The Plone Book

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Chapter 1: Introduction

What is Plone?

Plone is a free, open source Content Management System. The focus of Plone is to provide value at every level of an organization. It comes with a workflow engine, pre−configured security and roles, a set of content types and multi−lingual support. There are many developers, writers and testers from all over the world, contributing to Plone everyday. Plone is based on the Content Management Framework.

Homepage: www.plone.org

What is a Content Management System?

Finding a definition for what a Content Management System (CMS) is seems to be harder than finding someone willing to sell you one. Simply put a CMS allows you to manage content, usually for a web site. The main goals of CMS are to allow easy creation, publishing and retrieval of content to fit a business needs.

- "The trouble with content management is that its trivial or impossible."

Quote: from OSCOM, 2002

One common dividing line between different CMS's is the integration of the web and hence can two types of systems: a web based system, and non−web based system. Plone is a free, open source web based CMS

Why use a web based CMS?

Credit: *Marco Federighi*

The easiest way to understand a CMS like Plone is to compare it with a standard web site design tool, like Macromedia Dreamweaver. In both cases pages can be produced on a remote computer, and submitted for approval and publication. There are, however, four key differences:

- any user with the required permission can produce web pages from anywhere, using any standard browser, with no need for any specialist software. A CMS is easier to use than Dreamweaver and FTP, therefore very little training is needed, and many more production tasks can be allocated to unskilled staff. As a consequence, a CMS empowers more users to create and edit content on the Web. Also, less training and lower skills result in lower production and maintenance costs.
- pages are produced by typing text and uploading files into the site's pre−produced templates. This results in a more consistent corporate style. Thus, even though the number of people producing web pages for direct publication can be large, consistency of style and, more importantly, consistency in content structure is ensured.
- control of workflow in a CMS can be very finely grained, with the Webmaster's job being effectively devolved to many people working in different places without any lowering of security and, more generally, of quality standards.
- different versions of a document are automatically saved, resulting in a natural audit trail when required.

These benefits of Content Management Systems are obviously more significant for large organisations, or large collaborative projects, than for small businesses or organizations.

What is the Content Managment Framework?

The Content Managment Framework (CMF) is an application that contains a series of tools for Zope. These tools form a framework that provide many of the key services a Content Management System would need. The CMF can be used as a standalone product, or in Plone's case, built on top of. The CMF provides core tools like Workflow, Personalisation and Cataloguing. The CMF development is lead by Zope Corporation and is an open source product that benefits from the input and hard work of many developers around the world.

Homepage: cmf.zope.org

What is Zope?

Zope is a open source web application server, written in Python. It is a scalable, stable, powerful system that includes an object database, a web server and several templating languages. Zope is developed and supported primarily by Zope Corporation, but also by many developers worldwide.

Homepage: www.zope.org

Why use Zope and Plone?

Credit: *Marco Federighi*

Zope and Plone are Open Source Software (OSS), that is, the source code is available to anyone for free. The business model of the people who produce Zope and Plone relies on earnings from consultancy services, chiefly for customisation and enterprise use. Other examples of open source CMSes are Midgard, Bitflux, OpenCMS, and Wyona.

Proprietary and open source CMSes are technically not very different. In both camps we find very good, mediocre and poor products; the quality of the documentation and support also varies widely. The main difference is that open source CMS are produced by rather smaller companies than proprietary ones. This raises doubts on the long−term continuity of these firms, and of the support that they can provide. In my view, however, the difference is more apparent than real. Open source producers are smaller and thus more vulnerable to, say, the loss of one customer or the departure of a key individual; proprietary producers are bigger, but are affected by takeovers (e.g. Allaire, by Macromedia) and the vagaries of the IT stock market (e.g. Broadvision). All in all, BOTH kinds of producers can easily disappear. The difference is that, with an open source product, the source code is available to the user and so is the possibility of maintenance, customisation and development, none of which is available to users of proprietary systems without the active intervention of the producers. This is the key reason to use an open source product.

Why Plone and Zope rather than other open source CMS? Plone is based on Zope, which is a framework for building content management software. In a sense, Zope is an operating system for web applications, one of which is CMF (Content Management Framework), an application to facilitate the building of CMSes. Plone is one such CMS, based on CMF, running on Zope, but with its own set of templates and file types. In our view ("our" meaning the view of the CMS Working Party set up by the Web and Internet Steering Group, WISG), Zope is significantly better than other competing products for the following reasons:

- Zope is object−oriented, in the sense that everything appearing in a Zope web site (web pages, images, links, files) is an object and is contained in an object database. The database is hierarchical, not relational, and is particularly suited to hierarchical file structures. Technical people think of databases in terms of collections of tables of rows and columns, related by primary keys. This is different, and mirrors much more closely the structure of an ordinary file system, with objects within objects.
- The Zope database contains all older versions of an object: this is particularly useful for undoing changes, for the control of versions of collaborative documents, and for items requiring an audit trail.
- Zope contains a number of tools that are specially suited to large organisations and collaborative work, and will be described later in the manual. One example: pre−defined database searches, based on flexible search criteria, which automatically display all objects satisfying certain user−specified conditions.
- Zope can be used an all platforms: Unix, Linux, Mac OS, and all flavours of Windows (98, 2000, XP, NT). This is not true of most other CMS, open source and proprietary alike.
- Zope is a very friendly developing environment. The possibility of creating a customisable copy of a script at the touch of a button, while keeping the default version in its original location, is the best safety net I have ever come across.
- In Zope it is easy to design structured XML documents, with workflow linked to the document structure. This is an essential feature for the administrative systems in a large organization, and promotes both corporate consistency in the style and structure of documents and a streamlined work flow.
- Finally, Zope was created for use by large organisations, with the following characteristics: large number of contributors to collaborative projects, with contributors located at different sites and using different platforms; strong organisational requirements for flexibility and security, with the need to define local roles with different permissions to view, write, edit and approve different parts of large projects; scalability to large numbers of objects and servers.

The seventh, *cultural* difference between the Zope team and their competitors is crucial for large organisations. One of the Zope customers is the US Navy, which uses Zope for the management of RDprojects: a big, public−sector organisation with a keen eye on flexibility and security. The same could be said for most large corporations, public as well as private.

As far as Plone (as distinct from Zope) is concerned, I regard it as rather more than a generic CMS that happens to be based on Zope. Plone adds to Zope at least two very useful features, which are especially important for the purposes of the Engineering Sciences web site:

- A neat, elegant framework for navigation, relying on folders and the viewing of their content rather than links in html documents (which would have to be updated) and aided by navigational shortcuts such as the Bulletin (which displays objects created or modified within the last few days, thus eliminating the need to navigate the site to find them).
- A simple tool for the creation of complex structured documents such as PIQ and UPC forms, with different parts of each documents visible to different audiences and a customizable approval path.

The first feature makes a Plone−based site uniquely easy and fast to use; the second makes it useful for administration, unlike most CMS which are conceived primarily for the publishing rather than the processing of content.

What is Python?

Python is a powerful, interpreted, interactive, object−oriented programming language. Python is open source and can run on almost any platform or system. Zope is written primarily in Python, with some optimisations in C.

- ***python***, (Gr. Myth. An enormous serpent that lurked in the cave of Mount Parnassus and was slain by Apollo) 1. any of a genus of large, non−poisonous snakes of Asia, Africa and Australia that suffocate their prey to death. 2. popularly, any large snake that crushes its prey. 3. totally awesome, bitchin' language that will someday crush the $'s out of certain other so−called VHLL's ;−) *

Quote: "What is Python" from python.org

Homepage: www.python.org

Who is this book aimed at?

This book is aimed at people in several different roles including:

- the system administrator
- the site content manager (who oversees adding and publishing content)
- the site member (the person who joins and may be adding content)
- the site developer (people who person who writes HTML, or code)

What version of Plone does this cover?

This book covers the latest release of Plone as downloadable at Plone.org. It does not cover the CVS version or earlier versions. It also covers a "default site" as defined by the customisation policy. Whilst other versions such as the "private site" may exist, this is not covered here, however most of the concepts are the same.

Feedback

If you have feedback we'd love to hear it. The best thing you could do is send us an email to the documentation mailing list and someone will reply.

Page Editor: *Andy McKay* $Id: 1,v 1.4 2003/06/23 05:11:19 zopezen Exp $

Contents | Next [Chapter 2: Installing and Upgrading]

Previous [Chapter 1: What is Plone] | Contents | Next [Chapter 3: Using Plone]

Chapter 2: Installing Plone

This chapter covers installing Plone so that a user can get Plone up and running. ***Upgrading Plone*** mandatory steps at the bottom of the chapter!

Requirements

Plone will install on any of the platforms that Zope supports: Windows, Mac OSX, Linux, most Unixes and Solaris. Windows 2000 requires writing to the registry in order to install the software, this may require more rights than you have.

Server

A higher performance computer will obviously make Plone perform better. Plone is not a toy. It requires horsepower and memory. In general, you shouldn't go into production with a machine slower than 1.5Ghz and less than 1GB of RAM if you are serving a large website. It works fine with setups as low as 500MHz and 64MB of memory for more modest sites, however. For advanced information on performance see Chapter 9, Optimizing Plone

For a base installation of Plone about 50Mb of hard drive space is required. If you already have installations of Zope or Python, then this can be dramatically reduced. You must also account for the Plone object database which can grow to almost any size depending upon the amount of data you store.

Client

Plone only requires a web browser that can access the server. If users want to login, cookies must be enabled. JavaScript is not required but will provide a richer user experience.

Required browsers for Plone 1.0:

- Internet Explorer 5.5 and up (6.0 and up recommended)
- Netscape 7.0 and up
- Mozilla 1.0 and up (1.4 and up recommended)
- Opera 7.0 and up (7.20 and up recommended)
- Konqueror 3.0 and up (There are some inconsistencies, but you will have to live with those until they are updated against the KHTML code base changes made in Safari)
- Safari 1.1 and up

Plone also is fully functional in the following browsers, but might look different from the original Plone look:

- Netscape 4.7 and up
- Internet Explorer 5.0
- Internet Explorer 4.0 (not extensively tested, but should work well)
- Konqueror 2.x
- Lynx (text based)
- w3m (text based)

* AWeb
* links (text based, with optional graphics)
* Any browser that handles a basic set of HTML and form input + cookies (including most mobile/PDA browsers)

Downloading Plone

The latest Plone is always available at http://www.plone.org/download. In the future CD's might be available.

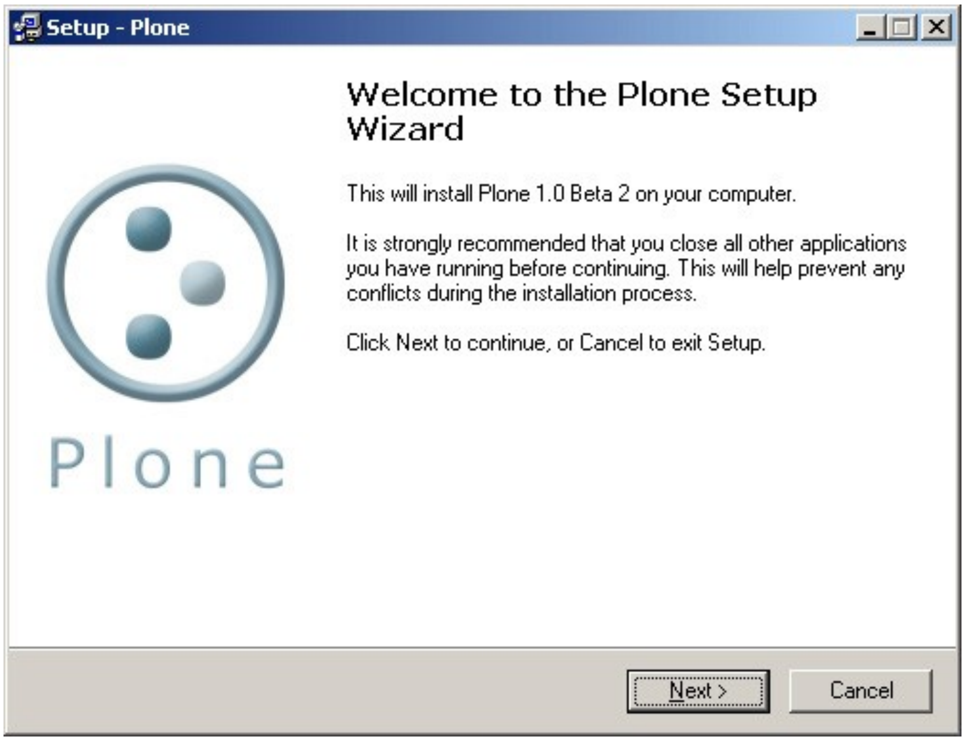
Installing using the Windows Installer

Credits: *Steve Rauch* helped with screenshots and documentation.

The Installer

The Windows installer automates the installation of Plone on Windows. Windows versions 9x, ME, NT 3.51+, 2000 and XP have been tested, but it may work on others. It is recommended you have administrator access on the computer you wish to install on. If you already have Zope or Python installed, you may want to look at installing the source seperately to save hard drive space. The installation includes extra packages and options, a pre−loaded database and other goodies.

The Plone installer for Windows can be downloaded from the Plone.org website, in the downloads section. Once you have downloaded the installer, double click on the installer to begin the install. You should be presented with the following screen:



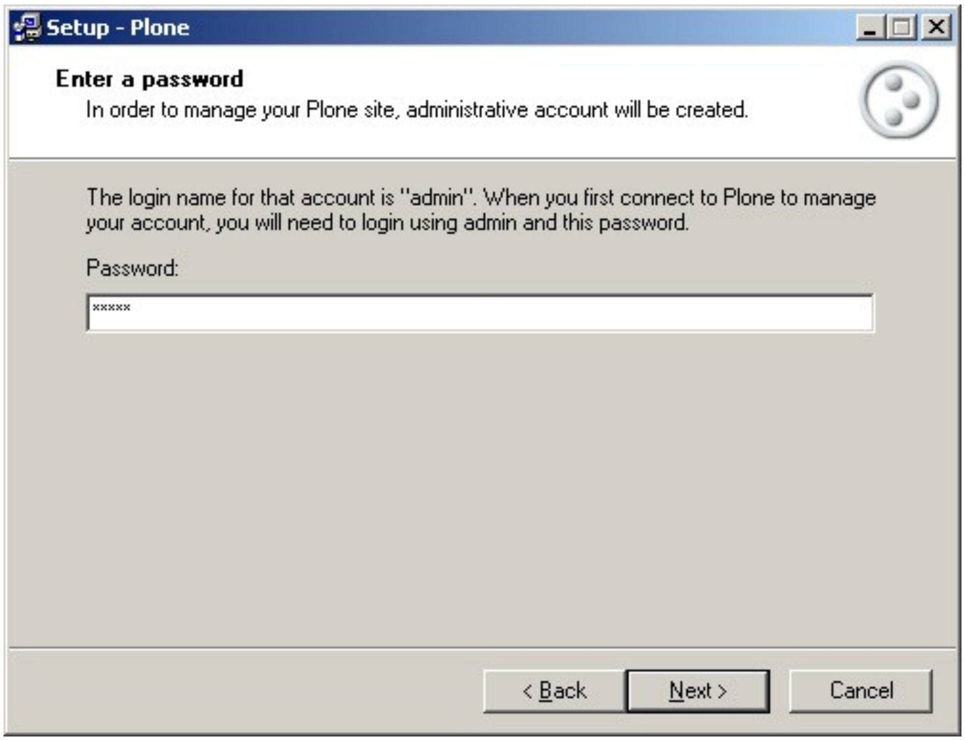
Welcome to the Plone Setup Wizard

This will install Plone 1.0 Beta 2 on your computer.

It is strongly recommended that you close all other applications you have running before continuing. This will help prevent any conflicts during the installation process.

Click Next to continue, or Cancel to exit Setup.

The installer goes through the usual steps for installing software, follow the options at the bottom for "Next" or "Cancel". There is no need to discuss all the steps, most are self explanatory. When you get to the "Enter a password" screen (shown below), you must enter a password. This will register a password for the "admin" user. You will need this password later, so make note of it. If you do lose this password you can enter a new one through the Plone Controller.


Setup - Plone

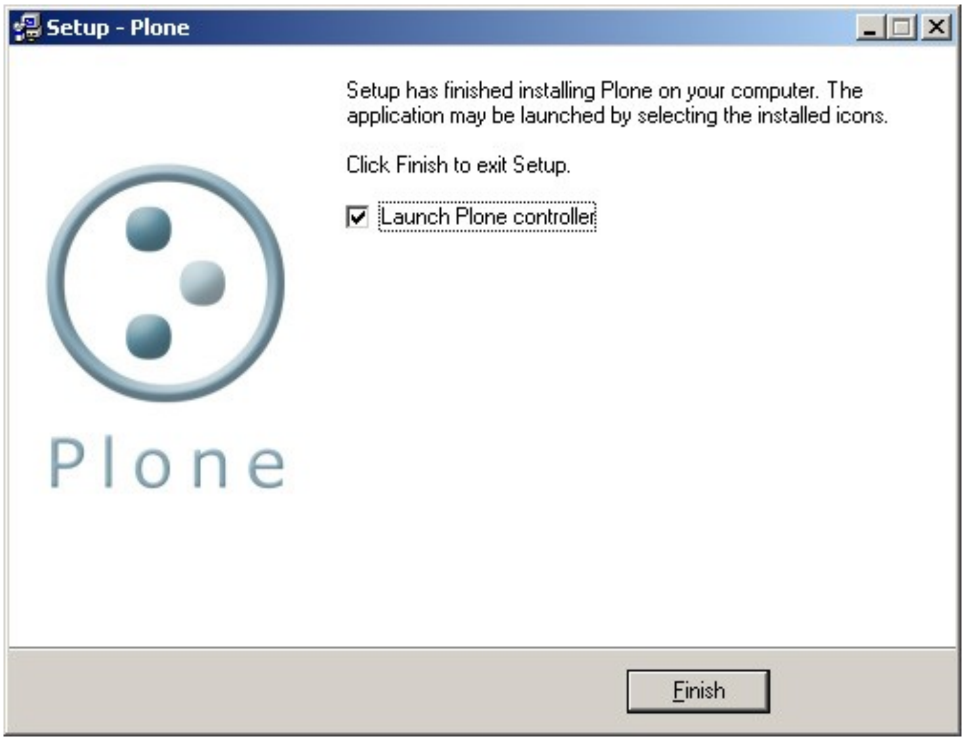
Enter a password

In order to manage your Plone site, administrative account will be created.

The login name for that account is "admin". When you first connect to Plone to manage your account, you will need to login using admin and this password.

Password: *****

The installation takes somewhere around 3 minutes, depending upon the speed of your computer. A few tasks are performed at the end of the installation, such as compiling all the Python files. When the installation has finished, Plone is not started by default. If you leave the "Launch Plone Controller" box checked the Plone Controller will be launched which will enable you to start Plone.


Setup - Plone

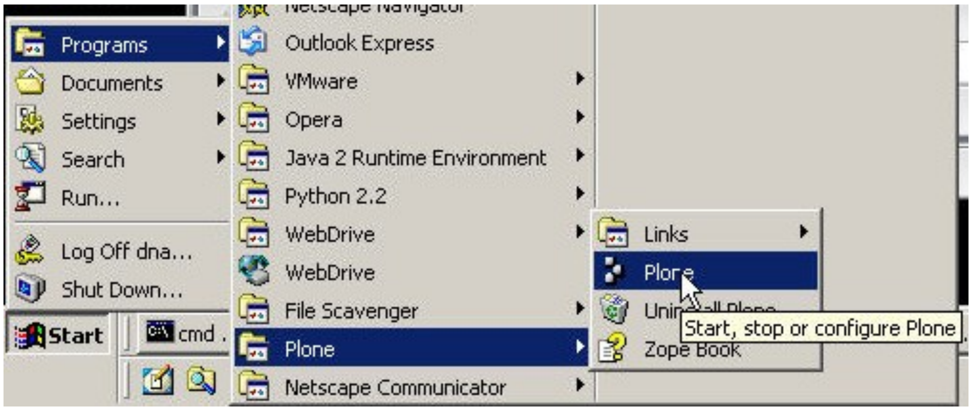
Setup has finished installing Plone on your computer. The application may be launched by selecting the installed icons.

Click Finish to exit Setup.

