From Technologies to Solutions

Managing and Customizing

OpenCms 6

A practical guide to creating and managing your own website with this proven Java/JSP-based content management system

Matt Butcher

PACKT PUBLISHING

Managing and Customizing OpenCms 6

A practical guide to creating and managing your own website with this proven Java/JSP-based content management system

Matt Butcher

BIRMINGHAM - MUMBAI

Managing and Customizing OpenCms 6

First published: June 2006

Production Reference: 1310506

Published by Packt Publishing Ltd.
32 Lincoln Road
Olton
Birmingham, B27 6PA, UK.

ISBN 1-904811-76-0

www.packtpub.com

Cover Image by www.visionwt.com

Credits

Author
Matt Butcher

Technical Reviewers
Ashok Hariharan
Sami Honkonen

Technical Editor
Martin Brooks

Editorial Manager
Dipali Chittar

Project Manager
Patricia Weir

Development Editor
Douglas Paterson

Indexer
Abhishek Shirodkar

Proofreader
Chris Smith

Production Coordinator
Manjiri Nadkarni

Cover Designer
Manjiri Nadkarni

About the Author

Matt Butcher is the Principal Consultant for Aleph-Null, Inc., a systems integrator located in Chicago, USA. Specializing in open-source software, Matt has worked on a wide variety of projects, including embedding Linux in set-top boxes and developing advanced search engines based on artificial intelligence and medical informatics technologies. Matt is involved in several open-source communities, including OpenCms and Gentoo Linux. In addition to his software development, Matt has worked as a freelance journalist covering areas of interest to the open-source community. Currently, Matt is working on his Ph.D. He enjoys spending his spare time with his wife and two daughters.

When I first started working with open-source developers, I thought it was all about the software. After years of work with some fantastic people from many nations and walks of life, I realize that the software is an added benefit, but it is really all about the people and the community.

Thanks to Jon Hodge for lending technical expertise and to Jane Hodge and Anna Butcher for allowing me to include pictures of their artwork. And thanks to Ashok Hariharan and Sami Honkonen for well-consider comments on drafts of this book. Thanks to Alexander Kandzior and Michael Emmerich for lending their expertise and answering questions.

Thanks also to Olli Arro, Joachim Arrasz, Paul D. Bain, Alex Epshteyn, Harald Gottlicher, Patricia Weir, Douglas Paterson, Ernesto De Santis, Alfredo MacLaughlin, and Stephan Hartmann each of whom contributed (directly or indirectly) more to this book than I could describe here.

I'd also like to thank the members of the OpenCms developers' list who have proved an invaluable resource in all of my OpenCms projects, and especially in writing this book.

And, of course, thanks to the OpenCms developers for building a great product around which an even greater community has formed.

Also, thanks to Dr. Wes Munsil and Janet Siebert for lending technical expertise as well as their encouragement.

Special thanks to Angie, Annabelle, and Claire for their encouragement.

Table of Contents

Preface

OpenCms is an open-source enterprise-grade content management system based on Java and XML technology, and is designed specifically for creating and maintaining websites. It provides a full set of tools for dealing with content creation, editorial workflow, publishing, and versioning.

The focus of this book is on using and administrating OpenCms. It has been written with users, managers, and system administrators in mind. In this book, we will examine the tools OpenCms provides for creating, managing, and publishing content with OpenCms. We will also look at installing and managing an OpenCms server, configuring OpenCms workflow, and customizing OpenCms using HTML and Java Server Pages (JSP).

Like many of the open-source content management systems, the feature-rich OpenCms is daunting on first use, but its power and flexibility reward the investment in learning to use it. This book exists to ease you into getting the most from OpenCms.

What This Book Covers

Chapter 1 gives us an introduction to OpenCms, its features, and its history.

Chapter 2 walks through the process of installing and configuring OpenCms. We look at installing OpenCms and the components it requires (such as a database and Java servlet engine).

Chapter 3 is intended to provide the user with an introduction to the OpenCms Workplace, the primary interface for managing OpenCms content. In this chapter, we cover navigating through the Workplace, understanding how the CMS works, and creating and managing basic content.

In *Chapter 4* we turn our attention to the administration tools. The Workplace has a full suite of tools that help OpenCms administrators manage the server and the content. In this section we will cover everything from user, group, and role management to backing up the database and managing the built-in search engine.

Chapter 5 is focused on the workflow. In this section, we will cover how to use the OpenCms workflow tools to streamline the process of managing collections of content. We will discuss how to configure OpenCms to facilitate work by organized groups of editors and managers.

In *Chapter 6* we look at customizing the OpenCms templates. In this section, we will examine the tools used to create custom templates for presenting a polished and cohesive website to the website's viewers. This chapter is more technical than the others, and will make use of HTML code and special JSP markup tags.

There are two appendices. The first explains the job scheduling tool in more detail. The second covers upgrading from an older version of OpenCms to the new 6.2 version.

What You Need for This Book

To use this book, you will of course need OpenCms. This is freely downloadable from `http://www.opencms.org/opencms/en/download/opencms.html`.

OpenCms has its own requirements for installation: Java 2 SDK version 1.4 or higher, Apache Tomcat 4.1 or later, and MySQL 3.23.x or higher.

You will find details of where to download these applications from in Chapter 2, but each is freely available.

For most of the book, you will need only a basic knowledge of web architecture. Most of the interaction with OpenCms is done through a web browser. Installation of OpenCms (Chapter 2) will require knowledge of installing and running server software on either Windows or Linux. The administration chapter (Chapter 3) will assume familiarity with client/server architecture, databases, and server administration. Chapter 6, which covers creating custom OpenCms templates, will require knowledge of HTML, but we will also make use of Java Server Pages (JSP) tags, which look like HTML tags, but instruct the server to perform special tasks.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

There are three styles for code. Code words in text are shown as follows: "Find the file in `/system/shared/decorations/` named `default.cfg`"

A block of code will be set as follows:

```
<p>
The path to the index file is:
<cms:link>/sites/default/index.html</cms:link>
</p>
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items will be made bold:

```
<p>
The path to the index file is:
<cms:link>/sites/default/index.html</cms:link>
</p>
```

Any command-line input and output is written as follows:

```
mysql> create table books (name char(100), author char(50));
Query OK, 0 rows affected (0.03 sec)
```

New terms and **important words** are introduced in a bold-type font. Words that you see on the screen, in menus or dialog boxes for example, appear in our text like this: "Check the box labeled Clear Cache and then press the Ok button to continue".

> Warnings or important notes appear in a box like this.

Reader Feedback

Feedback from our readers is always welcome. Let us know what you think about this book, what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply drop an email to `feedback@packtpub.com`, making sure to mention the book title in the subject of your message.

If there is a book that you need and would like to see us publish, please send us a note in the SUGGEST A TITLE form on `www.packtpub.com` or email `suggest@packtpub.com`.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer Support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the Example Code for the Book

Visit `http://www.packtpub.com/support`, and select this book from the list of titles to download any example code or extra resources for this book. The files available for download will then be displayed.

> The downloadable files contain instructions on how to use them.

Errata

Although we have taken every care to ensure the accuracy of our contents, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in text or code—we would be grateful if you would report this to us. By doing this you can save other readers from frustration, and help to improve subsequent versions of this book. If you find any errata, report them by visiting `http://www.packtpub.com/support`, selecting your book, clicking on the Submit Errata link, and entering the details of your errata. Once your errata have been verified, your submission will be accepted and the errata added to the list of existing errata. The existing errata can be viewed by selecting your title from `http://www.packtpub.com/support`.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with some aspect of the book, and we will do our best to address it.

1
Introduction to OpenCms

This book is a guide to using and administering the OpenCms content management system, and is primarily intended for editors and administrators. We will cover installing OpenCms, managing content, using the suite of administration tools, and customizing the presentation templates.

This chapter will provide a brief introduction to OpenCms, and to the concepts and technologies that it employs. More specifically, we will cover:

- The concept of a content management system (CMS)
- The features of OpenCms
- The history and community surrounding OpenCms
- A technical overview of the OpenCms project

What is a Content Management System?

It would be difficult, perhaps even impossible, to give an exhaustive definition of the expression "content management system" (CMS). There are many content management systems available today, each offering a unique set of features, but all CMS systems have one thing in common—a CMS provides a digital environment in which documents can be *stored*, *managed*, and *edited*. We will look at each of these three.

What does it mean to say that a CMS *stores* documents? Firstly, it means that the CMS writes the document to some sort of persistent memory. In this sense, a CMS may store documents in a database, a file system, or some other retrieval system.

Most content management systems will become the primary interface for saving and retrieving documents stored by them. Whereas a typical word processor writes data to the file system in a way that the file itself can be moved to a different storage mechanism, modified, or deleted by some other application, a CMS application usually requires that all retrieval and storage of a document be handled through the CMS itself.

A second function of a CMS is *management* of documents. A CMS manages content when it mediates between the storage system and the user. Management includes the ability to take a request for and deliver the correct document, but most CMSs provide more than this.

CMSs provide search engine capabilities for finding documents. They handle metadata (data about the data in the CMS), such as who authored a document and when the document was last modified. They retain version information so that changes to a document can be traced. Many CMSs (OpenCms included) have the ability to manage documents differently for different users. For example, a CMS may provide an editor or administrator with access to a whole suite of tools for manipulating its contents, while only allowing a visitor to see a certain subset of the documents.

The third function of a CMS is *editing*. A CMS provides qualified users with the ability to add, modify, and remove documents. Some CMSs use external tools (word processors, for example) to change the content of a document. Others, OpenCms included, provide editing tools within the CMS application.

In addition to these three general functions, many CMSs provide tools for workflow, publishing, and delivery.

Workflow refers to the structured or semi-structured process of creating and editing a document. This is important when more than one person works with the document. (For example, one person comes up with the idea for a document, another person writes the document, yet another person edits the document, and so on.) A CMS typically automates some or all of the process of moving a document through these stages of its development.

Publishing refers to the process of making a document available to the user. In the simplest case, publishing is nothing more than marking the document as "done" or "complete".

Delivery is closely related to publishing. It refers to the process of moving a document to the desired place. With some systems, "delivery" means sending a document to a printer. Here what we are primarily concerned is web-based delivery, where a CMS takes requests from a user on the Web and returns a published document ready for viewing in the user's web browser.

What are Content Management Systems For?

We have already seen that there are many different content management systems. But why are there so many? One answer to this question is that different CMSs have different purposes. There are three different target areas that CMSs address:

- Target medium
- Target size
- Target model

Target Medium

Typically, a content management system will be designed to handle some particular kind of content, which is usually dictated by the medium that the CMS is targeting. For example, if the target medium is a print publication, then the type of content that the CMS can handle usually consists of images, formatted text, etc.

However, if the target medium is the Web, the CMS would be organized so that it can help prepare content for online delivery and would deal primarily with content that is understood by web clients, such as HTML, web-friendly image formats (GIF, JPEG, PNG), CSS, and XML.

The target medium for OpenCms is the Web. In fact, OpenCms is a web-based content management system designed to deliver web content. This means that OpenCms itself is a web application, so that editors and administrators access it through a web browser. It also means that OpenCms is designed to store and deliver web content such as HTML, CSS, images, and XML.

Target Size

A second way of classifying a content management system is by target size—number of users, number of editors, number of simultaneous connections, etc. The most common two metrics are the number of documents stored in the CMS and the number of editors working on the CMS. The two are often related, and for the sake of brevity we will assume that they are.

Some CMS systems target small sites, where there are few people editing the content and where the number of documents will not exceed a few thousand. Systems in this range tend to emphasize ease of use rather than support for workflow, advanced editing, and robustness. Because they are optimized for small sites, they often do not scale well. Adding multiple editors can introduce management problems, and storing too much content may render the CMS unusable. But, for so long as the site is small, the CMS performs well.

CMSs targeted to medium-sized sites usually provide for numerous editors with different roles, perhaps organized into groups or units. They are built to handle tens or even hundreds of thousands of documents, and often attempt a delicate balance between ease of use and features that support complex content development cycles. Workflow, versioning, publishing support, and differentiation between types of users (administrators, content editors, managers, etc.) are almost indispensable for this type of CMS. These CMSs can often be extended to handle larger numbers of editors, but design limitations may keep them from scaling to larger numbers of documents.

OpenCms is designed to handle medium-sized sites. It can readily support dozens of editors organized into numerous groups and roles, and can support tens of thousands of documents without breaking into a sweat.

CMSs targeted to large sites usually provide for a large amount of documents and an editorial process that is complicated enough to manage content requiring custom development. Such CMSs can handle millions of documents or more (terabytes or petabytes of data). Usually, such systems will require significant software development to create custom interfaces.

Target Model

A third way of classifying a CMS is by organizational model. The organizational model describes how different users (or groups of users) will relate to the CMS. Here we will look at two of the popular models—the community model and the provider-consumer model.

In the community model, the CMS is the center around which a community forms. It is assumed that the CMS's content will come from the community. Such CMS systems emphasize site membership options. The ability to contribute and maintain content is given out more freely, and emphasis falls on providing each user with a particular space to which they may contribute content. Plone, another open-source CMS, represents such a model. Wikis also represent this trend. Although a carefully tuned OpenCms server can provide this sort of functionality, OpenCms is not primarily intended for this model.

The provider-consumer model places the CMS between one group, which provides the content, and another group, which uses the content. The first group has privileged access to the CMS, and can create and maintain the documents within the CMS. The second group, the users or visitors, has access to view the content (and sometimes limited access to generate some content such as comments), but does not enjoy access to most of the editorial components of the CMS.

OpenCms is intended primarily to fit the provider-consumer model. Editors and administrators can connect to the OpenCms Workplace and have full control over the contents of the CMS, but visitors have access only to published pages and have limited ability to contribute to the content. (By default, visitors cannot contribute any content.)

Is OpenCms the Right CMS?

OpenCms is a web-based content management system intended for medium-sized organizations and is based on a provider-consumer model. Of course, this is only a general guide, and most organizations will have requirements that this explanation does not address. Here are a few things to keep in mind when deciding whether OpenCms is the right CMS:

1. Many of OpenCms's advanced features, such as permissions and templating, can be configured so as not to be intrusive. This means smaller teams can avoid the hassle of dealing with complex functionality.
2. OpenCms uses a flexible model for publishing and workflow. If your organization needs workflow control, it is there. If not, there is no requirement to use it.
3. The OpenCms community is strong. Development is backed by Alkacon Software GmbH, which actively develops the product and provides professional support. There are lively email support lists and message boards that help you solve problems, and there are numerous consulting companies that provide OpenCms-related services.
4. OpenCms has a robust module system, which provides a straightforward method for extending OpenCms's functionality. There are many existing modules that can be obtained and installed on your OpenCms system.

No CMS is a one-size-fits-all solution, and each CMS has its strong points. We have seen how OpenCms fits into the CMS landscape and looked at some of OpenCms's strong points. In the course of this book we will look at what OpenCms does and at how it can be used.

At this point, we will change perspectives from the broad CMS landscape and how OpenCms fits into this landscape to OpenCms itself. In the remainder of this chapter, we will get an overview of OpenCms and some of its basic technical details.

An Overview of the OpenCms System

We have seen that OpenCms is a web-based CMS for delivering web content. While this may sound redundant, my purpose was to draw a distinction between two different parts of OpenCms.

OpenCms is a web-based application. Adding, managing, administering, and editing content is all done through a web application. Content editors, administrators, and managers do not need special client applications to access OpenCms—all they need is a web browser.

This web application is called the **OpenCms Workplace** and provides all the tools necessary for managing content. With it, one can navigate files, perform searches, set access controls, and manage users. New content for a web application can be created using a sophisticated graphical editor that looks and feels like a fully fledged word processor—no knowledge of HTML is required. Documents and images can be uploaded from the user's workstation into the OpenCms file system. Software developers can even write code directly in OpenCms. In short, the system is designed to move as much work as possible to the web-based environment.

But the other task of OpenCms is to make this content available to visitors looking for information. These users are not going to be accessing the OpenCms Workplace. Instead, they will be accessing documents that have been published and marked as available to visitors.

OpenCms is designed primarily to handle web-based content. It can serve published content directly to visitors. One single OpenCms server can handle multiple websites and can provide some content to all users while restricting other content to users who have accounts.

Alternatively, the content can be exported, loaded onto another web server, and served from there. In this case, OpenCms does the heavy work of generating a static, fully functional web site that can be served (as static documents) from the other web server.

