portal_types` sind Anzeigen.
Archetypes	Ein Framework für die Entwicklung neuer Inhaltstypen in Zope/CMF/Plone (siehe Kapitel 13).
Authentifizierter Benutzer	Ein Benutzer, der im Zope-System angemeldet ist. Wenn gerade keine Benutzer angemeldet sind, werden anonyme Benutzer als authentifizierte Benutzer betrachtet.
Authentifikation	Der von Zope verwendete Identifizierungsprozess.
Benutzerschnittstelle (UI, user interface)	Die *Benutzerschnittstelle* (auch User Interface bzw. UI) besteht aus der Art und Weise und den Bildschirmen, wie Sie und mit denen Sie mit einem Software-Programm interagieren.
Besitz (von Objekten)	Benutzer, die Objekte in Zope erzeugen, werden zu *Besitzern* dieser Objekte. Jedes Objekt in Zope hat einen Besitzer, außer vielleicht jene, die bei der Installation von Zope angelegt werden.
Besitzer-Rolle (owner)	Das ist die Standardrolle eines Besitzers in Zope.
CMF	Das *Content Management Framework* ist ein Zusatz zu Zope, der Dienste enthält, die ein Content-Management-System benötigt.
CMFTypes	Der alte Name von Archetypes.
CMS	Ein *Content-Management-System* ist ein System zur, nun ja, Verwaltung von Inhalten.
Cookie-Authentifikation	`cookie_authentication` (auch bekannt als *CookieCrumbler*) ermöglicht eine formularbasierte Anmeldung (siehe Kapitel 9).
CSS	*Cascading Style Sheets* sind ein System in HTML, mit dem Elementstile definiert werden können. Plone verwendet CSS sehr ausgiebig. Sie finden einige Beispiele in Kapitel 7.
Dienste (services)	Das Ziel von CMF ist die Vereinheitlichung der Verwaltung von Inhalten und die Bereitstellung einer Reihe von Diensten, darunter Katalogisierung, Workflow und Syndizierung. CMF und Plone bieten viele Dienste für Ihre Site. Es gibt öffentlich verfügbare Dienste wie Suche und Diskussionen, ebenso wie Verwaltungsdienste wie Workflow.

Tabelle C.3: Plone-Definitionen (Forts.)

Objekt	Beschreibung
DTML	Die *Document Template Markup Language* (DTML) ist eine serverseitige Templating-Sprache, mit der dynamisch Teile von Inhalten erzeugt werden können. Sie wird vor allem zusammen mit HTML verwendet. In Plone wird sie kaum benutzt und wird allgemein als veraltet betrachtet (siehe Page Templates).
Diskussionen	Das Werkzeug `portal_discussion` enthält die Policy, die angibt, wie Diskussionen in einem Plone-System funktionieren.
ECMAscript	Im Wesentlichen ist das JavaScript.
Eigenschaften (properties)	Im Wesentlichen sind das Attribute eines Objekts. Die Eigenschaften eines Zope-Objekts können Sie sehen, wenn Sie auf den PROPERTIES-Reiter im ZMI klicken, wenn Sie dieses Objekt gerade anzeigen. Eigenschaften werden auch in Objekten in der Plone-Schnittstelle benutzt, um mögliche Eingenschaften von Objekten zu beschreiben, z.B. Stichwörter.
Externe Methoden	*Externe Methoden* sind im Wesentlichen Python-Module im Dateisystem, die mit Zope über das Objekt *External Method* verbunden sind, das Sie aus dem Dropdown-Menü heraus erzeugen können. Externe Methoden sind mächtiger als *Script (Python)*-Objekte, weil sie nicht genauso streng unter das Sicherheitsmodell von Zope fallen, wie das bei *Script (Python)*-Objekten der Fall sein kann.
Factory	Eine *Factory* ist ein Werkzeug zur Erzeugung anderer Objekte.
Factory-Typinformation (FTI)	Die *Factory-Typinformation* enthält die Information, die im Werkzeug `portal_types` geladen wird.
Ebene(Layer)	In Plone ist eine *Skin* eine durchnummerierte Sammlung von *Ebenen* (Layers). Bei Ebenen ist nicht genau festgelegt, was sie machen können. Sie können visuelle Aspekte einer Plone-Site ändern. Sie können neue Inhaltstypen in einer mehr oder weniger präsentationsneutralen Weise einführen, oder sie können das Verhalten in anderen Skins ändern/überschreiben.
GPL	Die GPL (GNU Public License) beschreibt die Lizenzbedingungen von Plone.
Globbing (ZCatalog)	Mit dem *Globbing*-Mechanismus können Sie im ZCatalog mit Hilfe von Jokern (*) suchen. Damit kann man auch nach Teilwörtern im ZCatalog suchen.