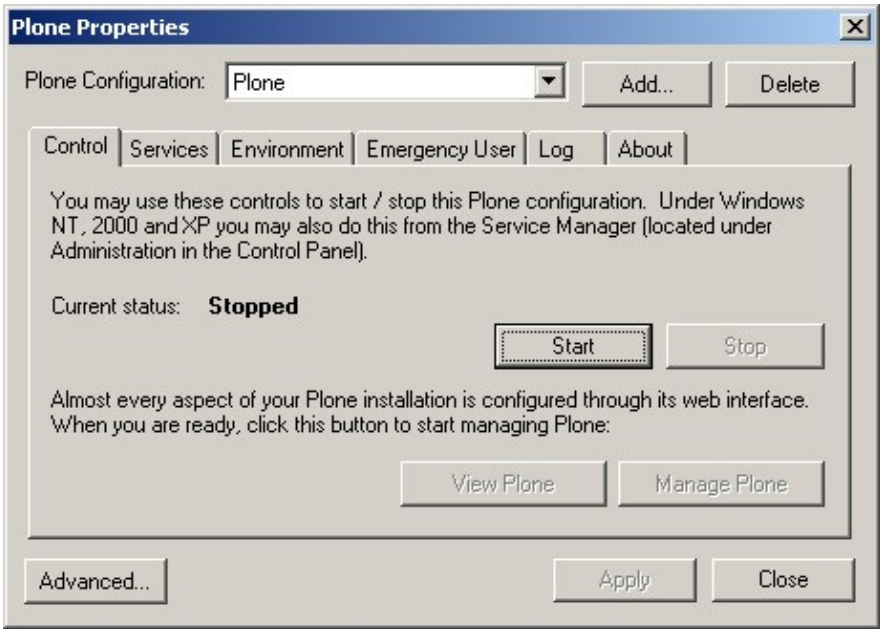
☑ Launch Plone controller

The Plone Controller

The Plone Controller allows you to easily administer the Plone instance from a GUI and gives you control over things like log files, ports and how Plone is started. To start the Plone Controller, select "Plone" from the Start menu.



The controller starts with the "Control" page which lets you easily start or stop your Plone instance. For the impatient, click "Start" to start your Plone.


Plone Properties

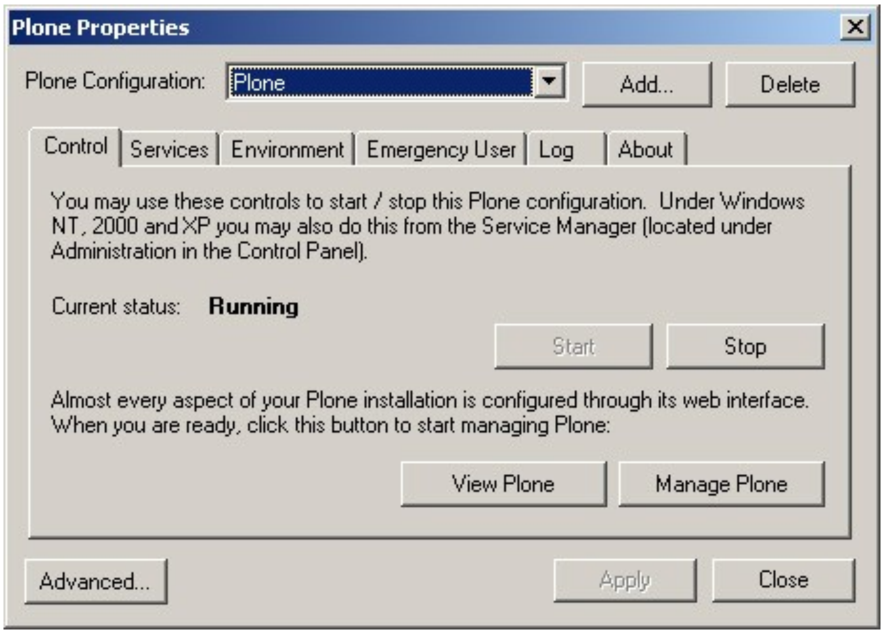
Plone Configuration: Plone

You may use these controls to start / stop this Plone configuration. Under Windows NT, 2000 and XP you may also do this from the Service Manager (located under Administration in the Control Panel).

Current status: **Stopped**

Almost every aspect of your Plone installation is configured through its web interface. When you are ready, click this button to start managing Plone:

This screen shows you if your Plone is started or stopped, by the buttons that are highlighted and the "Current Status" message. Since Plone does not start automatically you will have to click "Start" to start Plone. On some slower computers it may take a while for Plone to start up, especially the first time, this causes the Controller to time out and think Plone has not started. Clicking "Start" again will rectify this. This bug will be fixed in later versions.


Plone Properties

Plone Configuration: Plone

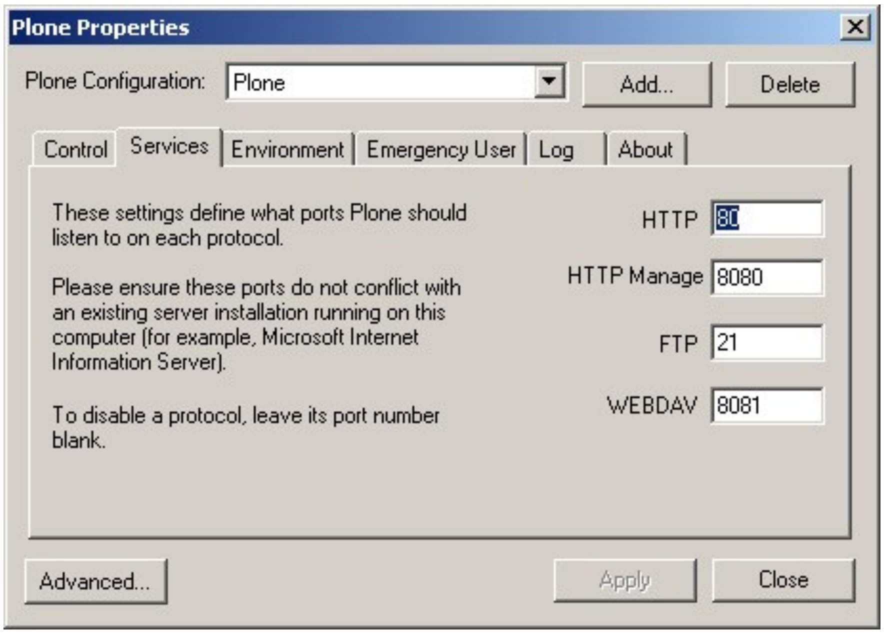
You may use these controls to start / stop this Plone configuration. Under Windows NT, 2000 and XP you may also do this from the Service Manager (located under Administration in the Control Panel).

Current status: **Running**

Almost every aspect of your Plone installation is configured through its web interface. When you are ready, click this button to start managing Plone:

When Plone has started you can access the Plone site by clicking on the "View Plone" button. This will start up a browser and access the Plone site for you. The "Manage Plone" button takes you to the management interface of your Zope if you have defined a "HTTP Manage" port, by default this port is set to 8080 for you.

Services



These settings define what ports Plone should listen to on each protocol.

Please ensure these ports do not conflict with an existing server installation running on this computer (for example, Microsoft Internet Information Server).

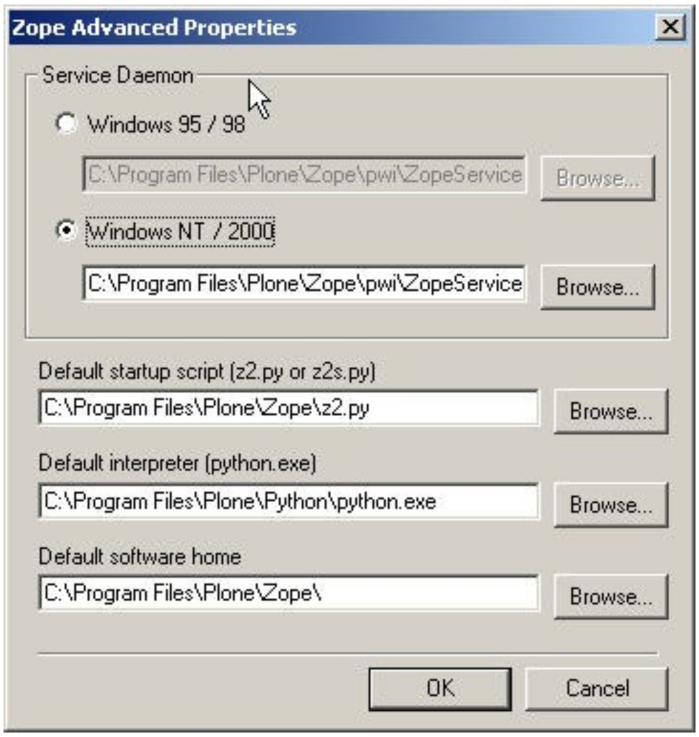
To disable a protocol, leave its port number blank.

HTTP: 80
HTTP Manage: 8080
FTP: 21
WEBDAV: 8081

The "Services" tab allows you to specify the ports that Plone listens to for incoming connections such as HTTP, FTP and WebDAV. If you leave a port blank then that port will not be disabled. Ensure that no other server is listening to the same port as Plone, things such as IIS, Apache, PWS could be listening to similar ports.

- HTTP: specifies the port to access Plone for the user, the default is port 80. Although this port is not required, without it you will not be able to access Plone with a web browser. If this port is enabled and Plone is running, the "View Plone" button is enabled on the Control tab.
- HTTP Manage: specifies the port to access Plone as the manager, the default is port 8080. This port gives you access to Zope's management interface for the root of Zope. If this port is enabled and Plone is running, the "Manage Plone" button is enabled on the Control tab.
- FTP: specifies the port to access Plone via FTP, the default is port 21.
- WebDAV: specifies the port to access Plone via WebDAV, the default is port 8081.

Advanced



Service Daemon: Windows NT / 2000 (selected)

Windows 95 / 98: C:\Program Files\Plone\Zope\pwi\ZopeService

Windows NT / 2000: C:\Program Files\Plone\Zope\pwi\ZopeService

Default startup script (z2.py or z2s.py): C:\Program Files\Plone\Zope\z2.py

Default interpreter (python.exe): C:\Program Files\Plone\Python\python.exe

Default software home: C:\Program Files\Plone\Zope\

The "Advanced" windows lets you specify some of the more advanced options. If you are unsure what these items refer to, please leave the default values alone. Of most interest is the "Service Daemon" option. This lets you specify how Plone will be run and mostly relates to the use of services in Windows.

If you are running Windows NT, 2000 or XP you can run Plone as a service (the "Windows NT/2000" option). This will run Plone in the background, no icon will be visible and Plone can be accessed as a service. As a service the "Restart" button will appear in the ZMI so an authorized user can choose to restart Plone through the web. On other advantage of running as a service is that can be started and stopped from the Management Console (inside the Windows Control Panel) or through the command line using the `net start Plone` and `net stop Plone` commands.

If you are running Windows 95, 98 or ME you must run Plone using the "Windows 95/98" option. This runs Plone as a process with an icon that shows up in the system stray. You can still control Plone from the Controller or from the system tray. Windows NT, 2000 or XP can use this option if they want.

These options are selected for you automatically during the installation.

Running Zope Manually

If there are problems running Plone with the Controller, you can also start Plone from the command line. Further details on this are available in the Zope documentation. The command that the Plone Controller calls is as follows:

```
c:\Program Files\Plone\Python\python.exe
    c:\Program Files\Plone\Zope\z2.py
    -X -w80 -w8080 -f21
    INSTANCE_HOME=c:\Program Files\Plone\Data
    HTTP_MANAGE=8080
```

- Line breaks added for easy reading, this command is all one line
- Assuming that your Plone installation is at c:\Program Files\Plone

Windows Installer Contents

In addition to Plone, the components installed in the Windows installation are:

- CMF 1.3.1
    - The CMF is required by Plone. The latest version can be found at cmf.zope.org
- Zope 2.6.1
    - This is the latest version of Zope and can be found Zope can be found at zope.org
- Python 2.1.3
    - Python 2.1.3 is required with Zope 2.6.0. Python can be found at www.python.org
- Python Win32 Extensions
    - The Win32 extensions by Mark Hammond provide access to Windows API's and functions. Win32 extensions can be found at the Starship
- PIL 1.1.3
    - The Python Imaging Library allows you to manipulate and alter images from Python. PIL can be found at pythonware.com − PIL is used by CMFPhoto to rescale images.
- ReportLab 1.15
    - ReportLab provides and interface to PDF's. It allows to create PDF's from Python. ReportLab can be found at reportlab.com
- Zope Controller 1.0
    - The Zope Controller is a Windows GUI application to provide easy access to start, stop and configure Zope. The Plone version is deriative of the Zope Controller
- CMF Collector 0.9b
    - The CMF Collector is a bug tracking system for Plone. CMF Collector is developed by Zope Corp. and can be found at cvs.zope.org
- CMF Wiki 0.1
    - The CMF Wiki 0.1 is an implementation of the Wiki system for Plone. It is developed by Zope Corp. and can be found at cvs.zope.org
- CMF Quick Installer
    - The CMF Quick Installer allows you to quickly install new Zope products easily by selecting the appropiate products in the Zope Management Interface. The Installler can be found at the SourceForge Collective project"
- CMF Photo
    - CMF Photo is a product that allows you to view photographs through Plone and allow them to be dynamically resized.CMF Photo can be found at the SourceForge Collective project"
- CMF Forum
    - CMF Forum allows you to make bulletin boards for user to add and edit comments. CMF Forum can be found at the SourceForge Collective project"
- External Editor 0.6 (Client and Server app)
    - External Editor is a program that allows you to edit Plone objects and content locally. It can be found at "zope.org": http://www.zope.org/Members/Caseman/ExternalEditor
- Zope Book 2.5
    - The original source of the Zope Book is on Zope.org and the HTML Help version is here
- Localizer
    - Localizer allows to you translate Plone into different languages
- Translation Service and Translations

- Translation Service allows translations in Zope Page Templates and Plone. This provides the integration between Plone and Localizer. The Translation Service can be found here
- There are several translations for Plone maintained at the SourceForge Plone i18n project − the installer installs all the translation files it can, although some of the translations may not be complete.

A database with an instance of Plone is installed, unless a database already exists in that directory (if so, the database is not installed). This database contains an instance of Plone (with CMF Collector and CMF Wiki installed), an Access Rule, a Site Root and External Editor installed.

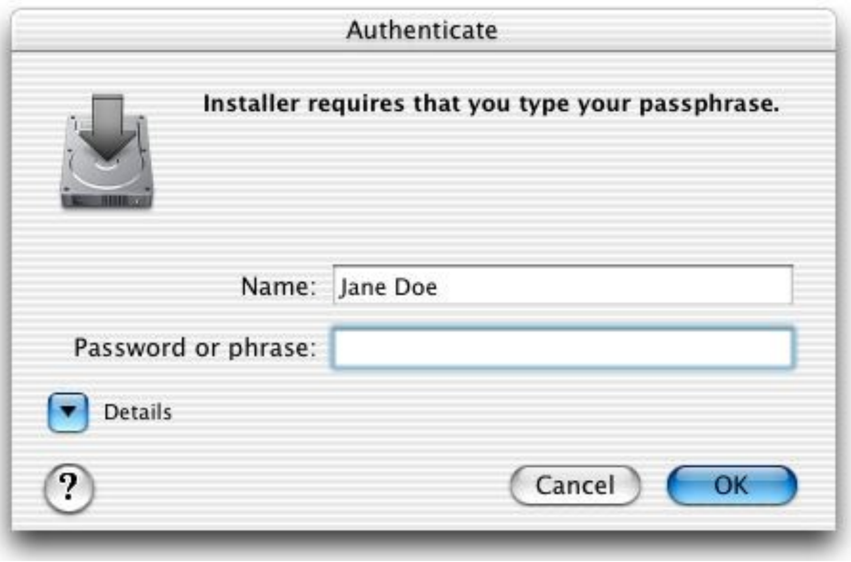
Installing using the Mac OSX Installer

Credits: *Jim Roepcke* helped with screenshots and documentation.

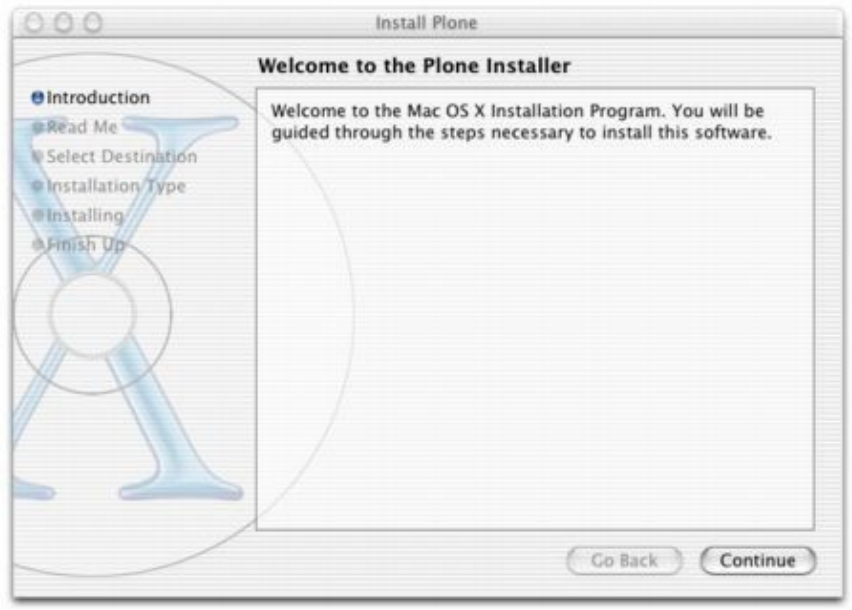
The Installer

The Mac OS X installer automates the installation of Plone on Mac OS X. Mac OS X 10.2.3 and up have been tested and are supported. It is recommended you have administrator access on the computer you wish to install on. If you already have Zope or Python installed, you may want to look at installing the source seperately to save hard drive space. The installation includes extra packages and options, a pre−loaded database and other goodies.

The Plone installer for Mac OS X can be downloaded from the Plone.org website, in the downloads section. Once you have downloaded the installer, double click on the installer to decompress the archive, and double click the resulting installer package to begin the install. You should be presented with the following screen:



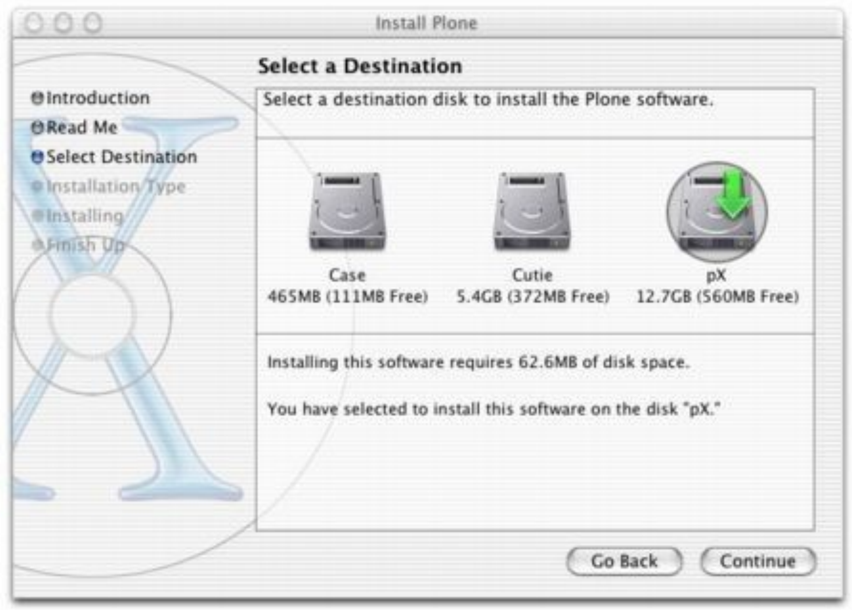
Enter your Mac OS X account password to authorize the installation. Your account must have Administrator privileges to do this. If your account does not have Administrator privileges, log out and log back in as someone who does, and then re−launch the installer. (You might want to move the installer package to /Users/Shared before you log out so you can access it from the other account) Once the installation is authorized, you will see the following screen:



Welcome to the Plone Installer

Welcome to the Mac OS X Installation Program. You will be guided through the steps necessary to install this software.

The installer goes through the usual steps for installing software, follow the options at the bottom for "Continue" or "Go Back". There is no need to discuss all the steps, most are self explanatory. However, when presented with the choice of volumes to install Plone on, you ***must*** choose the partition on which Mac OS X is installed.



Select a Destination

Select a destination disk to install the Plone software.

Case	Cutie	pX
465MB (111MB Free)	5.4GB (372MB Free)	12.7GB (560MB Free)

Installing this software requires 62.6MB of disk space.

You have selected to install this software on the disk "pX."