To summarize, OpenCms has two major components. One is the OpenCms Workplace—a web-based application for editors, administrators, and managers. The other is the public website, which provides visitors with access to published documents.

Features of OpenCms

OpenCms is an open-source, enterprise-grade content management system designed specifically for maintaining websites. It is written in Java, and runs on Windows and on UNIX variants (including Linux). OpenCms is a web application that runs as a Java servlet, so that it requires a servlet-enabled web server, and it uses a relational database, such as MySQL or Oracle, to store most of its content.

The main features of OpenCms are the following:

- **Browser-based interface**—the only tool CMS users need, be they system administrators, authors, editors, or project managers, is a web browser. OpenCms was designed to behave in a way similar to a normal Windows application making it easier for a beginner to learn.
- **WYSIWYG editing tool**—authors and editors do not need to learn HTML. OpenCms provides a rich graphical user interface (GUI) for composing content. The editor is cross-platform and works in IE5+ as well as all Mozilla-based web browsers (Netscape, Mozilla, Firefox, etc.). Users familiar with word processing can easily learn how to use the OpenCms composer.
- **Role-based workflow**—OpenCms has a sophisticated role-based workflow for managing the entire process of content creation.
- **Permissions model**—using a combination of access control lists (ACLs), permissions, and roles, OpenCms provides methods for setting and restricting the access available to CMS users.

- **Sophisticated publishing**—OpenCms provides project-based and file-based publishing. It also performs link and dependency checking to ensure that all the necessary files are published together. Content can be exported from the CMS and deployed on other systems, or OpenCms itself can serve content to the public.
- **System administration**—OpenCms administration is browser-based. Tasks ranging from scheduling to manually flushing caches can be done using a browser.
- **Online help**—OpenCms has a complete online help system providing everything from context-sensitive help to documentation and tutorials showing how to accomplish specific tasks. Help is available in numerous languages.
- **Module-based framework**—OpenCms provides a module mechanism for adding functionality to the system. Modules ranging from online documentation (tutorials and references) to advanced content handling (calendars, news, search engines, etc.) are available as add-on software.

OpenCms is Open-Source Software

OpenCms is released under the **Lesser GNU Public License** (**LGPL**), which is an **Open Source Initiative** (**OSI**)-certified open-source license created by the Free Software Foundation. The LGPL dictates that the source code for OpenCms must be made freely available. The source code for any changes made to it must also be made freely available.

However, external components, such as modules, wrappers, or JSP files, are not restricted by the LGPL and can use different licenses (including proprietary, "closed-source" licenses). This gives the application developer the power to decide how many restrictions should be placed on the code.

> More information can be found on the following websites:
> `http://www.opencms.org/` (OpenCms)
> `http://www.gnu.org/licenses/licenses.html#LGPL` (the Free Software Foundation's LGPL License)
> `http://www.opensource.org/` (the Open Source Initiative)

The History of OpenCms

Alexander Kandzior began tinkering with content management software in 1994. By 1998, the pieces began to come together into a single content management system. In March 2000, the OpenCms Group, a loose collection of individuals working on the system, released OpenCms under an open-source license.

Under the auspices of the OpenCms Group, a number of companies and individuals contributed to the formation of OpenCms. However, by the time OpenCms reached version 5.0 in 2002, some of the original members of the group were ready to move on to new projects, and the OpenCms Group was dissolved.

Wanting to continue the development of his code, Alexander Kandzior started a new company, Alkacon Software, which took the reins of the OpenCms project. Alkacon Software now maintains the project, providing the "roadmap" and release cycle, and contributing code. To fund ongoing development of OpenCms, Alkacon provides support contracts, training, and consulting services.

The latest version of OpenCms is OpenCms 6.2. Alkacon and the OpenCms community have grown, and OpenCms has gained many new features as a result. Alexander and the Alkacon team are a visible presence in the community, and Alexander often contributes help and advice on the public OpenCms developer's list.

The OpenCms Community

Like many open-source projects, OpenCms has a lively community of developers and users. Many software developers contribute code directly. Others create add-on modules to provide additional services and features. Many more just use the product. All three of these groups participate on mailing lists and forums surrounding OpenCms.

With many active contributors, the `opencms-dev` mailing list is a great resource for help installing, configuring and developing OpenCms, and a useful source of code and new modules.

> To join the `opencms-dev` mailing list, go to:
> `http://www.opencms.org/opencms/en/development/mailinglist.html`
> The list archives are available at:
> `http://www.opencms.org/opencms/en/development/mailinglist-archive.html`

The "unofficial" OpenCms forums hosted by Synyx oHG provides an online forum for discussion. These forums have a section for HowTos and sample code generated by members of the community. `OpenCms.org` hosts completed modules on its website, and developers will often release beta modules, modifications of existing modules, and "semi-formal" documentation directly to the mailing lists or mention them on the forum site.

> The OpenCms forums are at `http://www.opencms-forum.de/`

The Purpose of This Book

This book provides a thorough guide to installing, using, and administering OpenCms. It is intended primarily for those who have used web-based applications before, for administrators and for those who will need to manage OpenCms.

It is assumed that the reader knows how to get around in a web browser and has a good working knowledge of basic computer concepts such as file systems, databases, and the Web.

The reader should be familiar with HTML, XML, and JSP. While we will deal with these at a conceptual level throughout the book, only one chapter requires the ability to work with HTML and JSP. The last chapter (Chapter 6) deals with customizing the OpenCms presentation templates. Creating or modifying these templates will require writing HTML with occasional JSP tags. One short section will involve using Java embedded in a document.

Some chapters, such as the chapter about installation (Chapter 2), will require knowledge of the command line of your operating system.

Technical Overview

OpenCms is written in Java. It makes use of industry-standard XML and uses Java DataBase Connectivity (JDBC) to store data in a relational database. Since it is built using Java, OpenCms can run on different platforms, including numerous versions of UNIX and Linux, and Windows. OpenCms is designed for scalability, and it will run on hardware ranging from laptops to a distributed collection of servers. Being a web-based application, OpenCms runs as a Java servlet inside a servlet container such as Apache Tomcat or BEA WebLogic. For data storage, it can use a number of SQL databases, including MySQL, PostgreSQL, and Oracle.

Here is a brief summary of how each of these components works (and how they all work together).

The Web Server and Java Servlets

The web server handles incoming connections. It passes connections intended for OpenCms to the servlet container for processing. The servlet container manages one or more Java servlets. Whereas a CGI script runs only for the amount of time required to process a single request, a servlet stays running until the server explicitly stops it (which usually only happens when the server shuts down). The job of the servlet container is to provide the run-time environment for the servlets.

While it is possible to run OpenCms on the command line, it is almost always run as a servlet.

The Database

OpenCms uses a database for persistent data storage. Information about file types, templates, and publishing is stored in the database—as is all the content. OpenCms supports most of the major SQL-based databases, including MySQL, PostgreSQL, and Oracle.

OpenCms uses JDBC to connect the servlet to the database during startup. Content is managed inside the database, though it can be exported from the database into static files during a publishing cycle.

Pages, Templates, and Java Server Pages

Content is stored in the database in the form of XML files. Layout information and processing code is also stored in the database, but not in the same XML document as the content. When a page is requested, the content is pulled out of the database and put into a template. Any special processing needed is then done, and the results are sent back to the requester (usually a web browser) in the form of a complete HTML file.

Templates and custom code are written in Java Server Pages, a standardized language for embedding Java processing instructions in an HTML or XML document.

Bringing it Together

A typical response to a request for a document (called `test.html`) looks something like this:

1. The web browser requests `test.html`.
2. The web server recognizes that the request must be handled by OpenCms and passes it to the OpenCms servlet request handler.
3. OpenCms retrieves information about `test.html` (including the content) from the database.
4. OpenCms puts the content for `test.html` into its template, adding all of the necessary layout elements and interpreting any JSP code that it needs to fulfill the request.
5. Once OpenCms has created the complete HTML document, the document is returned to the browser.
6. The web browser interprets the HTML, runs any scripts that it finds and displays the `test.html` page.

A Few Closing Notes

Throughout this book, there are a number of screenshots of OpenCms in action. Most of these screen shots involve a Linux operating system (Ubuntu 5.10) running the Mozilla Firefox web browser. Screenshots were taken with the GIMP (Graphical Image Manipulation Program) image tool—an Open Source alternative to proprietary photo-editing software. Of course, OpenCms can be accessed from Windows and Mac as well.

> Ubuntu Linux is a user-friendly Linux desktop system. You can learn more about Ubuntu Linux at `http://www.ubuntu.com/`. Mozilla Firefox is the new generation of the Netscape browsers. It is lightweight and extremely fast. It is hosted by the Mozilla Organization: `http://www.mozilla.org/`. Information and downloads for the GIMP can be found at `http://www.gimp.org/`. All three of these tools are free, open-source software, and can be downloaded and used free of charge.

Summary

By now, you should be familiar with the basics of OpenCms—its uses, history, and key components. The next chapter will cover the installation of OpenCms, and subsequent chapters will discuss the use and administration of OpenCms.

2
Installing OpenCms

This chapter walks through the process of installing and configuring OpenCms. In the last chapter, it was mentioned that OpenCms can run on a wide range of platforms and configurations, but we will focus on the two most common platforms, Linux/MySQL/Tomcat and Windows/MySQL/Tomcat, in this chapter. Installation on other platforms is similar enough for these instructions to suffice. We will cover:

- Prerequisites for installing OpenCms
- Deploying the `opencms.war` file
- Installing OpenCms with the Setup Wizard
- Configuring the client
- Troubleshooting the installation

Prerequisites

OpenCms will need a configured relational database and a functioning servlet engine. We will use MySQL for the database and Tomcat for the servlet engine. In this section, we will look at the process of installing and configuring these two servers.

To install these packages on Linux, you will need to be logged on as the `root` user unless your system administrator has configured the system so that you can install these packages in a different way (such as using `sudo`).

In Windows, you will need to install the software as the Administrator user. You will need to be running at least Windows 2000. (Later versions will work, too.)

Configuring the MySQL Database

MySQL (`http://www.mysql.com`) is an open-source relational database server maintained by MySQL AB. While OpenCms supports other databases, including Oracle and PostgreSQL, there are three distinct advantages to using MySQL:

- MySQL runs on Windows *and* Linux.
- It is Open Source and free of charge.
- It is the database on which OpenCms developers work.

OpenCms 6.2 can use versions of MySQL from 3.2 to 5.0. If you are using Linux, you may prefer to get the latest release from the MySQL website, but most Linux versions come with some version of MySQL, and if you are new to MySQL it is best to use that version. That way, you can rely on your Linux vendor to provide updates and fixes. If you are using Windows, it is best to use the newest stable release from the MySQL website.

> OpenCms has been officially tested on the following databases: MySQL versions 3.2.x-5.0.x, Oracle8i, 9, and 10, and PostgreSQL 7.4 and 8.0. PostgreSQL support is new in version 6 of OpenCms. Experimental support for MS SQL Server has been added in 6.2. ANSI SQL-92 compliant database servers should work as well.

MySQL on Linux

There are two ways to install MySQL on a machine running Linux. You can build the database from the source code, or you can download and install the binary code provided by MySQL. Most Linux distributions provide a MySQL package (which is often installed already), and this is usually the best version to install.

> OpenCms does not require the database to be on the same machine as the servlet engine. It simply needs access to the database over the network (via JDBC).

To install the database, consult the documentation for your Linux distribution and the documentation on the MySQL website, `http://www.mysql.com/documentation/index.html`.

In many Linux distributions network-based database connections are turned off by default, so you may need to enable the network-based server. This usually involves adding the line `port=3306` in the `[mysqld]` section of the `/etc/my.cnf` file. For example, here is the relevant portion of my `my.cnf` file:

```
[mysqld]
port=3306
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
# The file continues on...
```

The text in bold tells MySQL to listen on the network port 3306 of your server. OpenCms will connect to this port in order to communicate with the database.

You may also want to ensure that the `skip-networking` parameter is not set anywhere in the `my.cnf` configuration file, as that will disable network startup. (Debian-based Linux distributions set this parameter by default.)

Make sure the database is running before proceeding. Most Linux systems provide initialization scripts for starting and stopping services. These scripts are usually located in `/etc/init.d/`. For example, in Gentoo Linux you can check to see if the database is up with the command `/etc/init.d/mysql status`. If the server is not running, using this command with `start` instead of `status` will start the server:

```
root # /etc/init.d/mysql status
* Status: stopped
root # /etc/init.d/mysql start
* Starting mysql              [OK]
```

Once the database is running, you are ready to move on to finishing the MySQL setup.

MySQL on Windows

Although it is possible to build MySQL from source code on Windows, it is much easier to download the binary code from the MySQL site. Detailed documentation can be found at `http://dev.mysql.com/doc/`.

> Windows users may find it helpful to download the MySQL documentation in CHM or HLP help-file formats. It can then be added to the Windows help system.

Make sure that you install the MySQL server on your `C:` drive (the default location, `C:\mysql`, is fine). Register MySQL as a service by opening a command shell and running the following command:

```
shell> C:\mysql\bin\mysqld –install
```

Alternatively, if the MySQL icon appears in your system tray, you can right-click on it and choose the option to register the service. After that, you should be able to use the Windows Service utility to manage MySQL.

Before proceeding, start MySQL. You can start the service from the MySQL system tray icon, or by using the Services utility in the Windows Control Panel.

Finishing the MySQL Setup

OpenCms occasionally needs to import large files (primarily in the form of OpenCms modules) into the database. By default, MySQL does not allow large files to be inserted into the database. This is a security measure, and it makes sense in many contexts. But in this case we will need to allow MySQL to insert larger files. To do this, you will need to edit MySQL's configuration file (in `C:\mysql\my.ini` in Windows and `/etc/mysql/`

my.cnf or /etc/my.cnf in Linux). Set max_allowed_packet to allow files as large as 16 megabytes:

```
max_allowed_packet = 16M
```

Save the file and restart MySQL so that the changes take effect. (In Windows you can do this from the MySQL icon in the system tray. In Linux, you can do this from the init script: /etc/init.d/mysqld restart.)

Once MySQL is running again, set the password for the MySQL user named root. This root user has total control over the database. (The MySQL root user is not the same as the operating system's root user and can have a different password.)

```
>mysqladmin -u root password mypassword
```

> In Linux, mysqladmin is usually in the $PATH. In Windows, you may have to open a command shell and go to the C:\mysql\bin directory.

Next, connect using the mysql client:

```
>mysql -u root -p mysql
```

In the command above -u root indicates that you are connecting to the database as the root (administrative) user. You should only connect as root for administrative tasks (as we are about to do now). -p provides an interactive password prompt, and mysql at the end indicates the name of the database that we will use.

> A database server such as MySQL or Oracle can have many different databases, where each database is allocated to store some particular set of information. OpenCms will use its own database (which we will name opencms) to store its data. MySQL has its own database, named mysql, which contains information that MySQL uses to manage the other databases.

The mysql database is used for storing information about permissions. MySQL uses this information to determine who has access to which databases. Once you have connected to the database, it is time to create the opencms database and add the user accounts that OpenCms will use to connect to that database.

The MySQL command for creating a database is CREATE DATABASE [dbname], so the command to create a database named opencms is:

```
mysql> CREATE DATABASE opencms;
```

The semicolon at the end of the line indicates that the command is complete. (Without it you would get the prompt ->, which indicates that the SQL interpreter expects you to add more to the command. If this happens, you can just type ; to indicate that the command is

complete.) To make sure that the database has been successfully created, you can enter the `SHOW DATABASES` command:

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| mysql    |
| opencms  |
| test     |
+----------+
3 rows in set (0.00 sec)
```

Next we now need to create a user to access this database. We will call this user `opencms`:

```
mysql> GRANT ALL PRIVILEGES ON opencms.* TO opencms@localhost
     IDENTIFIED BY 'mypassword';
```

This statement gives permission to add, delete, and modify (`ALL PRIVILEGES`) all tables (`opencms.*`) in the `opencms` database to the user `opencms@localhost`, whose password is `mypassword`. If OpenCms is to be run on a different machine from MySQL, you will need to create a similar statement to give log-on permission to `opencms@<OpenCmsHostname>` (where `<OpenCmsHostname>` is the host name or IP of the host running OpenCms).