Tabelle C.3: Plone-Definitionen (Forts.)

Objekt	Beschreibung
HTML	Die *Hypertext Markup Language* ist die grundlegende Auszeichnungssprache im Web. Dieses Buch setzt voraus, dass Sie wissen, was HTML ist.
i18n (*Internationalization*)	Unter *Internationalisierung* versteht man die Aufbereitung eines Programms derart, dass es ohne weitere Änderungen am Quelltext in mehreren Sprachen benutzt werden kann. Der Begriff *i18n* entstand als Abkürzung, indem man alle Buchstaben zwischen dem ersten und dem letzten wegließ und dazwischen die Menge der entfernten Buchstaben angab.
Inhalt (content)	Aus der Sicht des CMF ist alles *Inhalt*. Das gilt für traditionelle Dinge wie HTML-Seiten, aber auch für dynamische Informationen wie Beiträge in einer Diskussion oder Kalendereinträge. Das heißt auch, dass Bilder, herunterladbare Programme, Programmlogik in Scripten usw. ebenfalls Inhalte sind.
Inhaltstyp (content type)	Ein *Inhaltstyp* ist der in einer CMF/Plone-Instanz erlaubte Inhalt. Plone enthält vordefinierte Inhaltstypen, aber Sie können eigene Inhaltstypen für Ihre eigenen Bedürfnisse erstellen und diese in Ihrer Plone-Instanz verwenden.
Instanz	Objekte werden auch als *Instanzen* bezeichnet. Eine Instanz bzw. Objekt ist eine Instanz einer Klasse.
JavaScript	Eine in Webbrowsern eingebaute Sprache, mit der Sie Webseiten dynamischer machen können. Ein gutes Beispiel für JavaScript ist das grüne Dropdown-Menü zum Hinzufügen von Elementen.
Kalender	`portal_calendar` bietet einen Mechanismus, der verwaltet, welche Inhalte im Kalender angezeigt werden.
Katalog	Ein interner Index der Inhalte in Plone, in dem gesucht werden kann. Auf das Katalogobjekt kann man über das Objekt `portal_catalog` im ZMI zugreifen (siehe auch Kapitel 11).
Klasse	Eine *Klasse* ist die Form, aus der Objekte gestanzt werden. Objekte sind Instanzen einer Klasse. Sie können sich eine Klasse als Entwurf eines Objekts vorstellen.
Klassen-Konstruktormethode	Eine *Konstruktormethode* für eine Klasse ist eine Methode, die die Ausführung von gewissen Aktionen erlaubt, gleich nachdem eine Instanz der Klasse erzeugt wird und noch bevor diese benutzt wird. Standardattribute z.B. würde man in einer Konstruktormethode setzen.

Tabelle C.3: Plone-Definitionen (Forts.)