The installation takes somewhere around 3 minutes, depending upon the speed of your computer. When the installation has finished, Plone is not started by default. The ReadMe.rtf file in /Applications/Plone contains a lot of useful information about running and managing your Plone installation, including how to start Plone. For your convenience, the command is presented below:

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

Mac OS X Installer Contents

In addition to Plone, the components installed in the Windows installation are:

- CMF 1.3.1
    - The CMF is required by Plone. The latest version can be found at cmf.zope.org
- Zope 2.6.1
    - This is the latest version of Zope and can be found Zope can be found at zope.org
- Python 2.1.3
    - Python 2.1.3 is required with Zope 2.6.0. Python can be found at www.python.org
- PIL 1.1.3
    - The Python Imaging Library allows you to manipulate and alter images from Python. PIL can be found at pythonware.com − PIL is used by CMFPhoto to rescale images.
- ReportLab 1.15
    - ReportLab provides and interface to PDF's. It allows to create PDF's from Python. ReportLab can be found at reportlab.com
- CMF Collector 0.9b
    - The CMF Collector is a bug tracking system for Plone. CMF Collector is developed by Zope Corp. and can be found at cvs.zope.org
- CMF Wiki 0.1
    - The CMF Wiki 0.1 is an implementation of the Wiki system for Plone. It is developed by Zope Corp. and can be found at cvs.zope.org
- CMF Quick Installer
    - The CMF Quick Installer allows you to quickly install new Zope products easily by selecting the appropiate products in the Zope Management Interface. The Installler can be found at the SourceForge Collective project"
- CMF Photo
    - CMF Photo is a product that allows you to view photographs through Plone and allow them to be dynamically resized.CMF Photo can be found at the SourceForge Collective project"
- CMF Forum
    - CMF Forum allows you to make bulletin boards for user to add and edit comments. CMF Forum can be found at the SourceForge Collective project"
- External Editor 0.6 (Client and Server app)
    - External Editor is a program that allows you to edit Plone objects and content locally. It can be found at "zope.org": http://www.zope.org/Members/Caseman/ExternalEditor
- Zope Book 2.5
    - The original source of the Zope Book is on Zope.org and the HTML Help version is here
- Localizer
    - Localizer allows to you translate Plone into different languages
- Translation Service and Translations
    - Translation Service allows translations in Zope Page Templates and Plone. This provides the integration between Plone and Localizer. The Translation Service can be found here
    - There are several translations for Plone maintained at the SourceForge Plone i18n project − the installer installs all the translation files it can, although some of the translations may not be complete.

A database with an instance of Plone is installed, unless a database already exists in that directory (if so, the database is not installed). This database contains an instance of Plone (with CMF Collector and CMF Wiki installed), an Access Rule, a Site Root and External Editor installed.

Installing on Linux

Debian

Plone is a standard package in Debian sarge and sid (aka testing and unstable). Thus, Debian users can install Plone simply using the command `apt−get install plone`.

Red Hat/Mandrake/Suse

RPMs are available for the Red Hat, Mandrake and Suse distributions. The latest packages can be downloaded from SourceForge

Be aware to the dependencies of the packages. If you use them with another version of distribution it may cause problems.

Installing from source

If you are familiar installing the products from the source distribution this might be you desired method of installing Plone. However installation assumes you are familar with tools such as tar, if you are not one of the other installation options might be prefered.

Requirements

- Zope 2.6.2 or above available from zope.org
- CMF 1.3.2 or above available from cmf.zope.org

Getting the latest Plone

Its available as tar ball from plone.org. To extract the tar ball, run the following commands[1]:

```
gunzip CMFPlone1.0.tar.gz
tar −xvf CMFPlone1.0.tar
```

The current Plone tar ball contains several products apart from Plone: DCWorkflow and Formulator. You can get it from Sourceforge CVS along with these other products by running the following commands [2]:

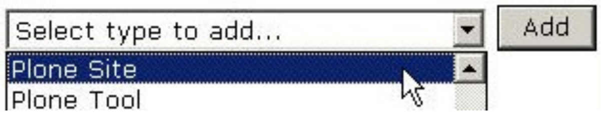
```
cvs −d:pserver:anonymous@cvs.plone.sourceforge.net:/cvsroot/plone login
cvs −d:pserver:anonymous@cvs.plone.sourceforge.net:/cvsroot/plone co −r Plone−1_0
cvs −d:pserver:anonymous@cvs.formulator.sourceforge.net:/cvsroot/formulator co Fo

cvs −d:pserver:anonymous@cvs.zope.org:/cvs−repository login
cvs −d:pserver:anonymous@cvs.zope.org:/cvs−repository checkout Products/DCWorkflo
```

Installation

- Install Zope, follow the instructions contained in `INSTALL.txt` to complete this.
- Install CMF, follow the instructions contained in `INSTALL.txt` to complete this.
- Copy the CMFPlone, ActivePak, DCWorkflow and Formulator directories into the Products directory of your Plone instance. Information can also be found in the `INSTALL.txt` file inside the CMFPlone directory.
- Restart Zope. How you do this depends upon how you installed Zope and your operating system.
- Go to the management interface, this is done by appending /manage to the URL of your Zope. You should now find "Plone Site" in the list of products to add. This means Plone has installed succesfully, see "Adding a Plone Site for more information".

Adding a Plone Site

This assumes you have successfully installed Plone using the one of the procedures described above. If you have used the Windows or Mac OSX installer then you will already have a Plone site created for you called "Plone" and there is no need to read this section unless you want to install another Plone site inside your Zope.

- Access the managment interface of your Zope, this is done by appending /manage to the URL of your Zope.
- Select "Plone Site" from the drop down list: 

- The following form prompts you for some information about your Plone site:
    - Id *required*: is the id of the Plone site and is a short name containing a restricted character set (just alphanumeric characters is recommended), this will be URL to your Plone site.
    - Title: is the name of the Plone site that will appear on all pages
    - Membership source: if you are unsure what this means, leave it at the default "Create a new...". You either have choice of adding an user folder inside your Plone site or using one in a higher folder.
    - Description: a description of the site
    - Site type: lets you define a different type of site, for example a custom site of skin. This manual assumes you have used the `Default` value.
- After "Add Plone Site" has been clicked, a new site will be created. This might take a few seconds as it loads up all the information.

Congratulations, you now have installed Plone.

Upgrading Plone on an existing installation

Note: these installation instructions should apply to all versions and future versions of Plone. However there might be extra steps needed, please consult the donwload instructions for your release.

If you have an existing Plone installation and wish to upgrade, then a migration tool has been added. This migration tool requires knowledge of, and access to the Zope Management Interface. ***Before performing a migration do a complete backup of your Plone site, just in case.***

Installing the software

- Stop Plone
- Back up your Plone installation, including the database. See chapter 9 for backing up a your database.
- Download and install the new version of Plone, how you do this depends exactly on the operating system and how you installed plone. This install will result in the files on your filesystem being updated to the new version. Usually installing directly over the old version is enough.
- Restart Plone
- For each Plone instance in your Zope, find the tool called `portal_migration` in the ZMI. It will tell you the current version of your Plone and the version on your file system. To migrate to the existing Plone just click on migrate. Plone will perform all the necessary upgrades needed.

Migrating from a pre 1.0 Alpha site

This is not covered in the migration procedure. We suggest adding a new plone instance using the new code and moving the content objects into the new site. Just use copy and paste in the Zope Management screens.

Migrating from a pre 1.0 Beta3 (but later than alpha)

In the ZMI you will need to add the migration tool. Select "Plone Tool" from the add object drop down box, in the following page select "Plone Migration Tool" and then click "Add". A new object has been added to you Plone instance called `portal_migration`. Select the object, and it will display some information about your Plone instance and the code on your file system.

However since this tool was only recently added it will not be able to tell what Plone instance you have and will display beta 1.3 for both the current instance and the file system instance. Select what you think the correct version should be and select force migration.

[1] Or whichever version of Plone you have downloaded.

[2] After using a cvs login command you will prompted for a password, just press enter (ie leave the password blank)

Page Editor: Andy McKay $Id: 2,v 1.11 2003/09/19 22:43:28 limi Exp $

Previous [Chapter 1: What is Plone] | Contents | Next [Chapter 3: Using Plone]

Previous [Chapter 2: Installing and Upgrading] | Contents | Next [Chapter 4: Workflow]

Chapter 3: Using Plone

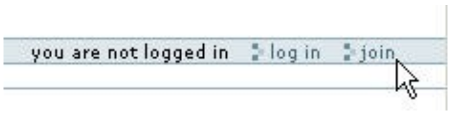
This chapter assumes you have a version of Plone installed or accessible. If you do not, Chapter 2 covers installing Plone.

Plone makes use of current browser standards and techniques, so using a recent version of a web browser is recommended. These instructions cover the default setup for Plone and may not be accurate for the site you are viewing, depending upon how it has been customized.

Joining a Site

Joining a Plone site gives you the right as a Member to add content, such as images, documents and so forth. Your exact rights depend upon how the site is configured.

To join a site, select the `join` link in the top right hand corner of the home page.



This opens the join form for you to complete. If this is the first Plone form you have encountered, you will note a few things: A red square next to the label of an input field means the field is required. When you select a field, help text will pop up to the left. You may navigate through the form fields by clicking on them.

There are several fields on the join form:

- ***Full Name***: Your name.
- ***Username***: The username you wish to use, most people choose an alphanumeric value without spaces such as "bob" or "jane97".
- ***E−mail***: A valid email address is required. That way if you lose your password it can be emailed to you. You will be able to change this email address later on by editing your member preferences.
- ***Password*** and ***Confirm Password***: The password you wish to use; it must be more than 5 characters in length and can contain letters, numbers and the underscore(_) character. Passwords are case sensitive.
- ***Mail Password?***: Check this box if you would like your password sent to the email address you provided.

Once you have completed this form, click `register` to submit your information. Then, click `log in` to access the site immediately.

Logging In

If you already have a username and password, you can log in to the site by typing them into the log in box in the lefthand column and clicking the `log in` link. Cookies must be enabled for you to log in to a Plone site.

Forgotten password

If you forget your password it can be sent to the email address you provided when you registered. You can request that your password be mailed to you by clicking on the `Forgot your password?` link located in the left−hand column of the website. If you did not specify a valid email address when you registered, you will need to contact your site administrator.

Logging out

Once you are logged in to the Plone site, in the upper right hand corner you will see a link to `log out`. It is good practice to `log out` of Plone when you are finished using it.

Member Folders

Each member has a folder where they can create and store content. They are located in the `Members` folder and contain a default home page called index_html. To see your member folder, click on the `my folder` link in the personal bar in the upper right hand corner of the site.

Setting User Preferences

In the upper right hand corner there is also a `my preferences` link that opens the `Personalize form`. This form allows you to set a number of preferences that change how you interact with the site.

- ***E−Mail*** − This is the email address associated with your membership and is used a number of places in a Plone site. Most importantly, if you lose or forget your password, this is the address the system will send it to.
- ***Content editor*** − When editing complex content you may want the help of an editor. If your site administrator has made one available, you can select it here. It will then be used when you click on the edit tab of an object. *Default: None*
- ***Listed status*** − This property specifies whether your profile will show up on the Members tab and when someone searches the members listing. *Default: Listed*
- ***Form Help*** − Each form field has an associated pop−up box that provides context−sensitive help. This is very useful for newcomers to the system as they will be notified of the usage of a particular form field when they are filling it out. Users who don't want help can turn this off by selecting no. *Default: Yes*
- ***Display Names*** − Objects have a `Name` property that is used for the internal representation of the content object. The Name also shows up in the object's web address (also called a URL). By default these look something like: *News_Item.2002−11−16.4102* but you could make it much simpler such as *november_news* by changing the Name value. *Default: Yes*

Note: When you change an object's name value, anything that references the older name will no longer be valid and would result in the page not being found. It is best not to change the name value after you submit an object for review or link to it from elsewhere.

- ***Look*** − Some people like to change the look and feel of Plone and several `looks` or `themes` are provided to allow this. Select the look you would like for your Plone site. *Default*: Plone Default

- ***Portrait*** − In larger organizations and in community websites it is useful to see pictures of other members. The portrait field allows you to upload a picture that is 75 pixels wide and 100 pixels high. (If you upload a picture that is a different size, it will be resized to 75 x 100).

Once you have made the desired changes to your preferences, click `save` to commit them.

Adding and editing content

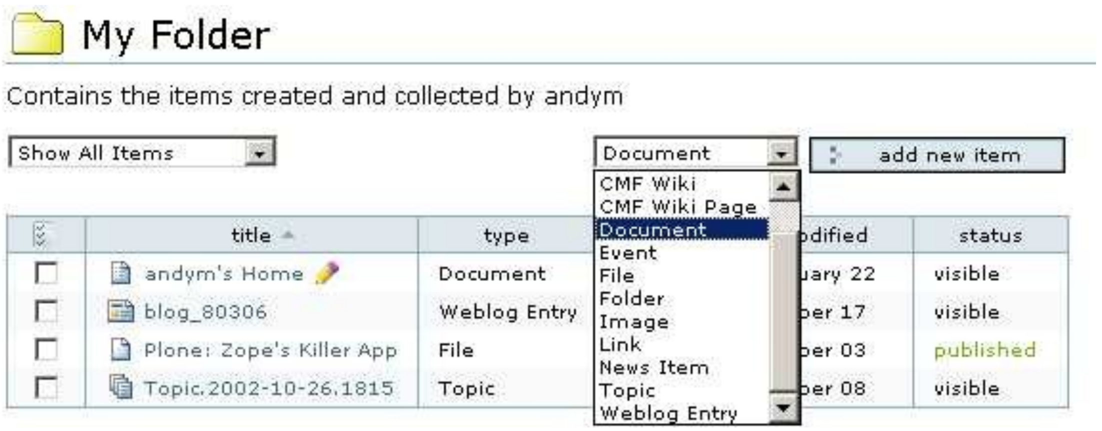
As a site member, you have a folder in the members section where you can store your content. Rather than detailing how to add and edit all the different types of content available, we will cover adding one type of content, a `Document`, in detail. All content types are added and edited in a very similar manner, so it is mostly a matter of repeating the steps here.

What is a Document?

A document is a page of content, usually a self−contained piece of text. Documents can be written in several different formats, plain text, HTML or Structured Text. The default home page for a Plone site is one example of a document.

Adding a document

To add a document, you will need to be in `Contents View` mode; the link is available on the left hand side of the page. This mode shows you the list of objects in the folder and lets you edit them. If you do not see the `Switch to Contents View` link, you do not have permission to add or edit content in this location.



My Folder

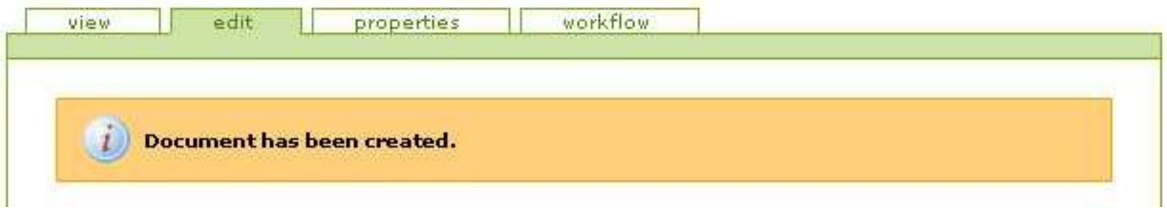
Contains the items created and collected by andym

	title	type	odified	status
☐	andym's Home	Document	uary 22	visible
☐	blog_80306	Weblog Entry	ber 17	visible
☐	Plone: Zope's Killer App	File	ber 03	published
☐	Topic.2002-10-26.1815	Topic	ber 08	visible

Once you are in Contents View, you will notice a drop down box in the upper right hand corner of the center panel next to the "add new item" button. This gives you the list of content types you can add in this folder. Select `Document` from this list, then click `add new item`. Once a document has been added you are taken immediately to the edit page for that document.

Editing a document

The document can be edited directly in the web browser, using the edit form. One thing to note is the highlighted edit tab at the top of the page. Messages appear at the top of this page, as shown below.



Document has been created.

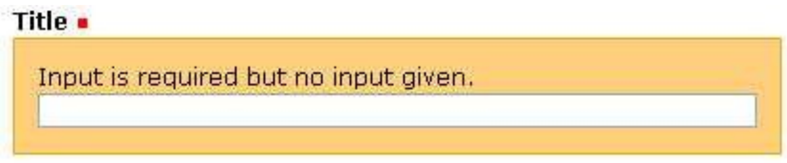
There are four fields for a document:

- ***Name***: This identifier will become part of the document's web address. It should be short, descriptive and contain no underscores or mixed case. For example, "audit−report−2003". If you do not provide a name, Plone will create one for you.
- ***Title***: This item will be shown at the top of the page, in the breadcrumbs, in the search interface, in the title of the browser and so on. This field is required.
- ***Description***: This is a short lead in to the document − usually no more than 20 words − to introduce the document and provide a teaser for the remainder of the document.
- ***Body Text***: This field contains the body of the document. The format for the content is set using the radio buttons below the field. They are:
- *Structured Text*: The default setting. Structured text is a format for taking plain text and producing HTML without the user having to learn or type HTML. Structured Text references are available by following these links:
    - An Introduction to Structured Text
    - Structured Text Help
- *HTML*: Any arbitrary HTML can be entered by a user.
- *Plain text*: Plain ordinary text with no mark up.

If you do have your document as a file on your computer you can upload this ***instead*** of typing it into the body text field. Use the upload button at the bottom of the page. The contents of an uploaded file will replace any content in the body field.

Once you have finished editing your document, click the `save` button. You will be returned to the view tab where you can see how the document will be rendered. To edit it again, click on the `edit` tab.

If you don't provide correct input on the edit form, when you save the document you will be returned to the edit page and your errors will be highlighted. At this point your changes have not been applied − you must correct the mistakes and click `save` again.



Input is required but no input given.

Assign properties to a document

Any object can have properties assigned to it. These properties are also known as metadata and provide information such as keywords, copyright and contributor information. The properties form has several fields that are common to all types of objects:

- ***Allow discussion***: This property lets this document be discussed by users who have the right to do so. If left in default, it will use the site wide policy.
- ***Keywords***: Keywords are a way of assigning metadata to an object. Use control+click to select multiple keywords from the list.
- ***Effective Date and Expiration Date*** `--` The effective date is the first day an object should be made available and the expiration date is the last day. Searches and navigation only show objects within this date range. Leaving these items blank makes an object infinitely available.
- ***Format***: The MIME−type of the object. If you don't know what this is, just leave it.
- ***Language***: The language in which the item is written.
- ***Copyright***: Copyright information for the object.
- ***Contributors***: The names of the people who contributed to the object. Each person's name should be on its own line.

Publishing your document

Documents are created in the `visible` state, which means people can access them, but they will not show up in the site navigation tree. Visible documents are available through the search feature and by linking directly to their URLs.

When you are satisfied with your document you will need to submit it for publishing. To do this, select the `state` tab. This page gives access to the publishing process.

- *Effective Date*: allows you to specify a date this content is effective from. Until this date is occurs, the content will not be published. If this date is not specified, then there will be no effect and the document will be published.
- *Expiration Date*: allows you to specify a date this content will expire on. After this date the content will no longer be published. If this date is not specified, then there will be no effect and the document will be published.
- *Comments*: provides an interface to add comments to this publishing process. These comments can be read by the person reviewing your content in the publishing process. Similarly if the reviewer rejects your content, they will be able to put comments here so that you will know why your content has been rejected.
- *Change state*: these are the states that the document can be saved to. Publishing and reviewing content is covered in more detail in Chapter 4 of this book, but for the moment you will want to publish your content, so select `Submit` and then click `save`.

Your document will now be in the `pending` state. A reviewer will review your content and decide whether to publish it as−is, edit it and publish it, or reject it.

Adding and editing other forms of content

There are several different object types that come by default in a Plone site. Adding and editing these object types is a similar process to the one described above for documents.

Image

Images are graphical pieces of content. This content type usually ends with an extension such as: gif, jpg, png, tif or pict. Images can be displayed inside of the Plone CMS without having to download them to the local computer if the image type (extension) is viewable in the user's web browser.

When you add an image, the id of the object is changed to be the file of the image. So if you upload an image called photo.gif, it will be accessible in Plone as photo.gif. When adding or uploading a new image you can select the image from your file system by using the `browse` button and selecting the file.

It is noteworthy that Macintosh .pict files are often not viewable by Microsoft browser platforms.

File

A file is any arbitrary object that can be uploaded from your file system. This could be any sort of item such as a plain text file, a Microsoft Word Document, an Excel Spreadsheet, a PowerPoint Presentation, an Acrobat PDF and so on. When you add a file, the id of the file is changed to be the name of the file. So if you upload a file called book.pdf, it will be accessible in Plone as book.pdf.

Link

Link objects are the primary way for users to share URL's. These URL's can be internet resources or local resources. Links can contain metadata like any other content object. Please note that if you are going to link to a internet resource you should preface your link with the suitable protocol (e.g. http://), otherwise your link will possibly be incorrect.

Topic

Topics enable users to create collections of resources by querying the central information repository. The collection is defined on the criteria tab. The criteria specified will match all content objects that are cataloged by the system. You can query a number of different aspects of the system: by physical location, time created, review_state, and many other facets. Topics only link to other resources; they do not keep physical resources inside them like normal Folders.

By default only users with the `Manager` role are allowed to create topics.

Folder

Containers are the simplest and most powerful mechanism for organizing content. A Folder is a container that can house any sort of content object, such as Files, Documents, or any other content type. By default all content types can be added to a Folder.

News Item

News Items are commonly used in websites. Published News Items show up in reverse chronological order on the `News tab` and in the News section.

Discussing content

Any piece of content in Plone can be discussed. The owner of the content (otherwise known as the person who created it) turns on the discussion feature by clicking on the `Properties` tab of the object and checking `Allow`. The `default` radio button is the policy for the content object that has been set by the Site Administrator.