> Increasingly, Linux distributions are using the default domain `localhost.localdomain` instead of simply `localhost`. If this is the case with your Linux distribution, you will need to alter the example above to grant permissions to `opencms@localhost.localdomain`.

To disconnect from the database and exit, type the `quit` command at the `mysql` prompt and you will be returned to the operating system shell:

```
mysql> quit
Bye
$
```

The database is now prepared for OpenCms. It's now time to configure the servlet engine.

Configuring the Tomcat Servlet Engine

OpenCms is a web-based application. It runs as a servlet inside a servlet container. Sun Microsystems, the company behind Java, has published the servlet standard, and since OpenCms adheres to the Java servlet 2.3 standard (`http://jcp.org/en/jsr/detail?id=053`), it will run in any servlet container that fully implements this standard.

Before you can run any Java applications, you will need to install the **Java System Development Kit** (**JSDK**). OpenCms 6.2 is written to work with JSDK 1.4 and 1.5, it and will not run with earlier versions. Sun also packages a *runtime-only* version of Java

called the **Java Runtime Environment** (**JRE**). OpenCms requires the JSDK, and *will not run with just the JRE*.

> By default, Windows and Linux do not include the Sun JSDK. If Java is not already installed, you can obtain Sun's version for free from `http://java.sun.com/`. At the time of writing, neither IBM's JDK nor Blackdown's JSDK has been thoroughly tested with OpenCms.

For this book, we will use the **Jakarta-Tomcat** servlet engine (usually called simply Tomcat), which is jointly developed by the Apache Software Foundation (makers of the Apache web server) and Sun Microsystems. Like MySQL, Tomcat is open source and is the main platform used by OpenCms developers. Tomcat source files and binary files are available from `http://jakarta.apache.org/site/downloads/downloads_tomcat.html`. The binary releases are almost always suitable for use, but you may download and compile the source code if you prefer. Although OpenCms can run on the older 4.1 version of Tomcat, we will use the newer (and very stable) 5.0 version. To install Tomcat on either Windows or Linux, unzip the archive into the desired directory and set the `CATALINA_HOME` environment variable to point to that directory. For Windows, there is a Tomcat release that uses a graphical installer.

Linux Configuration

In Linux, Tomcat is generally installed into `/opt/tomcat` or `/usr/local/tomcat`. Any user on the system can run Tomcat as long as they have permission to read, write, and execute files in the `tomcat` directory.

Assuming that Tomcat is in `/opt/tomcat`, set the `CATALINA_HOME` variable by running the following command (you may want to put it in your `.bash_profile` file.):

```
export CATALINA_HOME=/opt/tomcat
```

Also, make sure that the `JAVA_HOME` environment variable is set. You can check this by running `env|grep JAVA_HOME` or `echo $JAVA_HOME`. If either of these does not return the path to the JDK, you will need to set this environment variable to point to the location of your JDK installation.

To start Tomcat, run `$CATALINA_HOME/bin/startup.sh` and to stop it, run `$CATALINA_HOME/ bin/shutdown.sh`. To streamline things a bit, I usually create a small wrapper script that looks something like this (named `tomcat.sh`):

```
#!/bin/bash
##############################################################
# Simple script to start and stop Tomcat.
# This script should be named tomcat.sh, and be executable
##############################################################
export CATALINA_HOME=/opt/tomcat
# Usually this is already set. If not, set it.
```

```
# export JAVA_HOME=/opt/sun-jdk-1.4.2.11
case "$1" in
    start)
        $CATALINA_HOME/bin/startup.sh
        ;;
    stop)
        $CATALINA_HOME/bin/shutdown.sh
        ;;
    restart)
        $CATALINA_HOME/bin/shutdown.sh
        $CATALINA_HOME/bin/startup.sh
        ;;
    *)
        echo $"Usage: $0 {start|stop|restart}"
        ;;
esac
```

To start Tomcat with this script, you may just type `./tomcat.sh start`, and to stop it, type `./tomcat.sh stop`. This script will help you avoid one common mistake—it will keep you from accidentally typing `shutdown` (which shuts down Linux) instead of `shutdown.sh` (which shuts down Tomcat).

Windows Configuration

Once you have installed Tomcat, you will need to make sure that the Windows environment variables `CATALINA_HOME` and `JAVA_HOME` have been set.

To check this, right-click on My Computer and choose Properties. Go to the Advanced tab, select Environment variables, and make sure that `CATALINA_HOME` and `JAVA_HOME` are defined. If not, you will need to create these two variables. `CATALINA_HOME` should point to the Tomcat installation (e.g. `C:\Program Files\Apache Group\Tomcat 5.0`), and `JAVA_HOME` should point to the J2SDK directory (e.g. `C:\j2se1.4.2`).

Tomcat can be started and stopped either through the Service utility in the Windows Control Panel or by clicking on the Tomcat icon in the system tray.

> Tomcat can act like a stand-alone web server. This is useful for development. Tomcat is sometimes used this way in production environments as well. However, Tomcat can also run cooperatively with another web server such as Apache or MS IIS.

Check Your Configuration

Once Tomcat is installed, you may test it by opening your browser and typing in the server's IP address followed by port 8080 (for instance, `http://10.0.1.13:8080`). `http://localhost:8080` will automatically resolve to your local host, and if you are browsing from the machine on which you are installing OpenCms (that is, if you are running the web

browser and Tomcat on the same machine), using this URL is easier. For the remainder of this book, we will use the `localhost` notation for URLs directed to OpenCms.

Some machines, whether Linux or Windows, may have firewalls or other security measures that block access to port 8080. Consult your firewall documentation for information on configuring it.

You may choose to configure Tomcat to listen on the standard HTTP port, port 80. On a production server, you should make sure that the application is available on port 80—either by configuring Tomcat, or by setting it up to work cooperatively with another web server. This will ensure that your site is available to all web surfers—even those behind restrictive firewalls.

> The OpenCms server contains documentation on configuring Tomcat to work in cooperation with Apache. This is the most common configuration for production-level servers. For testing and development, though, using Tomcat on port 8080 is the most convenient.

In previous releases of OpenCms, Tomcat had to be explicitly configured to use the ISO 8859-1 character set. This is no longer required.

Tuning the JVM

One of the easiest, and most common, ways to increase the performance of a Java application is to adjust the memory settings for the JVM. In Sun's version of the `java` command, this can be done by setting the `-Xmx` and `-Xms` flags.

Use the `CATALINA_OPTS` environment variable to set initial and maximum heap sizes. The example below sets the maximum amount of RAM allocated to Tomcat to be 512 MB, and the initial amount to 256 MB. These settings would work well on a system with one gigabyte of RAM, but you could certainly experiment with more aggressive settings.

```
CATALINA_OPTS=" -Xmx512M -Xms256M"
```

These options will control how much memory the JVM attempts to use. The higher you can set these, the better. Just remember that the database will need plenty of memory too. If the applications cause the system to start swapping to disk, performance will actually diminish. It may take some experimenting to find the right balance, but the improvements gained with carefully tuned JVM memory usage can be quite noticeable.

Installing the OpenCms WAR File

As with most open-source projects, OpenCms is available in both source and binary releases. We will use the binary version. Download the 6.2 release (`opencms_6.2.0.zip`) from the OpenCms downloads page:

`http://www.opencms.org/opencms/en/download/index.html`

OpenCms, like most Java servlets, is packaged in a **WAR** (**Web ARchive**) file. A WAR file is a special kind of JAR file (which, in turn, is a special kind of ZIP file) that contains all the files necessary for running a Java servlet-based application. You can view the contents with standard JAR or ZIP tools. For example, to get a list of files in the archive, run `jar -tf` at a command line. In Tomcat, all WAR files are in the `webapps/` directory (in `$CATALINA_HOME/` on Linux and `%CATALINA_HOME%\` on Windows).

Once you have downloaded the `opencms_6.2.0.zip` file, unzip it to a temporary directory and copy the `opencms.war` file into Tomcat's `webapps/` directory. While Tomcat will eventually detect the new WAR file automatically, it is best to manually restart Tomcat, which will force it to reload all web applications. Note that OpenCms requires that the WAR be unpacked into its own directory (`$CATALINA_HOME/webapps/opencms`). By default, Tomcat automatically unpacks a WAR into its own directory, but other servlet containers may require additional configuration to elicit this behavior.