Objekt	Beschreibung
l10n (*Localization*)	Unter *Lokalisierung* versteht man die Vorbereitung der Daten für eine bestimmte Sprache. Plone z.B. ist i18n-fähig und hat Lokalisierungen für mehrere Sprachen. Der Begriff *l10n* entstand genau wie *i18n* als Abkürzung für *Localization*.
Lokale Rolle	*Lokale Rollen* werden an einen bestimmten Zope-Benutzer für ein bestimmtes Objekt vergeben. Eine lokale Rolle bestimmt die Rechte dieses Benutzers an diesem Objekt. Eine lokale Rolle kann dazu verwendet werden, die Rechte eines Benutzers am gegebenen Objekt einzuschränken. Sie können mit lokalen Rollen auch Benutzern, die evtl. nur beschränkte globale Rechte haben, erweiterte Rechte an einer kleinen Untermenge von Objekten geben.
Manager	Eine Standardrolle in Plone.
Metatyp	Ein eindeutiger String zu jedem Zope-Produkt im ZMI-Menü *Available Objects*. Instanzen von Produkten werden mit diesem Metatyp erzeugt. Jedes Produkt hat einen eindeutigen Metatyp.
Metadaten	Informationen über Inhalte (siehe *http://www.dublincore.org*).
METAL	Macro Expansion Template Attribute Language.
Migration	Eine *Migration* ist ein größtenteils automatisierter Vorgang, bei dem Sie Ihre Plone-Instanz auf eine höhere Version aktualisieren.
Mitgliederdaten (Memberdata)	In Plone speichert das Werkzeug `portal_memberdata` die Attribute von Benutzern.
Namespace	Ein *Namespace* enthält die Namen aller gültigen Variablen einer gegebenen Klasseninstanz (eines Objekts) in einem bestimmten Sichtbarkeitsbereich.
Nicht ordnerartige Objekte (nonfolderish objects)	Das sind Objekte, die die keine anderen Zope- oder Plone-Objekte enthalten können. Das können z.B. Dokumente oder Dateien sein.
Oberklasse (base class)	Eine *Oberklasse* ist eine Klasse, die ihre Methoden, Eigenschaften usw. an ihre Unterklassen weitergibt. Diese Unterklassen erben die Methoden und Eigenschaften ihrer Oberklasse.
Objekt	Ein *Objekt* ist eine Instanz einer Klasse.
Objektdatenbank (ODB)	Ein System zur Speicherung einer Hierarchie von Objekten. Die ZODB ist ein Beispiel einer Objektdatenbank. Solche Objektdatenbanken können Sie nicht genauso abfragen wie relationale Datenbanken.

Tabelle C.3: Plone-Definitionen (Forts.)

Objekt	Beschreibung
OOTB (out of the box)	Plone ist ein Beispiel einer OOTB-Anwendung, d.h. einer Anwendung, die man sofort nach der Installation benutzen kann.
Ordnerartiges Objekt (folderish object)	Ein *ordnerartiges* Objekt in Zope ist eines, das andere Objekte enthalten kann. Die Objekte *Folder* und *PloneFolder* sind Beispiele dafür.
Plone	»Wenn Sie es immer noch nicht wissen, ...«
Portal-Aktionen	Portal-Aktionen betreffen die ganze Site, anders als Inhaltstypen-Aktionen, die nur einen lokalen Geltungsbereich haben.
Portal-Typ	Der *Portal-Typ* ist ein eindeutiger String für alle Inhaltstypen in Plone. Jeder Inhaltstyp in Plone hat einen Portal-Typ zu seiner eindeutigen Identifikation (auch wenn mehrere auf dem gleichen Metatyp basieren können).
Portlets	*Portlets* sind die kleinen Abschnitte auf einer Plone-Site, die links und rechts auf einer Seite als kleine Kästen erscheinen.
Python	*Python* ist eine objektorientierte Scripting-Hochsprache, in der Zope geschrieben ist.
QuantumLeap	Sie werden bemerken, dass in Plone sehr lange Ergebnislisten auf mehreren Seiten angezeigt werden. Sie können eine dieser Seiten beliebig anspringen, wobei Sie in der Navigation benachbarte Seiten sehen können. Dieser Mechanismus wird liebevoll als *QuantumLeap(ing)*, d.h. QuantenSprung, bezeichnet.
Rechte (permissions)	Die Rechte bestimmen, was ein Benutzer in Zope tun darf. Rechte können nur auf Rollen angewendet werden. Sie können keine Rechte an einzelne Benutzer vergeben.
Registrierung	`portal_registration` verwaltet die site-weite Policy, die bestimmt, wie sich Benutzer im System registrieren können.
Repurposing	Inhaltstypen können auf der FTI anderer Inhaltstypen basieren, was man mit *Repurposing* (Umwidmen) bezeichnet. Dann können Sie für neue Inhaltstypen eindeutige Metadatenattribute wie *id*, *title* oder *description* angeben.
Skin	Stellen Sie sich eine *Skin* als Look-and-Feel einer Interaktion mit Plone vor. Eine Skin enthält HTML, CSS, JavaScript, Bilder und alle Interaktionen zwischen dem Benutzer und Plone. Sie können verschiedene Skins auf den gleichen Inhalt anwenden, d.h. ein Inhalt kann auf viele verschiedene Weisen angezeigt werden. Manche Skins bieten zusätzliche Features und Seiten.