If discussion is enabled, when viewing content the discussion will be shown and users will be allowed to participate.

Searching for content

There are two ways to search for content in Plone. At the top of your Plone site there is a search box that provides an easy way to do simple keyword searches. You can narrow down the search results by using the `Advanced search` functionality. This is accessible by clicking on the `search` tab at the top of page.

In the `Advanced search` form you are able to query content by a number of attributes including: title, keywords, description, review state, creation date, content type and even author.

In this chapter we have covered some of the basic elements such as adding and publishing content, searching and altering your Plone instance to suit your needs. Although each Plone site is different, each will have these basic elements.

$Id: 3,v 1.5 2003/09/14 16:23:22 yenzenz Exp $

Previous [Chapter 2: Installing and Upgrading] | Contents | Next [Chapter 4: Workflow]

Previous [Chapter 3: Using Plone] | Contents | Next [Chapter 5: Configuring Plone]

- Reviewing objects
- Editing an object after it has been published

Chapter 4: The Workflow System

Workflow is the process used to manage objects in a website. An example is a company's press release: an employee writes a press release and submits it to an editor for review before it is published on the website. This review process is called a workflow and is used by site managers to ensure that site content is correct. Plone has a very powerful and flexible default workflow system that is built around `Object States` and `User Roles`.

Object States

An object's state determines whether it is available to the various types of users defined in Plone, and what other states that object can be transitioned to. Plone's default workflow includes four states: visible, pending, published and private. Site managers and developers can create custom states − these are explained in more detail in Chapter 5.

- By default, objects are created in the `visible` state. All users can find visible objects through the search function and can access them directly by visiting the object url. Visible objects do not show up in the navigation tree. Visible objects within private folders are still visible to all users and available through the search function. Visible objects are editable by their owners and site managers.
- `Pending` objects have been submitted for publishing by site members. From an end−user standpoint, they behave like objects in the visible state. The difference between the two types is that pending objects are flagged for review; site reviewers are prompted to publish or reject pending objects. Pending objects are editable only by managers.
- `Published` items are visible to all site visitors. They appear in search results and the navigation tree. When a News Item becomes published it becomes visible under the News tab and also in the News box (see Chapter 5). Published items are editable only by managers, but can be retracted by owners for editing (retracting reverts an object to the visible state).
- Objects in the `private` state are visible and editable only by their owners and others with manager access to the folder in which they exist. They will not appear in search results or on the navigation tree for other users. Private items are editable by managers.

Note: Although most objects have their own states, some inherit their states from their parent objects. Forums are an example of this behavior. Forums within visible or published folders are available to all users through the search function. Forums within private folders are only available to users with access to that private folder.

User Roles

Plone uses roles to define what different users can see and do. In this way, Plone builds security into every aspect of its operation. The roles defined in a default Plone installation include anonymous, member, owner, reviewer and manger.

- Anyone who visits the site and does not log in takes on the `anonymous` role. In public sites, anonymous users can see published, pending and visible content by directly visiting a url or by searching, however only published content is visible on the site navigation tree. If you have set your site up as private, anonymous users cannot see anything.
- `Members` are users who have logged in to the site. Members have the added ability to create content in their own folder, which is then submitted to site reviewers to be published. In public sites, members

can see the same content as anonymous users. In private sites, members have access to published, pending and visible content. Members may also set their own preferences.

- As a `reviewer`, you can publish or reject content submitted by members. When a reviewer logs in, if there is content to be reviewed a "pending" message will appear in their personal bar. A review list also appears in the right−hand column of the page. Reviewers have access to the same content as members.
- Members have the `owner` role for all content they create. This allows them to edit the content, submit or retract it, or make it private. Assigning the owner role to other users is not recommended.
- Site `managers` can see content in all states (visible, pending, published and private). In addition to all the capabilities of the member and reviewer, managers can add, edit, delete and move content. Managers can also add, edit and delete users and assign them roles.

Plone inherits the role model from Zope itself. Please refer to the Zope Book for further information.

Transitions − Changing Object States

Owners and managers can change the states of objects they control. The states that are available are controlled by pre−defined transitions. For example, site members can submit visible objects for review or make them private and site reviewers can publish submitted items or reject them. Site managers can also customize this portion of the workflow system. (See Chapter 5)

To change an object's state in contents view, check the box next to the name of the object to be changed and then click the change status button at the bottom of the page. Scroll to the bottom of the Publishing Process page and choose the desired state, then click submit.

One can also change states in item view. Simply click on the state tab, scroll to the bottom of the page, choose the desired state and click submit.

Local Roles

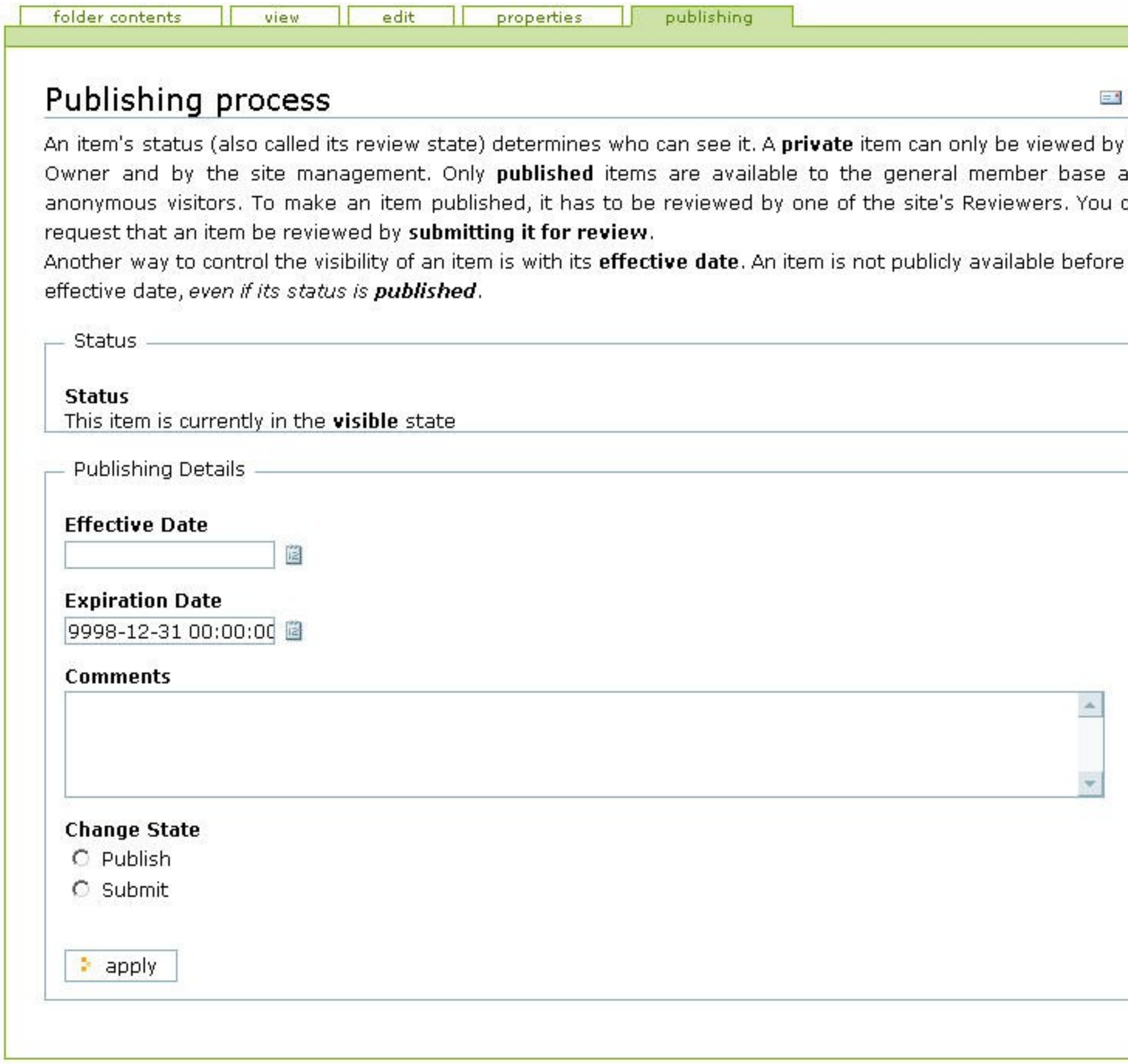
Site managers can give specific users additional rights in certain sections of the website. This can be accomplished by assigning local roles to folders. Managers and owners have permission to assign local roles.

To assign a local role, switch to contents view of a given folder and click Local Roles on the content tab. Search for the name of the user to whom you wish to assign a role, and check the box above their name. The choose the role to assign from the drop−down list and click "assign local roles to selected users".

This same interface can be used to delete a local role.

Publishing an object

Object owners can alter and edit objects in the `visible` state. Once an owner finishes editing an object, it can be published by clicking the `State` tab. This opens a publishing form:



Publishing process

An item's status (also called its review state) determines who can see it. A **private** item can only be viewed by
Owner and by the site management. Only **published** items are available to the general member base a
anonymous visitors. To make an item published, it has to be reviewed by one of the site's Reviewers. You c
request that an item be reviewed by **submitting it for review**.
Another way to control the visibility of an item is with its **effective date**. An item is not publicly available before
effective date, *even if its status is **published***.

Status

Status
This item is currently in the **visible** state

Publishing Details

Effective Date

Expiration Date
9998-12-31 00:00:00

Comments

Change State
Publish
Submit

apply

This form gives you several options:

- ***Status*** −− Tells you the current state of the object.
- ***Effective Date and Expiration Date*** −− The effective date is the first day an object should be made available and the expiration date is the last day. Searches and navigation only show objects within this date range. Leaving these items blank makes an object infinitely available.
- ***Comments*** −− Here you can enter any comments or reason for the change in state you are requesting. These comments will be preserved in the change log for the object.
- ***Change State*** −− The states to which a user can move an object. The states available here will depend on your role. Members can submit objects or make them private. Once submitted, objects are available to users with reviewing priveleges, who can then publish or retract them.

Reviewing objects

If you have reviewing privileges, when you log in to your Plone web site, you will be presented with an indication if any objects are awaiting review. This is shown in the personal bar as shown here:

Select this link to get a list of the objects ready for review, then select each object to review it. Click on the publishing tab and you will be presented with a form like the one above. At this point you have basically two choices − either approve an object and publish it (the *publish* option) or reject it (the *reject* option). If you reject the object you will probably want to provide a comment to the user explaining why you have rejected it.

Editing an object after it has been published

Managers may edit objects that are in any state. Members cannot edit an object after it has been published; they must first move it back into a visible state by retracting it. To do this, click on the publishing tab for the object and click `retract`. Then you can edit the object as much as you wish and resubmit it. This ensures that the review process is maintained for objects that have been edited.

$Id: 4,v 1.6 2003/09/14 16:23:22 yenzenz Exp $

Previous [Chapter 3: Using Plone] | Contents | Next [Chapter 5: Configuring Plone]

Previous [Chapter 4: Workflow] | Contents | Next [Chapter 6: Changing the look and feel]

Chapter 5: Configuring Plone

This chapter shows you some of the configurations that you can make to Plone through the web interface, without writing any code.

The Zope Management Interface

Plone is built on top of Zope and the Content Management Framework (CMF). Zope has a built in Management Interface that is accessible through the web. For most day to day activities with Plone, such as adding and editing content you will not need to access this. However to manipulate some of the more advanced features of Plone you will need to access this Management Interface (ZMI).

Accessing is as simple as adding /manage on to the end of your URL, for example: http://localhost/manage. If you are using the Mac or Windows installer you can access the site using port 80 and the root ZMI through port 8080. For example Plone will be accessed via http://localhost and the ZMI via http://localhost:8080/manage.

Navigation is given in the left hand frame and objects are managed in the right. For more information on the ZMI please see the Zope Book Chapter X.

Note: if you log in to the root of the ZMI this will log you in via HTTP Basic Authentication which has no method of logging you out. This means you will be unable to log out of your Plone site, without closing your browser. This is not a bug in Plone, but rather a lacking in the HTTP Basic Authentication API. For testing it's usually easier to have two browsers open, one logged in to each.

Layout of a Plone Site in the Management Interface

The Plone Site is seperated into a series of objects that are either content or tools. When accessing Plone through the web you cannot see the tools that handle a great deal of Plone's work, these are covered here. Not all the tools are unique to Plone, most of them come directly from the CMF. For more information on some of these please see the relevant CMF documentation.

...

Actions

In Plone there are certain things that can be performed at different times by different people in different parts of the site. Things like search, browse the folder, log in and so on, these are called Actions. Plone translates them into tabs, links and other elements. They are a highly configurable way of provide navigational elements for a site. Actions are edited through the Zope Managment Interface (ZMI) and are contained in a few places, such as `portal_actions`. Tools that have actions are are called tool providers.

Actions in general

Each action has the following properties that can be figured in the ZMI:

- Name –– a nice name for the action, this name is often used in UI, for example when the action is used as a tab, this is the text in the tab.
- Id –– a unique id (to this current tool provider) for the action
- Actions –– the action that is to be performed. For example when the action is used as a tab, this action is used as the link. This field is a tal expression, see the ZPT documentation for more information.
- Condition –– a condition that has to occur in order for the action to be used. For example when used as tab, if this condition is met the tab will appear. This field is a tal expression, see the ZPT documentation for more information.
- Permission –– the permission the user has to have in order to have this action. This permission has to be met in order for the action to be used.
- Category –– is used to categorise the actions. In Plone this used to distinguish the actions so they are used in different sections of the UI. The following categories are used:
    - folder –– actions in this category appear on folders.
    - object –– actions in this category appear on objects.
    - portal_tabs –– these actions appear at the very top of the Plone site the default ones are "Welcome", "Members", "News", "Search" and so on. Actions in this category are normally related to the entire portal, as opposed to particular content.
    - object_tabs –– these actions appear in the middle of the page at the top of the content. These actions are usually related to the content, in many cases they do not show up unless the current user has the right to edit the content.
    - folder_buttons –– are the buttons that appear on the folder contents page and relate to copying or moving content in a folder, such as copy and paste.
    - user –– are actions that relate to a user, such as "Log in" and "Join"
- Visible –– if the category is `active` or not. Since actions usually relate to visual elements, the term visible is used.

In Plone, the order of the actions in each category as presented in the ZMI determines where it appears in the skin. For example "Welcome" is the top most action and appears at the far left of the portal tabs at the top of the Plone site. Select the action and use the `Move up` and `Move down` buttons to move the actions up or down the list.

Action Providers

The action providers are a little spread out unfortunately and it sometimes can be a little hard to find where the action appears. To get a list of the current action providers in your Plone site, in the ZMI go to `portal_actions` and select `Action Providers`. This list all the objects that are able to have actions.

Portal actions (`portal_actions`)

Portal actions is the generic hold all for all actions that are not in other providers. The most commonly used actions contained here are the actions in the `portal_tabs` category.

Portal Membership Tool (`portal_membership`)

Contains the actions for a user that relate to being a member. These are usually all of the `user` category. These actions appear under the portal tabs in the standard Plone skin. Most of these actions have permissions relating to the user being logged in or not.

Portal Registration (`portal_registration`)

Contains actions for registering a user. In the default Plone there is only one action `join`.

Portal Discussion (`portal_discussion`)

Contains actions for discussions. In the default Plone there is only one action `reply`

Portal Undo (`portal_undo`)

Contains actions for undoing content, such as `undo` and `quick undo`. These provide the interface for users to undo content so that they can revert to previous versions of content.

Portal Syndication (`portal_syndication`)

Contains actions for syndication content. If you wish to syndicate content you must make this action `Visible`. Then a "Syndication" tab will appear on folders and give the user option to syndicate the content.

Portal Workflow (`portal_workflow`)

Actions here are contained in the workflow and as such are a key part of the workflow. Workflow is discussed in chapter XXX.

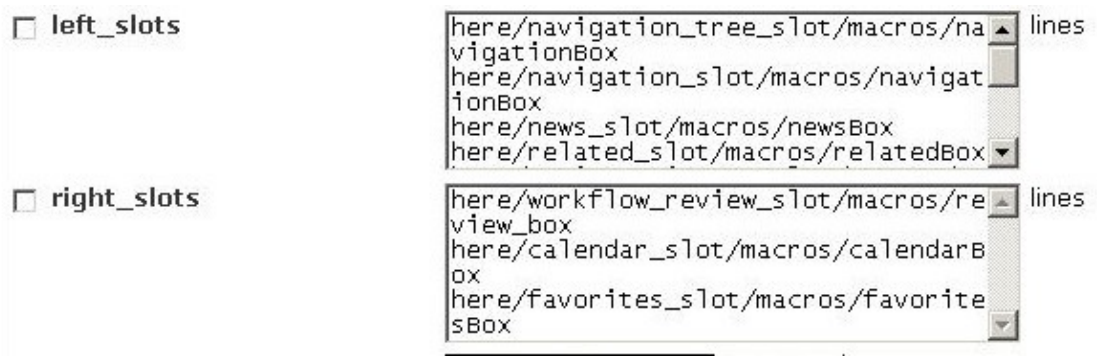
Portal Setup (`portal_properties`)

Contains an action for modifying the portal, such as `Plone setup` which allows you to modify some of the key plone parameters.

Slots

A slot is a little section of a web page that has a self−contained piece of information. In Plone these slots are found on the left and right columns of the page and contain information such as the calendar, navigation, login box and so on. These slots can present dynamic or static information to the user and can be easily configured through the ZMI.

To configure the slots you need to access the ZMI and select the "Properties" tab on the object you want to edit. In Plone's case this means selecting your Plone instance, then "Properties". This shows a list of properties including `left_slots` and `right_slots` as shown below:



```
left_slots    here/navigation_tree_slot/macros/navigationBox
              here/navigation_slot/macros/navigationBox
              here/news_slot/macros/newsBox
              here/related_slot/macros/relatedBox        lines
right_slots   here/workflow_review_slot/macros/review_box
              here/calendar_slot/macros/calendarBox
              here/favorites_slot/macros/favoritesBox    lines
```

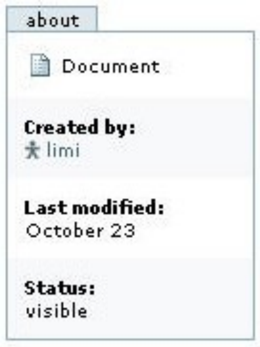
The `left_slots` refer to slots shown on the left of the page, the `right_slots` lists the slots shown on the right of the page, in the order they are given. The slots properties can be edited here and they will affect the whole site, or you can edit the properties on each folder to affect pages below it, for example in the standard Plone install you will notice that the Members folder has a different `right_slots` property.

Not all of the slots available are configured in Plone. Following are details of each slot in Plone, its file location and the path expression that needs to be added in to the slots so that it will show up.

About Slot

Filename: *plone_templates\ui_slots\about_slot.pt*

Path Expression: *here/about_slot/macros/aboutBox*



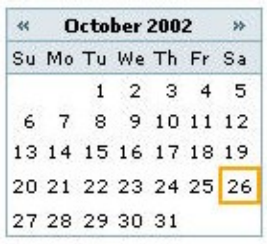
about
Document
Created by:
limi
Last modified:
October 23
Status:
visible

This shows information about the current object, such as who created it, when it was last updated and so on.

Calendar Slot

Filename: *plone_templates\ui_slots\calendar_slot.pt*

Path Expression: *here/calendar_slot/macros/calendarBox*



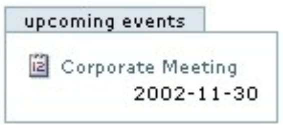
Su	Mo	Tu	We	Th	Fr	Sa
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

This displays the Plone calendar. This can be configured using the portal_calendar tool in the ZMI and shows the selected object types on the calendar.

Events Slot

Filename: *plone_templates\ui_slots\events_slot.pt*

Path Expression: *here/events_slot/macros/eventsBox*

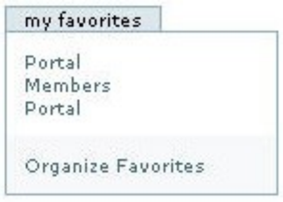


Displays a list of the upcoming events. Even if you have this item in the slot list it will not show unless there are some published events.

Favorites Slot

Filename: *plone_templates\ui_slots\favorite_slot.pt*

Path Expression: *here/favorites_slot/macros/favoritesBox*

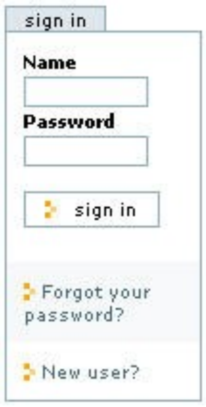


Displays a list of the users favorites and a link to organize them. Even if you have this item in the slot list it will not show unless the user has some favorites.