> Initially, the `opencms/` directory and the `opencms.war` file will contain the same information. But over time, OpenCms will write new files into the `$CATALINA_HOME/webapps/opencms` directory. These files will not be present in the `opencms.war` file. When running backups, make sure that you copy the directory, not the WAR file. Otherwise, you will lose critical data. (In fact, once the `opencms/` directory has been unpacked, you can safely delete `opencms.war`.)

Running the Install Wizard

Once the `opencms.war` file is in place and Tomcat is restarted, you are ready to run the installer. The installer is web-based. In order to successfully use it, you will need to make sure of two things. First, JavaScript must be enabled. Second, your browser must allow popups from the appropriate domain (`localhost`, in our case).

> The OpenCms installation and Workplace utilize advanced JavaScript. Because JavaScript implementations vary widely, OpenCms developers decided to target only the two most popular browser flavors: **Internet Explorer** and **Mozilla** (including **Netscape** and **Firefox**). Even between the two, there are some discrepancies in functionality. The OpenCms installation and Workplace may work with other browsers, but it has not been thoroughly tested.

Open a browser and enter the following URL to the OpenCms installer:

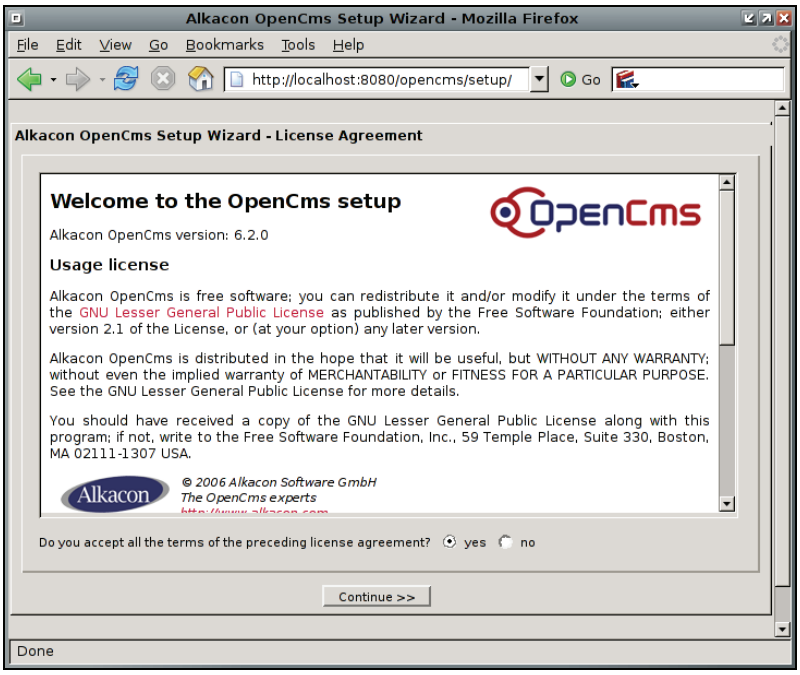
`http://localhost:8080/opencms/setup`

This will bring up the first screen of the OpenCms installation wizard. This first screen displays information about the license under which OpenCms is made available—the **GNU Lesser General Public License**. The full text of the license is available in the `license.txt` file that is included in the `opencms-6.2.0.zip` file. It is also available online at the GNU website:

`http://www.gnu.org/licenses/licenses.html#LGPL`.

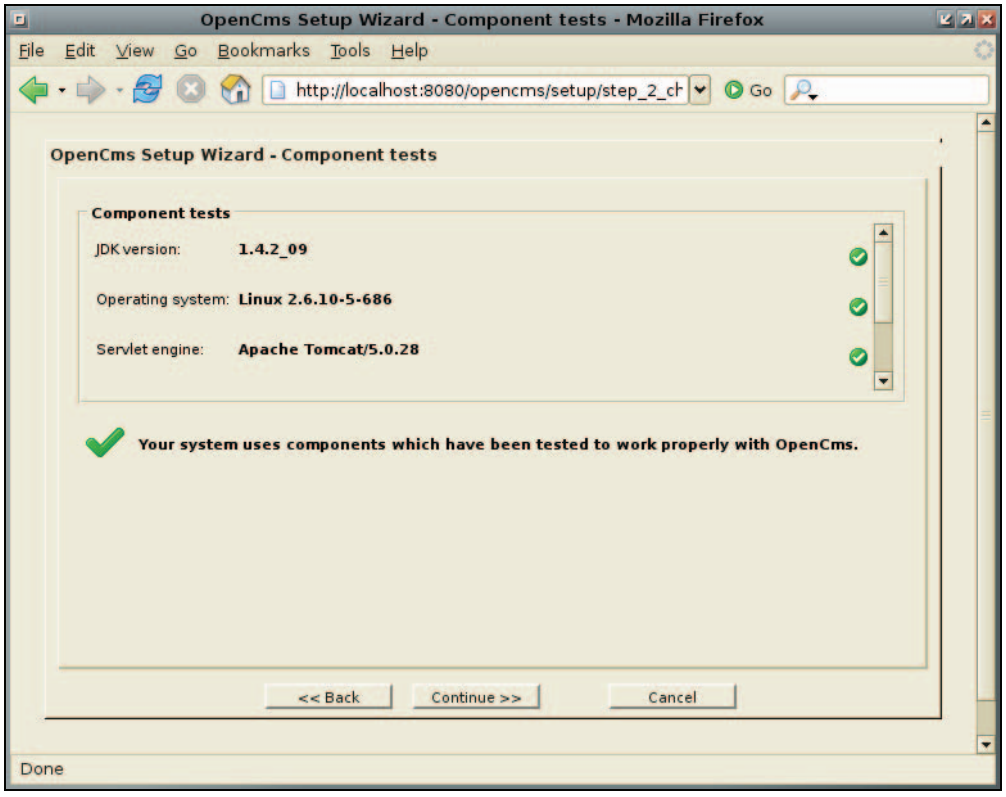
If you are not familiar with the terms of this license, you should read about it at the GNU website, as it may have some ramifications on how you develop in OpenCms.

On the first screen of the installer, you will be asked to accept the terms of the license:



You must agree to the license and select yes before the Continue button will become active.

Once you have agreed to the license, OpenCms will perform a quick test to determine whether your system already has the necessary components.

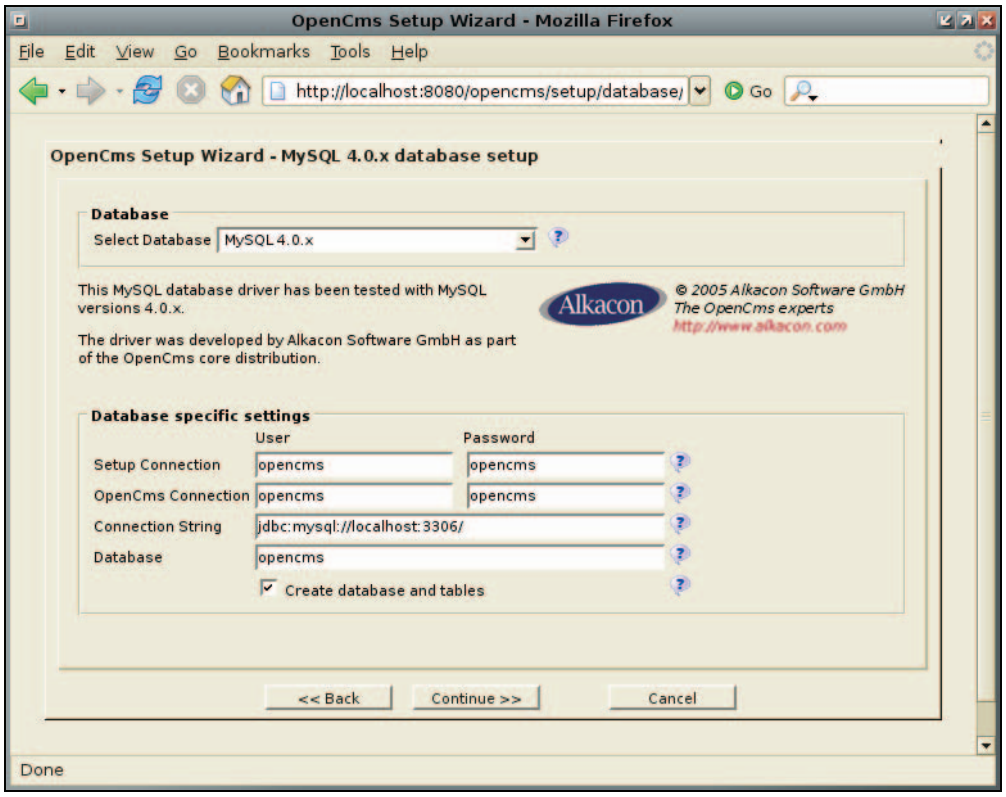


During the system check, the OpenCms installer is looking for certain key components that it requires:

- A modern JDK
- A known operating system
- A known servlet engine
- A DOM-compliant XML parser
- An unzipped version of the WAR file (See the section above entitled "Installing the OpenCms WAR File")

If you have the necessary components, the continue button at the bottom will be active, but if something is not installed correctly, you will not be allowed to continue. In such a case, OpenCms will indicate which components are not installed or are mis-configured.