Tabelle C.3: Plone-Definitionen (Forts.)

Objekt	Beschreibung
Stichwörter	Unter dem EIGENSCHAFTEN-Reiter von Inhalten können Sie *Stichwörter* zuweisen (in der Metadaten-Terminologie auch *Thema* (Topic) genannt). Mit diesem Mechanismus können Sie Inhalte zueinander in Verbindung setzen. Stichwörter können im Werkzeug `portal_metadata` vordefiniert werden.
Syndizierung (syndication)	Unter *Syndizierung* versteht man den Vorgang, mit dem eine Site Informationen mit anderen Sites austauschen kann. Die Syndizierung von Inhalten im CMF ermöglicht es, Inhalte für andere Sites bereitzustellen. Mit dem Werkzeug für diese Syndizierung können Site-Manager die site-weite Syndizierung von Inhalten verwalten. Diese Inhalte werden im RSS-Format in Ordnern verfügbar gemacht, in denen eine Syndizierung aktiviert wurde.
TAL	Die *Tag Attribute Language* ist eine Sprache zur dynamischen Auszeichnung von HTML.
TALES	Die *TAL Expression Syntax* ist eine Syntax für die Erweiterung von TAL.
Werkzeug (tool)	Ein *Werkzeug* ist eine Instanz einer Klasse in der Plone-Site. Anders als bei anderen Objekten kann es jedoch immer nur eine Instanz eines bestimmten Werkzeugs in einer Plone-Site geben. Manche dieser Werkzeuge, z.B. `portal_catalog`, bieten dem Site-Manager auch Verwaltungsoptionen.
Workflow	Eine Methode, um Geschäftslogik in einem separaten Modul zu kapseln (siehe Kapitel 8).
XML	Die *Extensible Markup Language* ist ein Standard für den Austausch von Daten.
ZMI	Das *Zope Management Interface* besteht im Allgemeinen aus der Web-Schnittstelle für die Verwaltung und Administration von Zope. (Beachten Sie am Ende der URL *http://ihre.zope.site:8080/manage* das *manage*, wenn Sie sich anmelden.)
Zope	*Zope* ist ein Open Source- Web-Application-Server, der in Python geschrieben ist. Plone setzt auf Zope auf.
ZPL	Die *Zope Public License* beschreibt die Lizenzbedingungen für den Einsatz von Zope.
ZPT	*Zope Page Templates* ist das System, das dynamische Seiten mit Hilfe von TAL erzeugt.

Tabelle C.3: Plone-Definitionen (Forts.)

Stichwortverzeichnis