Login Slot

Filename: *plone_templates\ui_slots\login_slot.pt*

Path Expression: *here/login_slot/macros/loginBox*



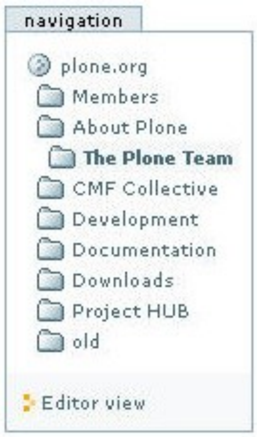
Displays the login box so user can login. Even if you have this item in the slot list it will not show if you are logged in.

Navigation Slot

Filename: *plone_templates\ui_slots\navigation_tree_slot.pt*

Path Expression: *here/navigation_tree_slot/macros/navigationBox*

Note: the old version of this slot navigation_slot, is deprecated

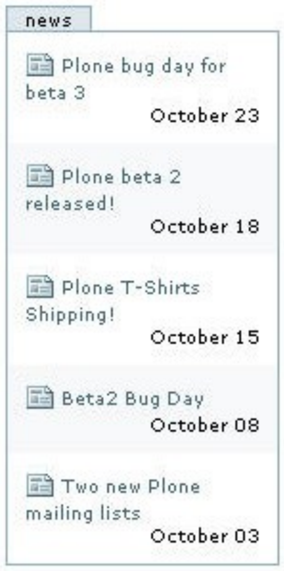


The navigation slot shows a simple tree of the folders in the current position in the form of a tree. It provides a very powerful and simple navigation tool. The navigation slot is extremely customizable and can be altered by accessing `portal_properties/navtree_properties` inside the ZMI.

News Slot

Filename: *plone_templates\ui_slots\news_slot.pt*

Path Expression: *here/news_slot/macros/newsBox*

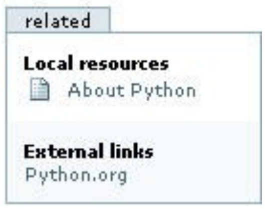


Shows a list of all the recent news items, with links to them. Even if you have this item in the slot list it will not show unless some news is published.

Related Slot

Filename: *plone_templates\ui_slots\related_slot.pt*

Path Expression: *here/related_slot/macros/relatedBox*

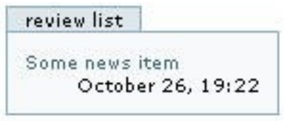


Shows a list of all the related items, as determined by the keywords. For example in the above image, the current page has the keyword Python and all other pages with that keyword are shown. Items are seperated into internal and external resources. Even if you have this item in the slot list it will not show unless there is some related objects.

Review Slot

Filename: *plone_templates\ui_slots\review_slot.pt*

Path Expression: *here/review_slot/macros/reviewBox*



Displays a list of objects that are in review status and are awaiting to be reviewed. This is only shown if the user has review status and there are items awaiting review. The number of items is also shown in the navigation bar as seen below.

Writing your own Slot

The use of path expressions to produce the slot is a rather poweful tool. It enables you to put almost any content from any object in a slot, Plone will just find the relevant result of the expression and display it. For example, if you have a Poll product and wanted to show this in a slot could do the following `here/polls/funnylaughter/frontpage_poll_view` (*note:* this is an example, this path expression will not work in a standard Plone install). Path expressions are key part of Zope and Plone, for more information please see: ...

If you wanted to just write your own custom slot, you would most likely add a Page Template with the correct HTML and TAL classes for more information please see: ...

Navigation and Forms

...

Syndication

Syndication is the ability to access a folder's contents through RSS (Really Simple Syndication or Rich Site Summary). RSS produces a list of all the objects in the folder. This summary can then be accessed programatically through many different news reading programs.

To enable syndication in the ZMI go to the portal_syndication object. There is one action for folders called "Syndication", this is not visible by default. Check the "Visible" check box to ensure the action is visible. Syndication for the site will also need to be enabled, click on the "properties" tab and click "enable syndication".

Any folder in your Plone site will then have a Syndication tab. You have to enable syndication for every folder you wish to have syndication for. Go to the folder you wish to enable this for and click the "syndication" tab. There is a simple button to enable syndication click this to enable syndication, if you get the message "Syndication not allowed" return to the portal_syndication object and ensure you enabled syndication in the properties tab.

Some of the values on the syndication page, after enabling it:

Update period
: how regularly the client should come and check that the RSS feed is up to date.

Update frequency
: ...

Update base
: ...

Maximum Items
: the maximum items to display

Note: syndication does not recurse through every folder below, it is for the current folder and its published contents only.

Page Editor: Andy McKay $Id: 5,v 1.5 2003/07/03 18:10:41 zopezen Exp $

Previous [Chapter 4: Workflow] | Contents | Next [Chapter 6: Changing the look and feel]

Previous [Chapter 5: Configuring Plone] | Contents | Next [Chapter 7: Plone Style Guide]

Chapter 6: Customizing Plone's Look and Feel

This chapter shows you how to modify almost any aspect of the look and feel of your Plone site, by explaining the elements of a page, the skins and how it all slots together.

A few basic concepts

Skins

A skin is a visual wrapper around content. It may contain small portions of logic directly releated to that visual representation as well. When you install Plone you will see the default skin, this is the one most users are familiar with and the default one you will see at plone.org and other sites. By writing a new skin you can change the way Plone looks, although it will still have that similar Plone feel − zopezen.org or zopera.org for example.

But Plone doesn't have to look at all like or be even vaguely recognisable, as a Plone site, if you put a totally custom site. Take, for example, the list of sites found here − all of them provide a totally different and custom experience for the user. In most cases the sites can easily flip between skins providing different skins for users. Many site use the power and flexibility of the Plone interface internally so users can add and edit content easily, whilst providing a totally different external view.

Skins are defined in the ZMI in `portal_skins` tool. When in portal_skins click on `properties`. This will present a list of the skins available on this site. For example there is a skin called "Plone Default". In the form you will see an entry for this skin and some values next to it, these are layers.

In the `properties` tab a user can alter the default skin to be shown to users and or allow users to change their skin. If users are allowed to change their skins, they can go to the `my preferences` screen and change it there.

Layers

Each skin contains `layers`. A layer is an individual set of templates and scripts that get presented to the user, by combining these layers a skin is formed. This allows a user to easily add or remove components to a skin by altering these layers. Again for example in the "Plone Default" skin (accessed through the ZMI by going to portal_skins −> properties), has a series of layers, "custom, ..." and so on. These are the layers for this skin.

Two things are important to know about layers:

- the order of layers is important, the top most layers will be examined first for an object
- each layer is an entry in portal_skins −> contents, and is usually a Filesystem Directory View or a Folder

Changing the logo, a quick example

After reading the above, you are probably itching to try altering something. The most commonly asked question is "how do I change the logo in the top left hand corner from the Plone logo". This is a very simple thing to do.

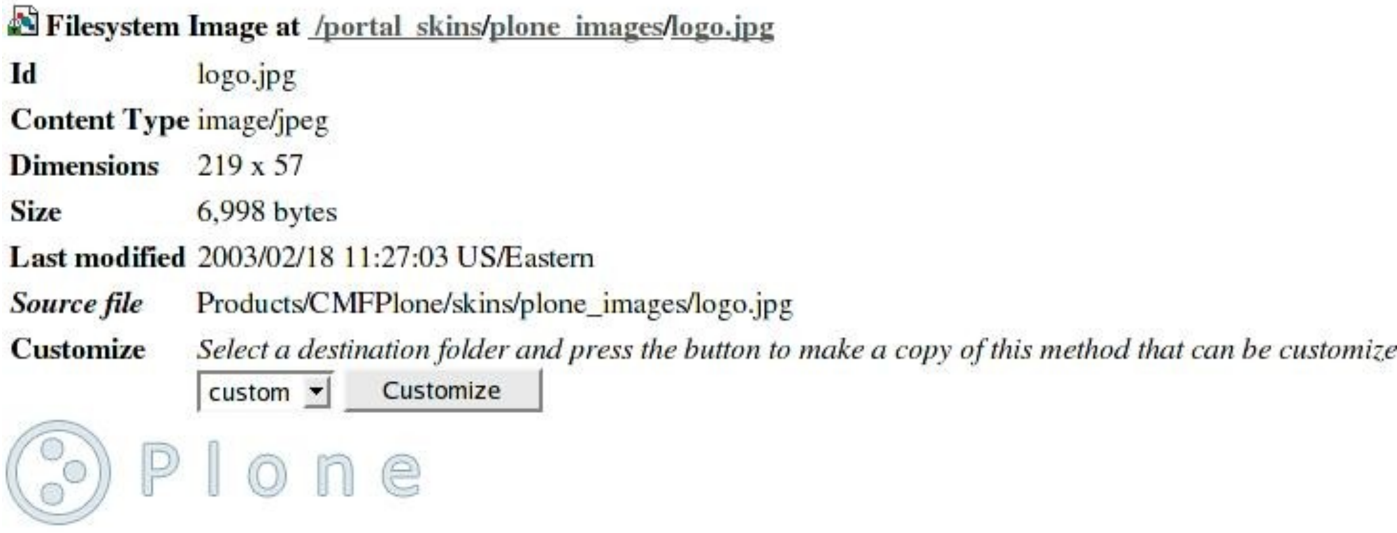
- in the ZMI go to portal_skins –> plone_images
- find logo.jpg in the list and click on it
- click on the `Customize` button
- upload a new image using the browse form and click `Upload`

Thats it, return to your Plone interface, the image has changed. Simple.

Changing the logo, the explanation

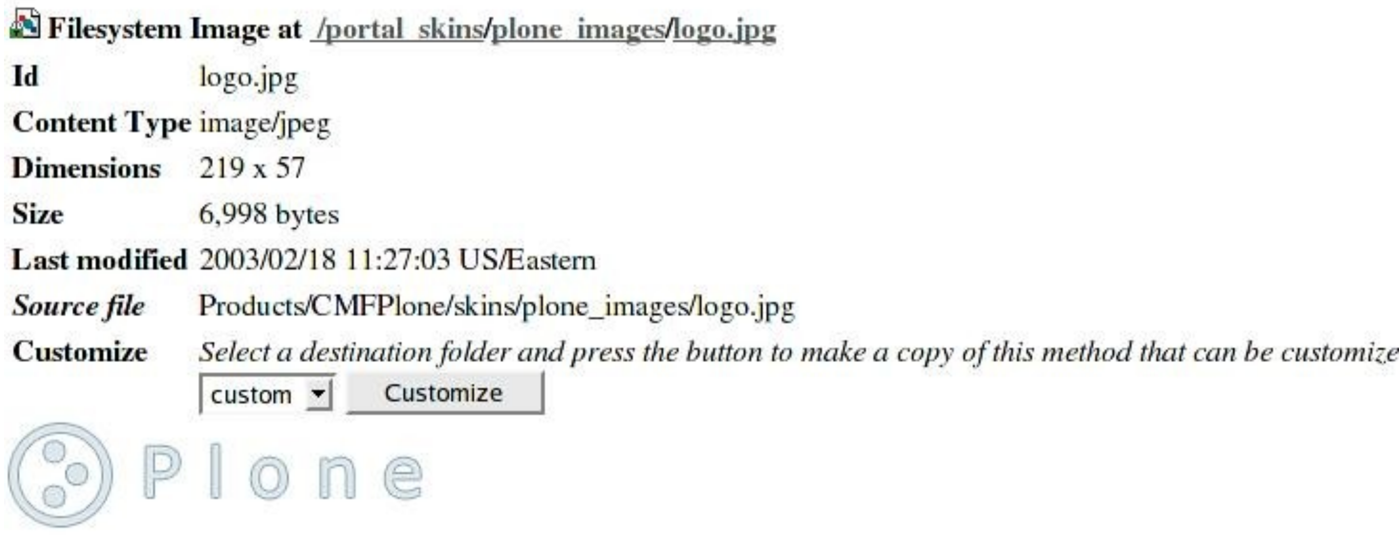
In the above example we showed how to change the logo for the main page. This works because we have used one of the key features of skins, the layers as described above. One of the most important thing about layers is the search order, the top most layer (the first one in the list) will always be examined first to find the item looked for.

So first let's look at the layers for the "Plone Default" skin. The first layer is "custom", this is the custom folder found at portal_skins –> custom. Somwhere down the list you will find the `plone_images` layer. This maps to a Filesystem Directory View and these files can be found on the file system inside your Plone installation. When you click on the image (portal_skins –> plone<u>images</u>, you are presented with no option to edit the image, but you can customise it, the text on the screen shot below helps explain it.


Filesystem Image at /portal_skins/plone_images/logo.jpg

Id	logo.jpg
Content Type	image/jpeg
Dimensions	219 x 57
Size	6,998 bytes
Last modified	2003/02/18 11:27:03 US/Eastern
Source file	Products/CMFPlone/skins/plone_images/logo.jpg
Customize	*Select a destination folder and press the button to make a copy of this method that can be customize*

You cannot edit this image, because it exists on the file system. You are given the default folder `custom` in the drop down list. Clicking `customise` makes a copy of the image inside the custom folder inside the ZODB and takes you directly to it. If you look carefully you will see the path to the image has changed, the meta type has changed and now you can edit the image.


Filesystem Image at /portal_skins/plone_images/logo.jpg

Id	logo.jpg
Content Type	image/jpeg
Dimensions	219 x 57
Size	6,998 bytes
Last modified	2003/02/18 11:27:03 US/Eastern
Source file	Products/CMFPlone/skins/plone_images/logo.jpg
Customize	*Select a destination folder and press the button to make a copy of this method that can be customize*

Why is this done, because in our list of layers, the custom layer comes before the plone_images layer, since this image has been copied into the custom folder this image is now the first image looked up. So when the browser requests `logo.jpg`, it will look through each layer, finding the image in custom first, it will return that image.

So far, so good. Now in step 4 above we uploaded a new image. This altered logo.jpg and since that image will be found in the list first.

This is the basic concept of skins and layers in Plone. By altering the layers and moving individual elements between layers you can easily alter and customise your site. It allows you to customise as little or as much of your site as you want.

Customizing the rest of the site

Before you delve too far into customising the site and page templates, the first place to look are the stylesheets.

Style sheets

Most of Plone's style is controlled from stylesheets and by configuring these stylesheets you are able to radically alter Plone. There are four stylesheets found in portal_skins −> plone_styles:

- ploneCustom.css: is a the custom stylesheet that is actually blank. You should customise this stylesheet first. *Note:* This sheet added in Plone 1.1.
- plone.css: is the main stylesheet that controls most of the setup for the site, if possible alter ploneCustom.css first.
- ploneNS4.css: is the stylesheet for Netscape 4
- plonePresentation.css: contains code for Opera's presentation mode
- plonePrint.css: contains code for printing pages

One further element, stylesheet_properties contains all the actual definitions of colours, fonts and sizes that are used in the above stylesheets. Customising these properties is an easy way to alter the look and feel. For example if you customise this object and you can alter the main font, by altering the mainFont property from "65% Verdana, Helvetica, Arial, sans−serif".

The other folders in portal_skins −> plone_styles are the skins that come with Plone and provide good examples of different sites.

Images

All images (except specific products) are contained in plone_images.

Plone content

Contains the templates relating to content. These are all Page Templates, which is Zope templating languages, used mostly for HTML and used almost exclusively for Plone. Page Templates are an extremely powerful and elegant templating language[1]. By altering any of these templates you can alter how content is represented.

Plone templates

This contains probably the most important part of the site, the main templates, the standard header and footer and so on.

Really the main trick to customising a new site is finding the parts you need to customise. This is actually easy to do. First find the URL of the page you are calling, if there is a last element such as `login_form` or `document_edit`, then these will directly relate to objects in the portal_skins directory. To find the actual object go to portal_skins −> find and enter the last element of the URL. For more complicated elements you might need to examine the portal_types tool to see what action is being called. If all else fails, use find or grep to find the files on the file system.

Making a totally new skin

Of course modifying skins is all well and good, but at some point you will want to make a new skin.

- go to portal_skins and add a folder, give that folder a name such as `alpha`
- then go to portal_skins −> properties and add a skin, to do this enter the name and then the list of layers, just enter `alpha`

For a new skin you may or not want to add in previous Plone and CMF elements, this is up to you, if you do add them in after the new alpha layer. That's it, you now have a new skin called alpha that picks items up primarily from alpha.

Example site

One example site that uses Plone is ZopeZen.org and all the code for ZopeZen's skin is available. Futhermore a set of three articles discuss how this skin was made. Although with new Plone releases this has unforunately become a little outdated in parts, most of it is still revelant.

- "Building ZopeZen, Part One": http://www.zopezen.org/Members/zopista/News_Item.2002−09−30.2355
- "Building ZopeZen, Part Two": http://www.zopezen.org/Members/zopista/News_Item.2002−10−02.2007
- "Building ZopeZen, Part Three": http://www.zopezen.org/Members/zopista/News_Item.2002−10−09.3743
- Code and skin

Other configurations

Tabs

Tabs are determined by "Actions" and "Action Providers", covered in Chapter 5 of the Plone book. Tabs are generated by creating action entries. The category field affects the tabs in the following way:

- if the category is `portal_tabs` it will show up in the top of the site (for example: News)
- if the category is `object_tabs` it will show up in the green editable border (for example: Edit)
- if the category is `folder_buttons` it will show up as a button in the folder_contents (for example: Folder Contents)

Example: Adding a tab on Plone.org for documentation

One common task is to add a tab to the top of portal pointing to a folder of content. For example on Plone.org all the documentation is in a folder called `documentation`. Since we want users to be able to easily access the documentation an action was added.

- In the Plone interface:
- Create a folder called `documentation` in the root of your Plone site.
- Publish this folder (if necessary).
- In the ZMI:
- Go to `portal_actions`
- Scroll to the bottom of the page for the `Add` section and enter the following:
- Name: documentation
- Id: documentation
- Action: string:$portal_url/documentation
- Condition: *leave blank*
- Permission: View
- Category: portal_tabs
- Visible: checked
- Select `Add`

You now have a tab visible to everyone for documentation.

Dates

Plone presents dates in a certain format for easy reading. Those formats can be altered from the Zope Management Interface. There values are stored as properties on the Plone site. In the ZMI got to portal_properties −> site properties.

- localTimeFormat is the short time and date
- localLongTimeFormat is the time and date in a longer format

Both these values use Python's time format module to produce a time format. The reference for the formats can be found here − If you are unfamiliar with this module, here is a quick summary.

- %a Locale's abbreviated weekday name. (eg: Mon)
- %A Locale's full weekday name. (eg: Monday)
- %b Locale's abbreviated month name. (eg: Jan)
- %B Locale's full month name. (eg: January)
- %d Day of the month as a decimal number.
- %H Hour (24−hour clock) as a decimal number.
- %I Hour (12−hour clock) as a decimal number.
- %m Month as a decimal number.
- %M Minute as a decimal number.
- %S Second as a decimal number.
- %y Year without century as a decimal number.
- %Y Year with century as a decimal number.

For example the default setting for localTimeFormat is `%b. %d, %y`. This formats the date so that is in the format `abbreviated month. day number, year in two digits`, for example: `Oct. 24, 02`. If we want to included the day name, this would be a simple matter of changing localTimeFormat to read `%A, %b. %d, %y`. This would produce the following: `Thursday, Oct. 24, 02`.

Plone's page rendering

Credits: *Jean Jordaan* for initial documentation.

This covers how a Plone page is rendered so you can alter or edit this rendering to suit your own site. Presentation of content in Plone is via a page template called `main_template.pt`, that can be found on the file system in `skins\plone_templates` and in through the ZMI for customization at portal_skins −> plone_templates −> main_template.

Now of out date, needs updating for 1.1

[1] Page Templates are explained in the Zope Book see Chapter 6 and Chapter 13 for more information.

$Id: 6,v 1.5 2003/07/05 15:41:48 zopezen Exp $

Previous [Chapter 5: Configuring Plone] | Contents | Next [Chapter 7: Plone Style Guide]

Previous [Chapter 6: Changing the look and feel] | Contents | Next [Chapter 8: Extending Plone]

Chapter 7: Style Guide

This chapter covers the extending Plone, by altering or adding to the code and HTML so it can be customised for your needs.

Plone HTML

Credits: Laura Trippi

Temporarily discontinued, out of date

Plone Colours

Credits: Laura Trippi

Temporarily discontinued

Plone Stylesheet

Credits: Laura Trippi

Temporarily discontinued, out of date

$Id: 7,v 1.4 2003/06/23 05:11:19 zopezen Exp $

Previous [Chapter 6: Changing the look and feel] | Contents | Next [Chapter 8: Extending Plone]

Previous [Chapter 7: Plone Style Guide] | Contents | Next [Chapter 9: Optimizing Plone]

Chapter 8: Extending Plone

This chapter covers how to extend and expand Plone beyond its default installation. This is a huge topic that could form several books all on it own. Here we hope to cover some of the more common ways to extend Plone.

Creating a new content type

There are a lot of item types included in plone and cmf. However there may come a time, where the types included do not suffice your needs. This section will teach you how to add your own extensions to your plone instance.