Once your system has passed the system test, database installation will begin. OpenCms will create over thirty tables inside the database we set aside for it.



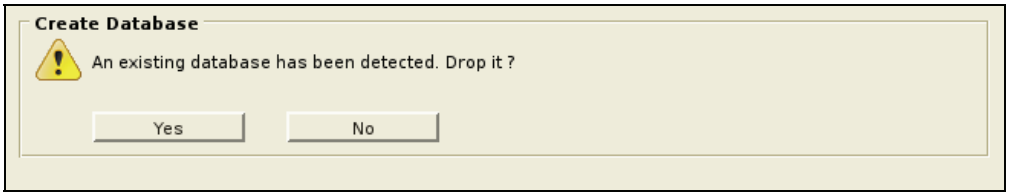
The first field on the screen, Select Database, allows you to choose the type of database OpenCms will use. Out of the box, OpenCms supports multiple versions of MySQL, Oracle, and PostgreSQL, as well as databases that are ANSI SQL-92 compliant. In 6.2, experimental MS SQL Server support has also been added. Choose the appropriate version of MySQL. The screen will refresh, and you may then complete the Database specific settings section of the screen.

In the section entitled *Finishing the MySQL Setup* we set up a MySQL user called `opencms`. We granted this user all privileges in the OpenCms database. Enter this username and password into both the Setup Connection and the OpenCms Connection sets of fields.

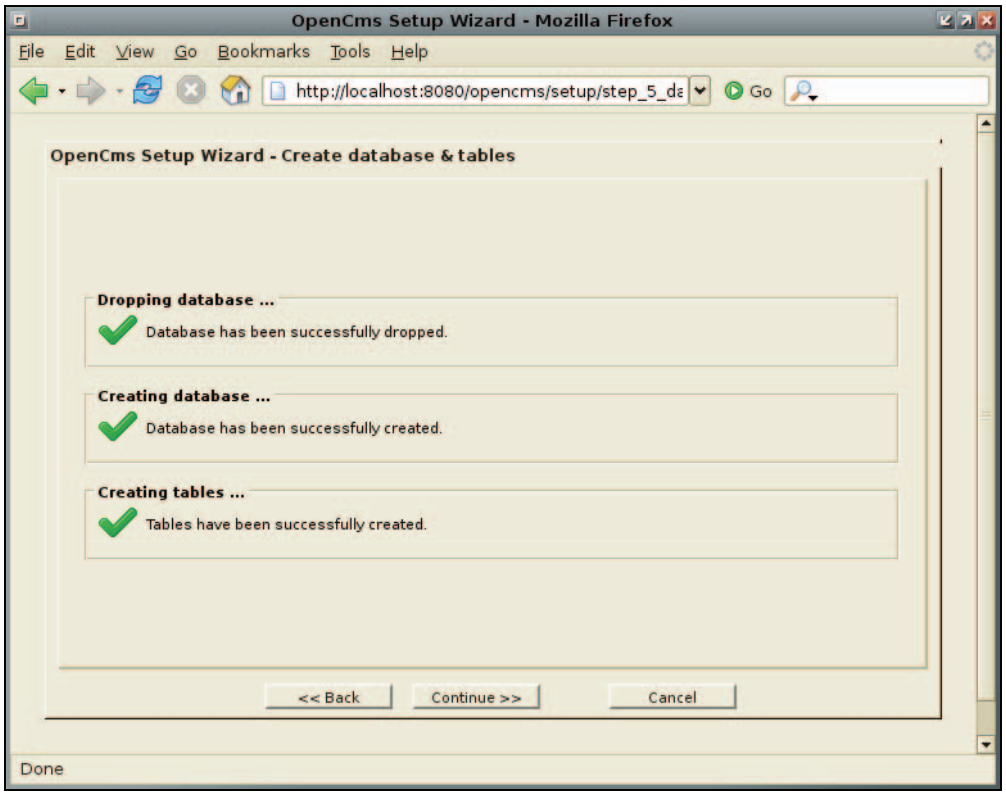
> For the most secure configuration, you may choose to restrict the opencms user to only specific database operations, such as INSERT, DELETE, and SELECT. While this does increase security, it may also prevent certain modules from installing or functioning correctly. Production systems will benefit from a more secured user, but such restrictions may slow development on pre-production servers. If you use a restricted opencms user, you will need to use some more privileged user to run the installation. Fill in the Setup Connection field with the privileged user, and OpenCms Connection with the less-privileged opencms user.

Both the Connection String and Database fields should be set correctly already. However, if you have a non-standard installation or if your database runs on a different server than the one on which Tomcat runs, you may need to change this for OpenCms to connect to the correct database.

At the bottom of the Database specific settings section of the form is a checkbox labeled Create database and tables. We have already created the database, but we have not created the tables. You will need to leave this checked. Click on Continue, and you will see a warning indicating that the opencms database already exists.

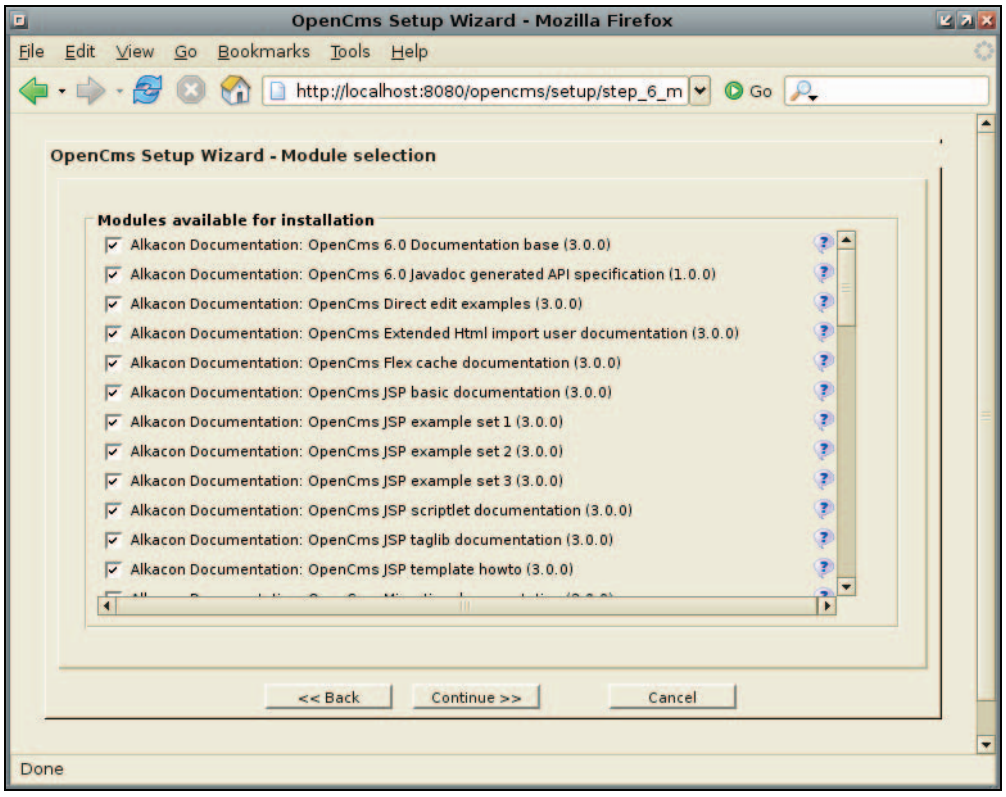


This warning prompts you to drop the database. The opencms database that we created earlier has no data in it, so we can choose Yes. (Clicking No will simply return you to the previous screen.) If your database settings were correct and the users in the Setup Connection and the OpenCms Connection fields have correct permissions, you should get a message saying that the database was successfully dropped, then created, and that the tables were all created successfully.