Adding your own types may require a little knowledge of python. There are several ways to create plone extensions:

- Repurposing existing content types
- Creating a new Zope Product
- Archetypes

Note: Some of this can also be done using ZClasses. We do not recommend or cover the use of ZClasses for this. There is an excellent How To on creating content types using ZClasss here:

Repurposing existing content types

A content type is designed for a particular task and function. Whilst Plone provides with several generic content types, they may, or may not fulfill your needs. Repurposing is a term for taking an existing content type such as a News Item and changing it to fit your needs in a better manner.

This is easier to follow with an example. Most companies issue Press Releases which are announcements of company news such as new products or deals. For this a News Item often suffices, but lets assume that we want to have some different things happen to these Press Releases:

- a different name, description and icon from News Items
- each Press Release will have custom text at the bottom describing the company, a contact and so on. Rather than have this be manually added to each Press Release, we would rather add this into one template and have all Press Releases show this information.

We will now go through each of the steps to complete this.

Creating a new content type from an existing type

In the ZMI access `portal_types`. The portal_types tool lists all the content types registered with your site. This will include all the standard ones, Documents, Folders and of course News Items. − Using the `Add` menu in the top right, select `Factory−based Type Information`

- Enter an id of `Press Release` and in the `Use default type information` select `CMF Default: News Item`. Then press `Add` to complete this form.

Each content type in Plone has information about describing the content type, these are listed in the menu. Selecting the News Item means that the Press Release will have the same initial properties and configuration as the News Item.

In the portal_types list you will now see that a new content type has been created called `Press Release`. You will also note that if you access Plone as a normal user, they will now be able to add Press Release object.

Customizing this new content type

At the moment the Press Release object is not very impressive, it will be exactly the same as a News Release, albeit with a different name. In the portal_types tool select the newly created Press Release object. This will bring up this type's properties, which will notice are the ones from the News Item. For example, the description reads "News Items contain ...". This is the description that appears for a Press Release and it should be changed to something more suitable, for example: "Press Releases contain news for the public".

By altering these values we can alter how the content appears.

Content type tool values

- Description -- the description that appears for this content
- Icon -- the id of the icon that is used for this content
- Product meta type -- the meta type for this content
- Product factory method -- the method that is called by the product factory to create this peice of content.
- Initial view name -- not used in Plone
- Implicitly addable -- can this content be added to a folder, unless explicitly specified otherwise.
- Filter content types -- if this object type is Folderish (e.g.: Folders) then enable this to filter the content types that can be added by users to this object.
- Allowed content types -- if this object is a folderish object and `Filter content types` is allowed, only the types of content specified in this list be added.
- Allow discussion -- set the default status for dicussion for all types of content

Customizing the templates for this content type

To complete the second of our goals we will need to alter the templates for this content type to present a different view to the user. In the portal_types tool select Press Releases and then the `Actions` tab. This presents the list of actions for each content type, you will probably recognise these actions as the same tabs that appear across the top of the page for the content.

In our case the action we want to change is the `View` action. This action is the default page that the user gets when they access this content. Each action points to a Page Template that is called when the action is called. Because we based this content type on a `News Item` the action of the view will point at the `newsitem_view`. Searching through the plone skins you will find that newsitem_view actually relates to the object `portal_skins/plone_content/newsitem_view`.

To alter the view we will make new page, the easiest way to do this is to select the object in the ZMI and click on the `customise` button. This will create a copy of the object in the custom folder, you can access this at `portal_skins/custom`. You will note however that the name of this object is still the same as the one specified in the action. To distinguish between the objects we will rename this object from `newsitem_view` to `pressrelease_view`. Return to `portal_types/Press Releases/manage_editActionsForms` and find the `View` action and change it from `newsitem_view` to `pressrelease_view`. The view action for a Press Release should now point at the new page template.

All we have to do now is alter the view so that we have our companies blurb at the bottom. This is a simple matter of altering `pressrelease_view` so that the required HTML is added to the page. For example after:

```
<p tal:condition="not: len_text"
   i18n:translate="no_body_text">
This item does not have any body text, click the edit tab to change it.
</p>
```

You might want to add:

```
<h3>About My Company</h3>
<p>Blah....</p>
```

This now enables us to complete the requirement of showing some extra content at the bottom of each page.

Overview

Repurposing content is a simple way of altering content in your Plone so that simple changes can be made to content. Whilst it allows simple changes to be made, it doesn't allow complicated changes beyond the scope of the original content type.

Creating a new Zope Product

Of all the methods of creating content types, creating a new Zope product is by far and away the most powerful. However it can be the most complicated to understand and requires knowledge of Python. If you are an experienced programmer or wish to get more familiar with the internals of Plone, this is the option for you. The limits of a writing a product are really your imagination.

However because this subject is so deep, we cannot cover it all here, instead we will make some assumptions.

- You understand Python.
- You know how to make a basic Zope product.

...

Using Archetypes

Archetypes is a framework for the development of new Content Types in Zope/CMF/Plone. Schema driven automatic form generation, simple integration with rich content types, and a lower entry bar to the complex requirements Zope places on new content objects.

Archetypes (formerly known as CMFTypes) is a Zope Product which simplifies the creation of new content types under Zope 2 and the Content Management Framework (CMF). A content type might be a document, event, image, or any other content object that is made available for users/content creators to add to a Zope content management site. Most content management projects involve introducing new types of content, which can be both time consuming and complicated. Archetypes provides a simple, extensible framework that can ease both the development and maintenance costs of CMF content types while reducing the learning curve for the simpler cases.

The creation of a new Type using Archetypes involves a main text file (python) that defines the fields and other objects within your type, their properties, and their behavior. Archetypes uses this information to auto generate on demand all forms and pages needed to add, edit, and view your types data. When you have written this file, you then have a product that you would install just like any other CMF/Plone product.

More about Archetypes, it's Developers Guide and other helpful information is located at http://plone.org/documentation/archetypes/

Plone's forms and navigation

Credit: Geoff Davis

Plone has a form and navigation system to help make creating and navigation forms easier.

Portal form

The `portal_form` tool provides validation and navigation services for forms. The easiest way to describe this is to see how portal_form works, for example the process of editing a Link object.

The URL for editing a link is `../myLink/portal_form/link_edit_form`. The HTML action of the editing form is itself, `../myLink/portal_form/link_edit_form`. The 'portal_form tool intercepts the url's traversal and checks for newly submitted values. If the portal form tool sees new values, portal_form finds a set of validators for the form in the `portal_properties/form_properties sheet` and invokes them on the submitted values. The chain of validators returns either `success` or `failure`. Portal_form looks in `portal_properties/navigation_properties` to see what should happen next, and it hands off to the appropriate destination. Because portal_form handles the invocation of the validation and the navigation, all your form processing code has to do is to update your object. The end result: simple, modular validators and simple form processing code, all of which can be reconfigured using the property sheets in portal_properties.

Here's what you need to do to make portal_form work for your forms:

- In your form: Change the action of your form so that it submits to itself (set the action to request/URL `−−` see `link_edit_form`). Add a hidden variable called form_submitted and set its value to template/id. Portal_form tests REQUEST.form_submitted to determine whether a form has been submitted or not.

- Register a set of validators for your form. Registration takes place via a call to portal_form's setValidators() method. The first argument is the name of the form, and the second is a list of validators. Or you can also register validators manually through portal_properties/form_properties. To set them automatically:

```
                portal_form.setValidators('link_edit_form', ['validate_id', 'validate_link
```

- The validators in the list are called in order. In the example above, validate_id is a generic validator used by most forms to check an object's id, and validate_link_edit checks properties specific to the Link class.
- Write a validator for your form. Your validator should return (1) a status (usually either `success` or `failure`), (2) a set of error messages in a dictionary, and (3) a dictionary of properties that are passed along to the next navigation state. If your validator is part of a chain, you may want to access the status / errors returned by the previous validator in the chain. These values are available in the REQUEST as 'REQUEST['validation_status']' and 'REQUEST['errors']'.
- One common application of the properties dictionary is to pass along a portal status message by setting the key `portal_status_message`. The navigation tool will pass these properties along to the next navigation state, either in the REQUEST (when the next state is a page template or a python script) or as query parameters (when the next state is a URL or action). Note that if you invoke a script after validation, you will need to have the script get your portal_status_message out of the REQUEST and pass it along to the next navigation state.
- Write a form handler. The handler should modify your object and its metadata from the REQUEST and then return a status code. Any additional code for invoking the validator and for performing navigation is now unneeded and should be deleted. Your form handler should return a tuple consisting of a status (usually either `success` or `failure`), the context that should be used by the next navigation state (returning a context allows objects to be instantiated through portal_factory -- see below), and a dictionary of properties that will be passed along to the next navigation state.
- Set up your form's desired navigation in `portal_navigation` (see below)

Portal navigation

The portal_navigation tool is part of a controller for handling navigationin forms. Navigation is completely configurable through the navigation_properties sheet in portal_properties. The format of a navigation property is as follows:

```
[type].[action].[status] = [next thing to do]
```

For example, the navigation for editing a link looks like so:

```
link.link_edit_form.failure = link_edit_form<br/>
link.link_edit_form.success = script:link_edit<br/>
link.link_edit.success = action:view
```

The navigation tool implments the state controller pattern from Design Patterns, and the navigation_properties sheet is a state transition table. The left hand side of the navigation properties above indicates the current state, and the right hand side indicates the transition.

For example:

```
link.link_edit_form.failure = link_edit_form
```

means that after receiving an outcome of `failure` from validating link_edit_form on a link, the navigation tool should re−invoke the page template link_edit_form on the current context. And:

```
                link.link_edit_form.success = script:link_edit</code></p>
```

means after receiving an outcome of `success` from validating link_edit_form on a link, invoke the script link_edit on the current context. And:

```
                link.link_edit.success = action:view</code></p>
```

means after receiving an outcome of `success` from invoking link_edit on a link, perform the view action on the current link.

The transitions specified on the right hand side can be the following:

PAGE_TEMPLATE
: invoke the page template PAGE_TEMPLATE on the current context

action:ACTION
: invoke the action ACTION on the current context

script:SCRIPT
: invoke the script SCRIPT on the current context

url:URL
: redirect to the specified (absolute) URL

In addition, the transitions can incorporate data from the request via brackets. For example, `url:http://www.zope.org/?myId=[id]` will send the user to the specified URL with [id] replaced by the value of REQUEST.id.

Diagnosing Security in Plone

If you are developing on Plone and something fails the security assertions you will get a login_form. This isn't very useful in development. So here are some mechanisms to help you gain much more insight in what is going wrong.

- In the ZMI (/manage) select the cookie_authentication object and erase the value for `Auto−login page ID` which is probably set to login_form.
- Install the VerboseSecurity product
- Set the environment variable, ZOPE_SECURITY_POLICY to PYTHON (For more information on environment variables to make Zope behave differently look at the file `doc/environment.txt` inside the Zope source)
- Restart your Zope
- When a security error is raised, VerboseSecurity should give you some more helpful information.
- This is very helpful for debugging, but please note this is a performance hit, so should only be run on development sites

Editor: *Andy McKay* $Id: 8,v 1.4 2003/10/09 11:22:29 yenzenz Exp $

Previous [Chapter 7: Plone Style Guide] | Contents | Next [Chapter 9: Optimizing Plone]

Previous [Chapter 8: Extending Plone] | Contents | Next [Chapter 10: Structured Text]

Chapter 9: Maintaining and Optimizing Plone

This chapter covers ways to maintain Plone and optimize it for higher performance. Most of these tips can be combined on a site. This section is for technical users and developers.

Debug mode

Although this may sound obvious, running Plone in debug mode is a huge performance hit especially on Windows. When accessing a page in Plone, if you are running in debug mode Plone will check every single image, script and template in the skins directory to see if it has altered. Whilst affects all operating systems it is particularly bad on and Windows this can make pages ***5 to 10 times*** slower.

When Zope is running in debug mode, the Refresh product will also check each product to see if the code has changed. Again this is a performance hit. For more information on Refresh please see the Zope documentation.

Debug mode is enabled or disabled by a command line switch `−D` at start up. By default the installers do not run in debug mode. For more information see Chapter 3 of the Zope Book

Simple optimizations)

Credit: Shane Hathaway and Sidnei de Silva

Starting python with the −O option does a tad bit of optimization. There is roughly a 5−8% speed increase by optimizing generated bytecode.

Starting your Zope using −t (number of threads to use) will give some increase of performance on *very* high traffic sites. But memory consumption will also increase. It *appears* 3 threads is enough for large sites, *remember* Zope is multithreaded but most of the I/O bound processing does not block threads. Zope is also a asyncronous application like Squid.

Monitor your cache size settings (see below). If you are seeing excessive amounts of object loads increase your cache size. Use the activity graph to gauge ZODB activity. typically over 1 hour you dont want more than a thousand or so loads continuelly. i.e. on plone.org sometimes we get 5000 loads ... but this is rare.. over a 30 minute period we usually get only 10−100 object loads.

The number of bytecodes processed until the interpreter switches can be edited in ***syscheckinterval***. This system attribute is set in $ZOPE/z2.py to 500 (in recent versions of Zope) but I have a sneaking suspecion that Plone and ZPT are executing much more processing than the average Zope. This is currently at 750 on Plone.org.

Using Apache's mod_proxy Caching for Static Content

Credit: Alan Runyan, Alexander Limi, JÃ¸rg Vidar Bryne

You can dramatically increase the performance of Plone by serving content that is static from Apache. Whilst Plone uses Zope as a fast and powerful web server, it will never be as fast and as scalable as caching the content in Apache. On plone.org this done with the following Apache configuration.

Apache is compiled with:

```
./configure \
"--prefix=/usr/local/apache" \
"--with-layout=Apache" \
"--enable-module=so" \
"--enable-module=proxy"
```

The mod_gzip configuration:

```
# http://www.remotecommunications.com/apache/mod_gzip/::
# MOD_GZIP configuration (after loadmodule-part of httpd.conf)
mod_gzip_on Yes
mod_gzip_minimum_file_size  1002
mod_gzip_maximum_file_size  0
mod_gzip_maximum_inmem_size 60000
mod_gzip_item_include mime "application/x-httpd-php"
mod_gzip_item_include mime text/*
mod_gzip_item_include mime "httpd/unix-directory"
mod_gzip_dechunk Yes
mod_gzip_temp_dir "/tmp"
mod_gzip_keep_workfiles No
mod_gzip_item_include file "\.php3$"
mod_gzip_item_include file "\.txt$"
mod_gzip_item_include file "\.html$"
mod_gzip_item_exclude file "\.css$"
mod_gzip_item_exclude file "\.js$"

#Logfile format adjusted to awstats (not implemented on plone.org, though)
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"
mod_gzip: %{mod_gzip_result}n In:%{mod_gzip_input_size}n
Out:%{mod_gzip_output_size}n:%{mod_gzip_compression_ratio}npct."
combined_gzip
```

The virtual host looks like this:

```
<VirtualHost *>
# Virtual Host Monster handling
ServerName plone.org
ServerAlias www.plone.org
ServerAdmin webmaster@plone.org
ProxyPass / http://localhost:8080/VirtualHostBase/http/plone.org:80/plone.org/V
ProxyPassReverse / http://localhost:8080/VirtualHostBase/http/plone.org:80/plon

#
# gzip handler
#
mod_gzip_item_include handler proxy-server
mod_gzip_on Yes

CacheRoot "/tmp/proxy/plone.org"
CacheSize 5000
CacheGcInterval 2
CacheMaxExpire 24
CacheLastModifiedFactor 0.1
CacheDefaultExpire 1
```

```
CacheDirLength 2
#
# Expire − by request
#
 ExpiresActive On
 ExpiresByType image/gif A86400
 ExpiresByType image/png A86400
 ExpiresByType image/jpeg A86400
 ExpiresByType text/css A86400
 ExpiresByType text/javascript A86400
 ExpiresByType application/x−javascript A86400

CustomLog /usr/local/apache/logs/plone−access_log combined
ErrorLog /usr/local/apache/logs/plone−error_log
</VirtualHost>
```

Whilst this is not the only approach, it does provide

Reference: http://www.zope.org/Members/softsign/ZServer_and_Apache_mod_gzip

Increasing the ZODB Cache Size

Credit: Alexander Limi, Shane Hathaway

It was found that Zope's default database cache size is set very low, and Plone benefits hugely from increasing the number of objects cached. So try the following completely painless optimization. Do the following:

- Log into Zope's Management Interface
- Go to the `Control Panel`
- Go to `Database Management`
- Click the `Cache Parameters` tab
- In `Target number of objects in memory per cache`, it probably says 400. Change this to 8000. (don't worry, a typical Plone object is not that big, so even on lower−end computers with little RAM this should work well)
- Press the `Change` button

Now, for the record there has been a lot of inconsistency in the usefulness of this field through the different areas of Zope, and according to Shane Hathaway, it might not work on all versions of Zope. On a modestly−powered 400MHz laptop, it made a world of difference and Plone feels ***much*** faster and snappier.

Caching slots

Credit: *Alan Runyan*

Recently some tweaks to RAM Cache Manager on plone.org and people who have good connections said the speed−up was significant. This is a good overview of the ideas behind caching in Zope/CMF/Plone, caveats and how we did it on plone.org to increase performance significantly.

Note: If you are unfamiliar slots, this are covered in Chapter 5

Plone has "slots" which you can put path expressions to macros in. In the same order of these slots, the expressions are evaluated and the resulting HTML is put on the page. `left_slots` puts content in the left column and `right_slots` puts content in the right column. Very simple. A path expression looks something like: `here/calendar_slot/macros/calendarBox` (this is in the default Plone setup `right_slots`). Rendered in a PageTemplate like python:path(`here/calendar_slot/macros/calendarBox`) would return the calendar widget. You can find more about path expressions using the Zope 2.6 book.

In the ZMI if you drop down the `items to add` box you will see near the `RAM Cache Manager`. The RAM Cache Manager is a default Zope tool and more information can be found in the Zope Book − putting one of these into your root folder will enable templates to register with it so there content is cached. You can have different REQUEST variables as keys into the cached content. So for instance on plone.org we created a cache_calendar RAM Cache Manager and its REQUEST variables are:

- month
- year
- HTTP_ACCEPT_LANGUAGE

So each time the calendar_slot is rendered it checks the REQUEST for these variables and matches them to the output of the expression. This allows us to have the calendar cached in multiple langauges and with different year/month settings.

This is quite nice.. right? The cache manager keeps the baked HTML and when your object/template changes it invalidates the cache (or you can manually/programmatically invalidate it). But there are some problems. Things could never be this easy, right?

Firstly you can not cache macros. So having calls like `here/calendar_slot/macros/calendarBox` will never be picked up by the cache manager. So you have to be explicit like `here/calendar_slot`. But `portal_skins/plone_templates/ui_slots/calendar_slot` is a well formed HTML document! You can't just include that rendered. So you must customize this template so that only the div shows. Then you must go to the Properties tab of your Plone Root (what contains portal_skins and other tools) and change the path expression to `here/calendar_slot`. Make a few requests and look at the statistics of RAM Cache and you will see the Hits. And when you take out the <html> tags and other stuff (because we dont want that to come with the template when you render it) your Plone HTML will still be well formed.

Secondly `here/calendar_slot` caches the template in the context of the object, this is where Acquisition will bite you. If you render the path expression here/calendar_slot in plone.org you get the calendar_slot rendered in the context of `/`, but if you are looking at say http://plone.org/Members/runyaga/ then you will get the calendar_slot rendered in the content of /Members/runyaga which is a different entry.

So what we want is not to say `tal:replace="here/calendar_slot"` but we want to say in the Properties tab for `left_slots`, `portal/calendar_slot`. And portal should be defined in `plone_templates/main_template` as `global portal here/portal_url/getPortalObject;` and then you will always render the calendar_slot in the context of `/`.

Then for the specific setup we have on plone.org.

The navtree:

```
- cache_navtree

- AUTHENTICATED_USER

- HTTP_ACCEPT_LANGUAGE

- URL1
```

If you cache here/navigation_tree_slot you should only get entries by URL1 in the currently SELECTED Language of the Logged in User. Anonymous is the user for people not logged in.

The News box:

```
- cache_news

- HTTP_ACCEPT_LANGUAGE
```

We only need to cache the portal/news_slot by languages just so that the i18n labels are cached.

Caching Pages

Cache Managers for Plone

Plone comes with two Cache Managers:

- ♦ HTTP Accelerated Cache Manager works by adding HTTP Cache headers to objects.
- ♦ RAM Cache Manager caches objects in RAM for fast retrieval.

Both these caches are discussed further in the Zope Book section on caching

Caching 404 Error Pages

Credit: *Geoff Davis*

While setting up a client machine running Windows NT, I noticed a large number of 404 errors in the log. It turns out the client machine was getting a lot of probes for backdoors left by various worms. Each time somebody tried to access one of these bogus urls, e.g. /system.exe, the server has to serve up a 404 message. I did a little benchmarking on my laptop here at home (850 MHz PIII running XP) and found that requests for bogus pages were being served up at about 4 requests per second.

Each time you get an erroneous request, plone executes `standard_error_message.py`, which hands off to `default_error_message.pt`. It is possible to get a big speedup on generating this 404 page by creating a very simple page called `default_404_message.pt` inside the ZODB. Then cache this page using a RAM Cache Manager and caching this new 404 page. To have this page rendered when a 404 occurs, you need to add the following two lines to `skins/plone_templates/standard_error_message.py`, which parses errors in the Plone site:

```
                if error_type=='NotFound'
                    return context.default_404_message()
```

Using ab[1], I got 3.9 requests per second before the change and 37 requests per second after. Just to put this in perspective, the client machine looked like it was getting a few dozen bogus requests per minute. If I got 24 bogus requests per minute on my laptop, the 404s would end up eating about 10% of my capacity. This technique will certainly help if you having to generate a lot of 404's for some reason.

[1] ab is a benchmarking tool that comes with Apache and is a simple tool to stress test your site. More information is available here

Packing Plone

Each change that occurs to an object in your Plone site is saved in the ZODB. This allows Plone to procide the Undo capability and review or undo changes to files. However this means that all these changes will be in the ZODB and can cause the ZODB to grow.

Regularly you will need to pack the ZODB to remove these changes if the database is getting too large. To do this go to Control Panel −> Database, enter the appropriate age of objects to remove and click `Pack`. Although packing will take some time, depending upon the size of your database, Plone will continue to function.

Packing is also covered in Chapter 23 of the Zope Book

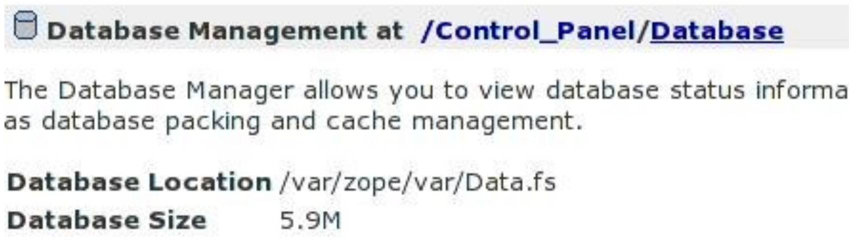
Backing up Plone

When to back up

This really depends upon the amount of changes occuring in your site. If you have a very static site a less regular backup (eg. once a week) might be more appropriate than a more frequently changing site with regular backups (eg. once a night). You should take into account packing the database when deciding a backup strategy.

What to back up

All data in your Plone site is stored in the Zope Object Database (ZODB). This ZODB is one file called `Data.fs`. To backup, you must make a copy of this file. If you are unsure where this can be found open the ZMI and go to Control Panel −> Database, you will see an image like the one below showing database size and location.


Database Management at /Control_Panel/Database

The Database Manager allows you to view database status informa
as database packing and cache management.

Database Location /var/zope/var/Data.fs
Database Size 5.9M

[1] ab is a benchmarking tool that comes with Apache and is a simple tool to stress test your site. More infor 64

In the same directory as the database you will also find the zope log files. You may, or may not wish to back these up. The Plone, CMF and Product data will need to be backed up if you have made custom changes to the filesystem code or plan to migrate to a new version of Plone, CMF or Zope. Otherwise this does not need to backed up.

If you have a customised Plone that includes relational databases or stores data on the filesystem, include this in the backup as well. This is not part of the standard setup though.

How to back up

Copy the files as mentioned in the section above. Most of the time this will just be the `Data.fs` file mentioned above. This can be performed using python, or simple shell utilities. For example the following script backs up a file then checks that md5 sums on the original and backup files are the same:

```
#! /usr/bin/env python
import sys, md5, shutil

BLOCKSIZE = 1024*1024

def hexify(s):
    return ("%02x"*len(s)) % tuple(map(ord, s))

def md5value(file):
    f = open(file, "rb")
    sum = md5.new()
    while 1:
        block = f.read(BLOCKSIZE)
        if not block:
            break
        sum.update(block)
    f.close()
    return hexify(sum.digest())

def main():
    args = sys.argv[1:]
    if len(args) != 2:
        sys.stderr.write("usage: %s src dest" % sys.argv[0])
        sys.exit(2)

    src = args[0]
    dest = args[1]
    shutil.copyfile(src, dest)
    if md5value(src) != md5value(dest):
        print "Error occured: md5 sums did not match"

if __name__ == "__main__":
    main()
```

Note: on Windows you will need to stop Plone in order to perform the backup. This is not necessary on most other platforms.

Note: it is recommended you do not rsync or scp the database over the network before making a local copy – `cp then scp`. This will ensure transactions are not lost.

Backing up is also covered in Chapter 23 of the Zope Book however at the time of writing there are a few errors in that chapter. Please note we do not recommend rsyncing the ZODB directly.

Editor: *Andy McKay* $Id: 9,v 1.4 2003/07/05 16:49:07 zopezen Exp $

Previous [Chapter 8: Extending Plone] | Contents | Next [Chapter 10: Structured Text]

Previous [Chapter 9: Optimizing Plone] | Contents | Next [Appendix A: Resources]

Chapter 10: Structured Text

Basic text formatting

- ***Italicized text***

  Enter this :

  ```
  *italics*
  ```

  to get this: *italics*
- ***Underlined text***

  Enter this :

  ```
  _underline_
  ```

  to get this: <u>underline</u>
- ***Boldfaced text***

  Enter this :

  ```
  **boldface**
  ```

  to get this: ***boldface***

Headers and paragraphs

- ***Text headers***

  Enter this:

  ```
  My header

      If you enter a single line paragraph (e.g. like the one
  above and then indent the first line of the next paragraph,
  the text in the one−line paragraph will be transformed into
  a header.

      Note that the second paragraph in the section also has
  its first line indented.

      Be sure to leave a blank line between paragraphs.  This
  paragraph's first line is indented, too.

      Subheading 1

      Use additional indentation to generate subheaders.
      Notice that the subheading's level of indenting is the
      same as the level for the previous paragraph, but that
      *this* paragraph is further indented.  It's the extra
      indenting of *this* paragraph that creates the subheading.

      Subheading 2
  ```

```
          Get smaller subheadings by indenting even more.

          Subheading 3

              Get even smaller subheadings by indenting even more.
```

To get this:

My header

If you enter a single line paragraph (e.g. like the one above and then indent the first line of the next paragraph, the text in the one−line paragraph will be transformed into a header.

Note that the second paragraph in the section also has its first line indented.

Be sure to leave a blank line between paragraphs. This paragraph's first line is indented, too.

Subheading 1

Use additional indentation to generate subheaders. Notice that the subheading's level of indenting is the same as the level for the previous paragraph, but that *this* paragraph is further indented. It's the extra indenting of *this* paragraph that creates the subheading.

Subheading 2

Get smaller subheadings by indenting even more.

Subheading 3

Get even smaller subheadings by indenting even more.

Preformatted Text

- ***Block of preformatted text***

Enter this:

```
The next block of text will be formatted exactly the way I have typed it::

    This is all preformatted.

        Your formatting will be used until...
        ... you stop indenting text.
        None of the *structured text* **commands** _work_ here.

    This is still indented.

This is no longer preformatted.
```

To get this:

The next block of text will be formatted exactly the way I have typed it:

```
This is all preformatted.

    Your formatting will be used until...
        ... you stop indenting text.
    None of the *structured text* **commands** _work_ here.

This is still indented.
```

This is no longer preformatted.

- ***Inlined preformatted text***

Enter this:

```
Some ordinary text here.  Some 'preformatted text here'.  More ordinary text.
```

to get this:

Some ordinary text here. Some `preformatted text here`. More ordinary text.

Lists

- ***Bulleted lists***

  Enter this:

  ```
  * First item

  * Second item.  Note that there is a blank line between
  each list item.

      * Make lists within lists by using extra indentation

      * Second indented item.

  * Third item in the main list.
  ```

  to get this:

  ♦ First item
  ♦ Second item. Note that there is a blank line between each list item.
  ◊ Make lists within lists by using extra indentation
  ◊ Second indented item.
  ♦ Third item in the main list.

- ***Numbered lists***

  Enter this:

  ```
  1 First item

  2 Second item.  Again, note that there is a blank line
  between each list item.
  ```

```
3 Third item in the main list.
```

to get this:

1. First item
2. Second item. Again, note that there is a blank line between each list item.
3. Third item in the main list.

Note that numbered lists cannot be embedded in bulleted lists.

- ***Definition lists***

  Enter this:

  ```
  First item -- More information about the first item.

  Second item -- More information about the second item.

  Third item -- More information about the third item.
  ```

  to get this:

  First item
  : More information about the first item.

  Second item
  : More information about the second item.

  Third item
  : More information about the third item.

Links

- ***Links***

  Enter this:

  ```
  "A link to CNN":http://www.cnn.com
  ```

  to get this:

  A link to CNN
- ***Email addresses***

  Enter this:

  ```
  "address@example.com":mailto:address@example.com
  ```

  to get this:

  address@example.com
- ***References***

  Enter this:

  ```
  I am going to refer to a footnote here [1].
  ```

```
Later in the text I will have a footnotes
section.

.. [1] My footnote.  The initial whitespace
controls indentation, then the two periods
followed by the space and the bracketed text
create the anchor.
```

to get this:

I am going to refer to a footnote here [1].

Later in the text I will have a footnotes section.

[1] My footnote. The initial whitespace controls indentation, then the two periods followed by the space and the bracketed text create the anchor.

Images

- ***Images***

  Enter this:

  ```
  <img src="logo.jpg" alt="Text shown when the browser does not load the image">
  ```

  to get this:

  You can use an arbitrary URL for the image, e.g.:

  ```
  <img src="http://www.plone.org/logo.jpg" alt="Plone logo">
  ```

Tables

- ***Tables***

Enter this:

```
|-------------------------------------|
| Fruit     | Nut         | Mammal    |
|=====================================|
| Apple     | Peanut      | Squirrel  |
|-------------------------------------|
| Orange    | Macadamia   | Woodchuck |
|-------------------------------------|
| Banana    | Walnut      | Dolphin   |
|-------------------------------------|
| This spans 2 columns!   | Cat       |
|-------------------------------------|
| Pear      | This spans 2 columns!   |
|-------------------------------------|
```

```
| This spans 3 columns!             |
|-----------------------------------|
```

to get this:

Fruit	Nut	Mammal
Apple	Peanut	Squirrel
Orange	Macadamia	Woodchuck
Banana	Walnut	Dolphin
This spans 2 columns!		Cat
Pear	This spans 2 columns!	
This spans 3 columns!		

View the Structured Text used to create this page

$Id: 10,v 1.4 2003/06/23 05:11:19 zopezen Exp $

Previous [Chapter 9: Optimizing Plone] | Contents | Next [Appendix A: Resources]

Previous [Chapter 10: Structured Text] | Contents | Next [Appendix B: FAQ and Recipes]

Appendix A: Resources

Websites

* Zope Labs is especially useful for Zope/CMF cookbook−style recipes and code snippets.
* Plone HOW−TO's usually contain up−to−date information.
* Mailing lists are the best way to communicate besides real−time chat like IRC.
* Python Cookbook is especially useful for new comers to Python

Development

* Collective is a community development initiative. People with proven Python skills can get access and contribute their modules. Many modules are not released and are only accessible using CVS.
* Archetypes is the preferred way of generating classes for use with Plone. TTWTypes are another interesting approach to constructing content objects with little to no programming.
* Zope CVS is a treasure trove full of goodies such as CMFStaging and PageDesign products. Many things from CVS are ill−documented and requires a `use the source` approach.

Learning Resources

* The Pavoda Videos are a set of videos made at the Pavoda sprint. They cover many Plone topics including: CMF Collection, CMF Member Groups, the Control Panel, Installation, and adding new content. They are available in both English, Japanesse and German.
* The Zope Book is a complete set of documents for the Zope Application Server, which Plone is based on. The Zope book will help power users or developers gain a better understanding how Plone works under the hood.

$Id: a,v 1.4 2003/06/23 04:59:52 zopezen Exp $

Previous [Chapter 10: Structured Text] | Contents | Next [Appendix B: FAQ and Recipes]

Previous [Appendix A: Resources] | Contents | Next [Appendix C: Commercial Plone]

    - ◊ Is the content accessible through WebDAV, FTP or Microsoft Web Folders?
    - ◊ Where is the CMF FAQ Located?
    - ◊ I would like to contribute to Plone, How can I get involved?
    - ◊ Why is the Plone Controller causing ZoneAlarm (or other firewall) to raise a warning?
    - ◊ What's this __getitem__ error?
- ♦ Recipes
    - ◊ What version of Plone am I running?
    - ◊ Where is Title and Description used? (mail templates and RSS feeds)
    - ◊ How can I promote a member?
    - ◊ Can I exist outside my Plone?
    - ◊ Disable member joining
    - ◊ Change the tabs for site navigation
    - ◊ Authoring Links via WebDAV
    - ◊ Make folders visible in left navbox
    - ◊ Override slots in margins
    - ◊ Fix cropping on Nav box root
    - ◊ Why does the calendar only appear in the top folder?
    - ◊ How can i get a listing of events, instead of a calendar?
    - ◊ How do I turn off the About box for public viewers?
    - ◊ I'd like my text full justified (or different font, font size, etc.)

Appendix B: Frequently Asked Questions and Recipes

Frequently Asked Questions

Credits: *Amr Malik* intial FAQ's

Static elements dont seem to cache... Why?

Alan Runyan:One of the mainstays of Zope is Acquisition. It is common to hear a guru say, "But no one expects the Spammish Acquisition!" This usually means that someone has been piggy backing on Acquisition without realizing it. One of the most common cases is when people write skins on top of Plone/CMF/Zope and do not specify absolute url's to their resources. A common mistake is in a stylesheet. If you do not absolute url the css i.e. site.css instead of http://site/site.css or /site.css your site will work − just fine. But lets say you goto http://site/about/us if you rely on relative url's it will try to fetch http://site/about/site.css instead of http://site/site.css − make your resources cache friendly on the browser; be aware of relative urls and Acquistion. Now **you too* can expect the Spammish Acquisition.

What does "Allow discussion" mean?

You can attached threaded discussions to any content object in CMF. If you allow discussion on a object there will be a `add comment` button that appears at the bottom of the content.

When I am running Zope2.6 and Plone I see this, "global name `localzone` is not defined"

Zope2.6 was shipped with a faulty version of DateTime.py. Simply replace $ZOPE/lib/python/DateTime/DateTime.py with this correct version

My Plone only stays up for a few hours or days on Linux, what could be the Problem?

Alan Runyan:Plone is running on top of the Zope Application Server. By default Zope ships with it being in debug mode, −D in the `start` script. When you run in ***debug mode*** all of the output is being logged to the console. So if you start Zope in debug mode and you log out, the next time it has something to output it will not find the controlling terminal (you had logged out) and will die. Remove the −D in the start script.

How do I relate articles to each other?

First you must activate the related slot, this is not done by default. For information on doing this please see chapter xxx.

If you share keywords in the metadata they will automatically be associated. You can see this from the related box. A good example is as follows: create a piece of content and in the metadata form add a keyword called, TopicKeyword and then save it. Create another piece of content in the system and add a keyword to it called TopicKeyword. When viewing either content you should be able to see the other content in a related box, on the left hand side.

What is Syndication?

Syndication shows you the last ten update objects on a folder in RSS 1.0 format. This format is designed to be read by other programs.

Are all of these interfaces done in XHTML/CSS/Javascript 1.3/TAL?

Yes. It is all 100% validated XHTML done with PageTemplates.

Is the content accessible through WebDAV, FTP or Microsoft Web Folders?

Yes, see Chapter xx.

Where is the CMF FAQ Located?

The CMF FAQ is located at http://cmf.zope.org/doc/FAQ

I would like to contribute to Plone, How can I get involved?

There are many ways to get involved with Plone, if you have time, initiative and want to help make CMF Plone the best content management system in the open source world. There are several areas we need expertise in documentation, developer, testing and user interface. Send an email to the plone users mailing list saying you would like to help and the area you can help and someone will contact you.

Why is the Plone Controller causing ZoneAlarm (or other firewall) to raise a warning?

The Plone Controller on Windows needs to check the status of Plone. The easiest and most reliable way to do this is to connect to the server by a HTTP connection. It does not connect to anything other than your local Plone instance and allowing the connection should cause no security issues.

What's this __getitem__ error?

Jean Jordann: I had a perfectly functioning Zope with a Plone site or two. I was playing around a bit, clicking here and there, trying different skins and products. However at one point I got an error "__getitem__" when viewing the site.

If you installed a product (such as ZopeZen) which creates a new skin (or if you created a new skin yourself) and you switched to the new skin, your browser got a cookie called `plone_skin` corresponding to this skin. Now you delete the skin, or the product supplying the skin, but you don't close your browser. If you now try to view your plone sites, you'll get the traceback below:

```
Traceback (innermost last):
Module ZPublisher.Publish, line 98, in publish
Module ZPublisher.mapply, line 88, in mapply
Module ZPublisher.Publish, line 39, in call_object
Module Products.CMFCore.PortalContent, line 113, in __call__
Module Products.CMFCore.utils, line 156, in _getViewFor
Module OFS.Traversable, line 158, in restrictedTraverse
Module OFS.Traversable, line 143, in unrestrictedTraverse
- __traceback_info__: ([], document_view)
AttributeError: __getitem__
```

Delete the relevant cookie for your browser.

Recipes

Recipes are a series of useful tips that people have found using Plone.

Credits: *Paul Everitt*

What version of Plone am I running?

If you ever want to report a bug or ask for help, you'll find that the Plone community is friendly, eager, and insistent on finding out the versions of the software you are running.

To find out the version of Plone, go to the Control Panel in your Zope site. If your Zope site is running on `http://localhost:8080/`, then the Control Panel can be found by going to `http://localhost:8080/manage` and clicking on `Control_Panel` in the folder list on the left. Next, click on `Product Management` in the right frame, then click on `CMFPlone`. Click on the `Properties` tab. The version number is listed beside `version`.

To find out your version of Zope, simply click again on the `Control_Panel` icon in the folder list. You can find information there about the Zope version, the Python version, the operating system, and more.

Where is Title and Description used? (mail templates and RSS feeds)

When adding a new Plone Site, you are asked for a Title and Description. Where is that information used?

Primarily in RSS feeds and emails sent by Plone (e.g. during member registration).

How can I promote a member?

You have someone that you want to collaborate with on your Plone site. How can you give them enough permission to participate?

The simplest way is to give them the Manager role. First, make sure they have joined the Plone site using the regular membership process. Next, if your site is at `http://localhost:8080/plone/`, then go to the Zope Management Interface (ZMI) at `http://localhost:8080/plone/manage`. In the folder list on the left, click on `acl_users`. Click on the person's username, then multiple−click (shift−click on Windows) the `Manager` role. Make sure the user still has the `Member` role before clicking `Change`.

The Manager role, though, gives them permission over everything in the site, including templates, user management, and more. If you would prefer to only give them limited permissions, you can use Zope's concept of *local roles* to give them more permission in just one part of the site.

To do this, navigate in Plone to the folder where you want the member to collaborate or take responsibility for the content. Click on `Switch to Contents view` in the navigation box. Next, click on the tab for `local roles`.

Giving them permission is a two−step process. First you find the member. Then you give them permission. In `Search by`, choose `User name` and then provide the member's name in `Search Term`. Next, click on `perform search`.

On the next screen you will see the Search results. In the `Role to assign` box, choose `Manager`. Then click on `assign selected role to user(s)`.

The local roles facility is very powerful and flexible. You can create new roles and customize the power given to those roles. However, this power confronts you with the Zope security management interface, which isn't for the faint of heart.

Can I exist outside my Plone?

No, this isn't the lead−in to a Camus novel. What if you already have a Zope manager account? Or, what if you want to have one login for many Plone sites inside a Zope process?

Yes, you can indeed keep your Zope user account and still be a manager or member inside a Plone site. Plone will even create a Member folder for that account, while re−using the `acl_users` entry outside Plone. Plone will *not* create a new entry in its own `acl_users`. Operations such as finding members in local roles, mentioned above, still work by looking in `acl_users` folder of parents.

However, you will still miss a couple of items. If you want a portrait and member information, you need to have an account inside the ports. That account can still have the same username as the account higher up in Zope. However, they are two separate entries. For instance, changing the password for one will not apply to the other.

Disable member joining

If you are running an intranet−oriented or publishing−oriented site, and not a community−oriented site, you might not want arbitrary people to become members. Fortunately it is easy to disallow joining. Visit the ZMI for your Plone site. For instance, the URL might be `http://localhost:8080/plone/manage`. Click on the `Security` tab. Find the permission labeled `Add portal member`. Uncheck the box in the column labeled `Acquire` (the first column). Next, check the box for the `Manager` and `Owner` column. Save your changes by clicking on `Save Changes`.

Change the tabs for site navigation

Plone sites have a series of tabs right under the logo. By default, these tabs are `welcome`, `members`, `news`, and `search`.

Fortunately changing these tabs is easy. First, go to the ZMI for the Plone site. For instance, the URL might be `http://localhost:8080/plone/manage`. In the folder list on the left, click on `portal_actions`.

You will now see many "actions" separated by horizontal rule lines. Look for the actions with a `Category` of `portal_tabs`. These are the actions that appear as tabs under the logo.

The easiest way to create a new action is to clone an existing one. Find the action with a `name` of `Welcome`. Based on its contents, fill in the fields at the bottom of the screen, under `Add an action`. Click on `Add`. The new action appears at the end of the list.

Next you need to put the new action in the right order by moving it up in the list. You may select (check) multiple actions, and when you click "move up" or "move down", they will all move. So, after you've added a new action, select all the actions between your new action and where you want it to be. Click "move down" and it's there!

Since the new tab is used as navigation, it probably points to a folder in the site. Make sure this folder exists and is published. If you are adding the folder now, *make sure* you add it from the Plone interface and not the ZMI. Pointing the tab at a folder will also make content under that tab display a highlighted tab to indicate the document is in that part of the site.

Fortunately, it is easier to hide or remove a tab. Find the action and uncheck the `Visible?` checkbox to hide a tab. To permanently remove it, click the checkbox, scroll down to the buttons, and click `Delete`.

Authoring Links via WebDAV

I'm a WebDAV nut, because I like to use my own tools for authoring, plus I like to create material while offline. In fact, I'm on a train right now, authoring these recipes.

Our strategy for adding link support is to support the use of the `Type` header in the input document. For instance, a new link object using structured text might look as follows:

```
Title: Test Link Title
Subject:
Publisher: No publisher
```

```
Description: Test Link Description
Contributors:
Effective_date: None
Expiration_date: None
Type: Link
Format: text/url
Language:
Rights:

http://www.plone.org/
```

Of course you might be lazy like me and only enter the `Title`, `Type`, and put the URL for the link in the body.

Zope/CMF/Plone gives builtin support for authoring documents through DAV or FTP. You just need to know the "source port" of your site. This is the port used in the `−W` argument to the Zope startup script. It is discussed in the help text for the startup script.

However, authoring other content types takes some work. To start, you need to install Sidnei da Silva's CMFCTRAddons from the Collective at `http://www.sf.net/projects/collective`. You might need to get the file from the Collective CVS.

The basic theory is that the CMF has a *content type registry* that handles the way content is created. The registry maps "predicates" (conditions in the request) to content types that can be added.

Once you have installed CMFCTRAddons, *including the External Method to put them in your Plone site*, you can change your Plone site to allow links from WebDAV or FTP. First, go to the ZMI for the Plone site. For instance, the URL might be `http://localhost:8080/plone/manage`. In the workspace (the right frame), click on `content_type_registry`.

The `Edit Predicate List` screen lets you add conditions (called "predicates"), change predicates, or re−order the precedence in which they apply.

In the `Add predicate:` box at the bottom, enter `rfc822link` as an identifier. This text could be anything you want, it just has to be unique. In the select list next to the `Add` button, choose `rfc822_headers`. Next, click `Add`. You now have a predicate with a label `rfc822link [rfc822_headers]:`. Choose `Link` for both select boxes in this predicate. Next, click `Change`.

One more tedious step. We need to move this predicate up in precedence. Otherwise our items created through DAV will still be documents. Click on the `Up` botton for the `rfc822link` predicate until it is at the top of the list.

You can now try adding links via DAV. Note that the CMF has some unusual behavior regarding DAV. For instance, line endings in structured text are always converted to Windows line endings. Authoring from a Mac can produce very unusual results.

Make folders visible in left navbox

In the Plone interface, have you ever wondered about the criteria for how a folder appears in the `navigation` box? Fortunately it is very simple. Folders in the top of your Plone site that are published are visible in the navigation tree. To hide them, visit the folder's `state` tab and select `Make private` under

`Change state`.

Override slots in margins

See the boxes in the left and right margins, such as the calendar? In Plone these are called *slots*. And like tabs, it is very easy to customize the slots in your Plone site. In fact, you can customize the slots on a *per−folder basis*.

For instance, to get rid of the calendar, start by visiting the ZMI for the Plone site. For instance, the URL might be `http://localhost:8080/plone/manage`. Find the folder where you would like to hide the calendar. In our case we will choose the root folder of the Plone site, since it already has slots set.

In the root folder you will see a ZMI tab called `Properties`. Click there and scroll down. You will see two properties called `left_slots` and `right_slots`. These are multivalue "lines" properties that define the objects, and object ordering, for the slots.

In the `right_slots` property, find the entry for `here/calendar_slot/macros/calendarBox`. It should be the second line. Remove this line and click on `Save changes` at the bottom of the page.

Now visit the home page of your Plone site. The calendar should not be visible.

As an additionaly exercise, a previous recipe removed the permission for people to join the site. But the login box is still displayed for anonymous visitors. One solution is to remove the login box from the `left_slot`.

Fix cropping on Nav box root

ev: also, on my browser, the navigation box chops off the last few chars on "Zope Europe Association", how bout you?

limi: yup

limi: it's defined in the nav box prefs, I believe

limi: cropping

Why does the calendar only appear in the top folder?

limi: because we hide the calendar in those folders

ev: ahh

limi: right_slots

ev: ahhh

limi: probably a good idea anyway

limi: so they will show up in nav

ev: in your opinion (IYO), the calendar is or isn't a useful nav tool for my site?

limi: it's a very visual complement to the event list

limi: you can see what's coming

limi: I like it

limi: and it's Eyecandy(tm)

limi: we like that

limi: breaks up the monotony

How can i get a listing of events, instead of a calendar?

limi: lookie, an event box

ev: no, pas vrai

ev: where is the zpt for it?

limi: filesystem

ev: it already existed?

limi: plone_templates/ui_slots/event_slot.pt

limi: I checked it into Plone the other day

How do I turn off the About box for public viewers?

ev: i'm not sure the stuff in the about slot (left margin) is that useful to the public

limi: so turn it off for anonymous in site_properties

ev: ok

limi: a nice big checkmark for you there :]

I'd like my text full justified (or different font, font size, etc.)

ev: the text is full justified

limi: yes

ev: whaddya think, do you like it like that?

limi: see that ploneCustom.css?

ev: first, do you like it like that, or should i change it

limi: you can add p { text−align: left; } to that

ev: does ploneCustom.css have precedence?

limi: yes

Page Editor: Andy McKay $Id: b,v 1.3 2003/09/14 16:28:51 yenzenz Exp $

Previous [Appendix A: Resources] | Contents | Next [Appendix C: Commercial Plone]

Previous [Appendix B: FAQ and Recipes] | Contents | Next [Appendix D: About this book]

Appendix C: Commercial Plone

Removed.

$Id: c,v 1.5 2003/10/24 19:38:46 zopezen Exp $

Previous [Appendix B: FAQ and Recipes] | Contents | Next [Appendix D: About this book]

Previous [Appendix C: Commercial Plone] | Contents | Next [Appendix E: Glossary]

Appendix D: About this book

This book is an experiment in Open Source publishing supporting an Open Source software product, Plone, intended both for a general audience of end users and for developers. Open Source products addressing the needs of ordinary users are pretty rare. In fact, the lack of good, comprehensible documentation keeps Open Source out of reach of many who might otherwise dive in! In this appendix, we explain how we hope to build this book and how you can contribute, to support the ongoing evolution of Plone.

The Book's Role

This isn't a book in the traditional sense. Instead of dead trees, it's a living archive −−− a growing collection of documentation available on the web, edited and organized for ease of use. The Plone book's goals are to be clear, correct, comprehensive and up to date. This means it also needs to be adaptable. While other documentation is and *should* be available, the book aims to be the one key source a user needs.

Book mechanics

It is written using structured text and stored in cvs at Sourceforge in the plone−docs project.

Documentation Process

This is the process for writing documentation. It is intended as a guideline.

- People write documentation in whatever form where ever they choose. This makes it easy for a someone to write documentation by lowering the boundary.
- They send an email to plone−docs giving the location of the documentation they have just written. This might be an long email, wiki, text file.
    - ♦ An editor reviews the documentation and if appropriate includes it in the book
    - ♦ A credit will be given to the author [2]
    - ♦ The documentation will edited as needed to fit in with the rest of the book in terms of style and content
    - ♦ The editor adds in the documentation
- The editor sends an email to plone−docs and the documentation author letting him know the change.
- When a new release occurs, if that documentation is no longer relevant or incorrect, the original author will be contacted for an update.

Scope and Users

We will concentrate primarily on end users of Plone for this book. There are many possible users and use cases, but the key is to improve the immediate usability of Plone. It could be tempting to document the CMF as well, but this should only be done when it coincides with Plone. It is too tempting to discuss Plone, CMF, Zope, DTML, ZPT and so on. This is not the aim of the book, where possible we should point to other references (for example the Zope Book) so users can find solutions.

Readers roles

These are the suggested roles are readers will have:

Site User

The person who goes in and adds content, writes docs and so on. Has very limited security.

Site Manager

The person who manages the site, fiddles with tabs, slots and so on. Does the reviewing of content and modifying of roles. Might even delve into the css and HTML.

Site Administrator

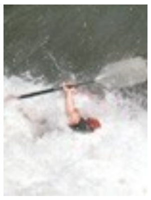
The person who installs and looks after the administration of Plone usually combined with one of the other roles. More concerned about things like Apache integration etc. Not really covered much in this book.

Developer

The person who develops new code for the site with new Products and or skins.

Editors

Andy McKay



Contact email: andy@agmweb.ca

Andy runs ZopeZen and Agmweb Consulting. As a result he spends far too long on the computer most nights. When not sat in front of computer Andy can be found walking his dog, fixing the house, surfing or his favourite past time, white water kayaking.

Copyright and Licensing

The contents of this book are licensed under the Open Publication License v1.0 without any options.

[1] How other contributors will be credited.

Page Editor: *Andy McKay* $Id: d,v 1.4 2003/07/05 16:49:07 zopezen Exp $

Previous [Appendix C: Commercial Plone] | Contents | Next [Appendix E: Glossary]

Previous [Appendix D: About this book] | Contents

Glossary of Terms

[A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Y] [Z]

[A]

Action
: In Plone terminology, "Actions" are a configurable way of providing navigational elements in a site. Some examples would be `Search`, `Browse Folder`, `Logging in` etc. See Chapter 5 of this book for more details.

Actions (Portal) ("Portal Actions")
: "Portal Actions" affect the whole site, as opposed to the "Content Types" that are more localized.

Acquisition
: Acquisition is a Zope mechanism for inheriting object properties. Zope Object hierarchy is built using Acquisition, and makes heavy use of it
  - ◊ What is Acquisition?
  - ◊ Acquisition Algebra

Anonymous Role
: This is a Standard Role in Zope Security Architecture. Anonymous role is assigned to a site visitor until they login using their zope id/password.

Archetypes
: Framework for the development of new Content Types in Zope/CMF/Plone.

Authenticated User
: An `Authenticated User` is a user who is logged in to Zope system. According to The Book of Zope, `"If No user is currently logged in, Anonymous users are considered the Authenticated User"`.

Authentication
: This is the identification process used by Zope.

[B]

Base Class
: A Base Clase or `Top Class` is a class which passes its methods, properties etc. to its subclasses. The subclasses `inherit` the properties and methods of their Base class.

Box
: `box stub`

[C]

Calendar
: The `portal_calendar` allows for a mechanism to administer what content is shown in calendar.

Catalog
: The catalog is an internal index of the content inside Plone so that it can be searched. The catalog object is accessible through the ZMI as the portal_catalog object.

CSS
: Cascading Style Sheets

Class
: A Class is the "Mold" from which objects are stamped out. Objects are instances of a class. Class can be thought of as a blueprint for an object.

Class Constructor Method

A constructor method for a class is a method which allows execution of certain actions as soon as a class instance is created, and before it is started to be used. For example, setting the standard attributes would be done in a Constructor Method.

CMF
: The Content Management Framework is a Zope addition to provide services that a Content Management system needs. See ZWACK Chapter 5 (dated)

CMFTypes
: old name for [A]rchetypes

CMS
: A Content Managment System is a system to, well manage content.

Content
: In the CMF worldview, everything is content. This applies to traditional things such as HTML pages. But it also applies to dynamic information such as posts in a threaded discussion or calendar events. It also means that images, dowloadable executables, logic in scripts, etc. are also content.

Content Type
: Content Type is the type of content allowed in a CMF/Plone instance. Plone comes with stock `Content Types`, but you can create your own content types specific to your needs and plug them into your Plone instance.

Cookie Authentication
: `cookie_authententication` aka `CookieCrumbler` allows form based login.

[D]

DTML
: Document Template Markup Language. DTML is a server side templating language used to produce dynamic peices of content, it is primarily used in conjunction with HTML.

Discussions
: The `portal_discussion` tool holds the policy regarding how the discussions work in a Plone system.

[E]

External Method
: External methods are essentially Python modules sitting on the file systems linked into Zope via the external method object that you can create from the drop down list. External Methods are more powerful than `Python Scripts` because they are not subjected to the Zope Security Architecture as rigourously as the Python Scripts may be.

[F]

Factory
: `factory stub`

Folderish Object
: A `Folderish` object in zope is an object that can contain other objects. The Folder and Plone Folder objects are examples of Folderish Objects.

Factory Type Information (FTI)
: `FTI stub`

[G]

Globbing (ZCatalog)

You can create a ZCatalog with globbing turned on, you can later on search the ZCatalog using wildcards (`*` etc.). Globbing also enables partial word searches in that ZCatalog.

[H]

HTML
: Hypertext Markup Language

[I]

Instance
: Objects are also called Instances. An Instance or Object is an instance of a given Class.

i18n
: Internationalization is preparing a program so that it can be used in multiple languages without further altering the source. For a more technical discussion, see this article. The term i18n is formed by the first and last letter of the word and the number of letters in between. Also see this HowTo by ***Alekibango***.

Interface
: See Interface based programming (EVIL EMPIRE ALERT!!)

[J]

Javascript
: Is the language that is shipped with web browsers that allow them to make web pages dynamic. A good example of Javascript is in the `Properties` tab when you add keywords.

[K]

Keywords
: in the Properties tab of content you can assign `Keywords` also known as `Subject` in Metadata terminology. This is a mechanism that allows you to relate content to each other. Keywords can be predefined in the portal_metadata tool.

[L]

Layer
: A *Skin* in Plone is an enumerated collection of *Layers*. Skins can be managed at `portal_skins/manage_propertiesForm`. All the folders in `portal_skins` can function as layers in a skin.

  Layers are not currently circumscribed in what they can do. They can change visual aspects of a Plone site (eg. `plone_styles/mozilla`), they can surface new content types in a more or less presentation−neutral way (eg. `plone_3rdParty/CMFCalendar`), or they can change/override the behaviour specified in other skins.

  Maybe you can also look at this PloneDev Archive Entry

Local Role
: Local Roles are assigned to a particular Zope user with respect to a given zope object. A local role determines that users" permissions for that object. A Local Role maybe used to restrict a users" permissions for a given object. A Local Role can also be used to give users − who may have limited global rights − expanded rights for a small subsets of objects.

Login
: This is the process you go thru when you enter your userid, password on the login screen. Same as authentication.

l10n
: Localization is the actual preparing of data for a particular language. For example Plone is i18n aware and has localization for several languages. The term l10n is formed by the first and last letter of the word and the number of letters in between.

[M]

Manager
: The `Manager` Security role is a standard role in Zope. A user with the `Manager` role has `ALL` permissions except the `Take Ownership` permission.

Meta Type
: This is a unique string for each Zope Product in the 'Available Objects" Menu in Zope Management Interface (ZMI). Product instances are created using this `Meta Type`. Each product has a unique `Meta Type`.

Metadata
: See Dublin Core Metadata Initiative

METAL
: Macro Expansion for TAL (See TAL [T])

Memberdata
: In Plone, `portal_memberdata` allows decoupling of storage of user attributes.

Migration
: `Migration` is the mostly automated process through which you upgrade your plone instance to a new release level.

[N]

Namespace
: A NameSpace contains the names of all valid variables of a given class instance (an Object) in a specific scope.

Non Folderish Objects
: These are objects that `Cannot` contain other zope objects. For example, DTML documents or Files cannot contain other Zope objects.

Navigation Properties
: (as in `CMFPlone/data/navigation_properties`)

[O]

Object
: An object is an Instance of a Class. (See Class [C])

Object DataBase (ODB)
: a system that stores a heirarchy of instances. The ZODB is an example of a Object Database. You can not query Object Databases like you can their Relational breathren. Object Oriented Database Facts gives some more insight into OODBMS.

Owner Role
: The `Owner` standard zope role has the `Take ownershhip` permission given to it by default.

Ownership (of objects)
: Users who create objects in Zope are given ownerships of those objects. Every object in Zope has an owner except perhaps the ones that are created by the Zope Install process.

OOTB
: `Out Of The Box`. Plone is an example of an OOTB web application.

[P]

Permissions
: Permissions are also called `Rights` in zope speak. They tell you what actions a user can take while in Zope. *Permissions can only be applied to Roles. You CANNOT give permissions directly to Individual Users.*

Properties
: Essentially these are the attributes of any given object. You can see a Zope object's properties by clicking on the `Properties Tab` in the ZMI when you are `Viewing` that particular object. Properties is also used on objects in the Plone interface to describe properties an object might have, such as keywords.

Plone
: *"Dude/tte, if you don't know, I'm not gonna tell you"*

Portal Type
: Portal Type is a unique string for each Content Type in Plone. In Plone each Content Type will have a unique string to identify it, although they may be based on the same Meta Type.

Python
: Python is an object oriented high level scripting language. Zope is written in Python.

[Q]

QuantumLeap
: When viewing large result sets in Plone you will notice they are presented in `pages`. You can `Leap` to any of these pages and the navigation will display `near−by` pages. This mechanism is affectionally known as QuantumLeap(ing).

[R]

Registration
: More specifically, `portal_registration` controls the site−wide policy for how users register with the system.

Request
: Each page view by a client generates a request to Plone. This incoming request is encapsulated in a request object in Zope, usually called `REQUEST` or `request`.

Response
: For each request a complimentary response is generated. This outgoing request from Zope is encapsulated in a response object, usually called `RESPONSE` or `response`.

Repurposing
: Content types can be based off other content type metadata, FTI. You can then specify unique metadata attributes for new content type such as: id, title and description.

[S]

Services
: The goal of the CMF is to unify the management of content and apply a suite of services. These services include ***cataloging***, ***workflow***, and ***syndication***. CMF+Plone provide many services to your site. There are *publicly available services* like ***search*** and ***discussion***, and *management services* such as ***workflow***.

Skin
: Think of skins as the `look & feel` part of a Plone experience. A skin contains the HTML, CSS, JavaScript, Images and all the interactions between the user and the Plone. Different skins can be apllied to the same content, meaning a content can be viewed in many different ways. Some skins provide extra features and pages over others.

Slots
: Slots are the little sections on a plone site that manifest themselves as little boxes on the left and right side of a plone instance. Consequently, they are also referred to as `left_slots` , `right_slots`. You can access slots for a Plone Instance by selecting that object, and then click on `Properties`. Along with other properties, you will see `left_slots` &`right_slots` as *Lines* . See Chapter 5 of this book for more information.

Syndication
: Syndication is the process by which a site is able to share information with other sites. Content syndication in the CMF allows you to make content available to other sites. The Syndication Tool allows site managers to control sitewide syndication of content. Syndicated content is made available in RSS format for folders where syndication has been enabled.

[T]

TAL
: Tag Attribute Language. TAL Wiki

TALES
: TAL Expression Syntax. See Also [M]ETAL TALES Wiki

Tool
: A tool is an instance of a class inside the Plone site. However, unlike other objects, there can only ever be one instance of a particular tool inside a Plone site at any one time. Some tools, such as `portal_catalog`, give administration options for the site manager.

[U]

UI (User Interface)
: The screens and way in which you interact with a software program.

[V]

View
: A presentation view displays information in a predefined structure. The actions in portal_types for instance are `views`.

[W]

Workflow
: Explain this at the business process level in content object context.

[X]

Xopus
: Xopus "is a browser based in−place wysiwyg XML editor. Xopus allows users to edit their XML data in an intuitive word processor alike way. Xopus allows common users to edit complex XML documents without knowing anything about XML without even realising they are editing XML." **Plone 1.0 is expected to have Xopus support Out of the box.**

XML
: "eXtensible Markup Language" is a standard for data interchange

[Y]

[Z]

Zope
: Zope is an open source web application server written in Python. Plone uses Zope.

ZMI
: Zope Management Interface. This generally refers to the Web Interface used for Zope Management and Administration. (when you login using http://your.zope.site:8080/manage manage at the end)

ZPT
: Zope Page Templates. See:
    - ◊ Zope Book − ZPT (Chap 5)
    - ◊ Zope Book − Advanced ZPT (Chap 8)
    - ◊ Zope Book − Appendix C (ZPT Reference)

ZPL
: Terms under which Zope is licensed.

Page Editor: Amr E. Malik $Id: e,v 1.3 2003/10/09 11:22:29 yenzenz Exp $

Previous [Appendix D: About this book] | Contents