Click Continue to go on to the module selection screen.

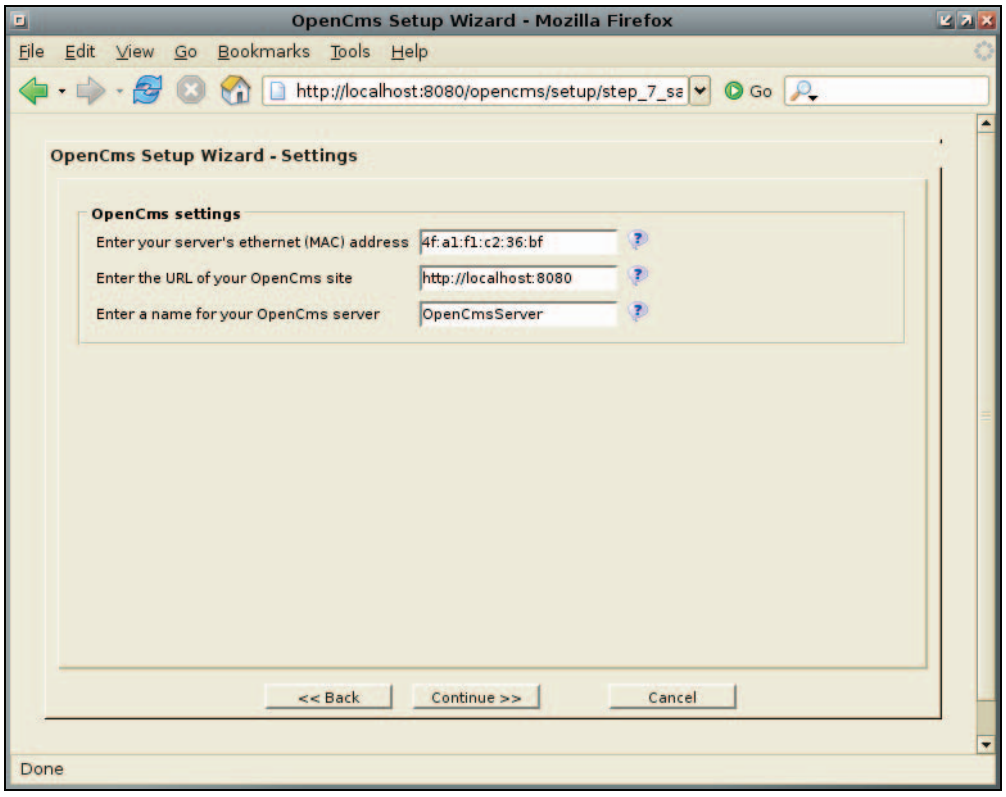
OpenCms uses a module framework for extending functionality. At the core of OpenCms are the basic content management and service capabilities, but most other functionality is contained in add-on packages called **modules**. On the module selection screen, you will have a chance to choose which of the built-in modules you would like automatically installed with OpenCms.



There are over fifty modules on the Module selection screen—all of which are selected by default. A few of them provide crucial functions to OpenCms. Some just provide tutorials, example code, how-to documents, and manuals. For the first install, it is best to leave everything selected. On a production system, you may want to uncheck the Alkacon Documentation modules and the TemplateOne demo modules.

> **A word of warning**: *Do not deselect a module unless you know what it does.* Some modules provide tools that are necessary for a fully functional OpenCms installation.

Click Continue to go on to the Settings screen.



On this screen, you will be prompted to enter or verify some basic information about OpenCms. For the first field, you are instructed to Enter your server's ethernet (MAC) address. A **MAC (Media Access Control)** address is an identifier assigned to Ethernet hardware by the manufacturer of the Ethernet card. Every Ethernet card has its own unique identifier. OpenCms uses this address to generate system-specific (and unique) identifiers (UUIDs—Universally Unique Identifiers). While it is easy for you to get your MAC address, the Java security model prevents Java applications from accessing that information, so you have to enter this information manually.

Finding your MAC Address on Linux

To find your MAC address on Linux, you can use the command-line program `ifconfig` to give you information about the configuration of your Ethernet interfaces. Most of the time, you will want information about the first Ethernet card on your system, so you will use the following command:

```
$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 4F:A1:F1:C2:36:BF
          inet addr:10.12.7.14  Bcast:10.12.7.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8871 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1359 errors:0 dropped:0 overruns:0 carrier:1
          collisions:0 txqueuelen:1000
          RX bytes:1575038 (1.5 MiB)  TX bytes:191763 (187.2 KiB)
          Interrupt:11 Base address:0xc000 Memory:cffff000-cfffffff
```

The field `HWaddr` (in the highlighted line) contains the MAC address: `4f:A1:F1:C2:36:BF`.

Finding your MAC Address on Windows

To find your MAC address on Windows, open a command prompt and type the following command: `ipconfig /all`. Look for the first line that begins `Physical Address.....`. The series of hexadecimal numbers at the end of that row is the MAC address—for example, `4f:A1:F1:C2:36:BF`. (If you have multiple Ethernet cards, you can use the MAC for any of them—they are all unique.)

If You Don't Have a MAC Address...

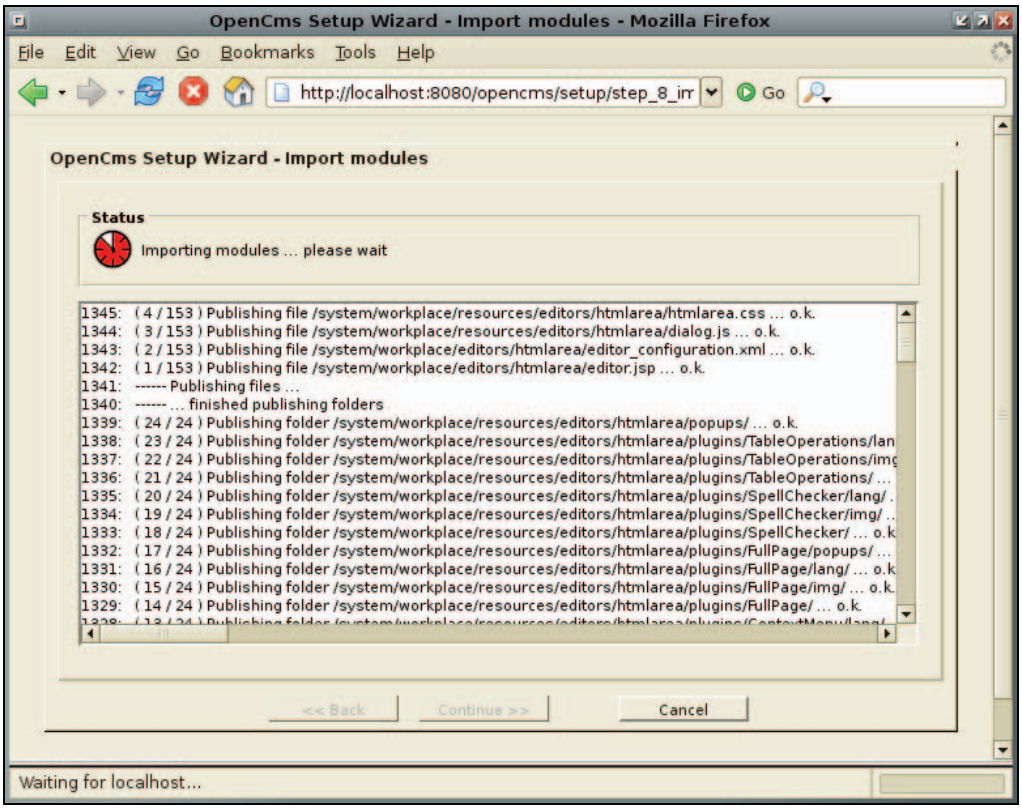
If you do not have a MAC address, you can leave the field blank and OpenCms will generate a fake one for you, but you could run into problems if your ID matches somebody else's (though the chances of this are minuscule).

Continuing Installation

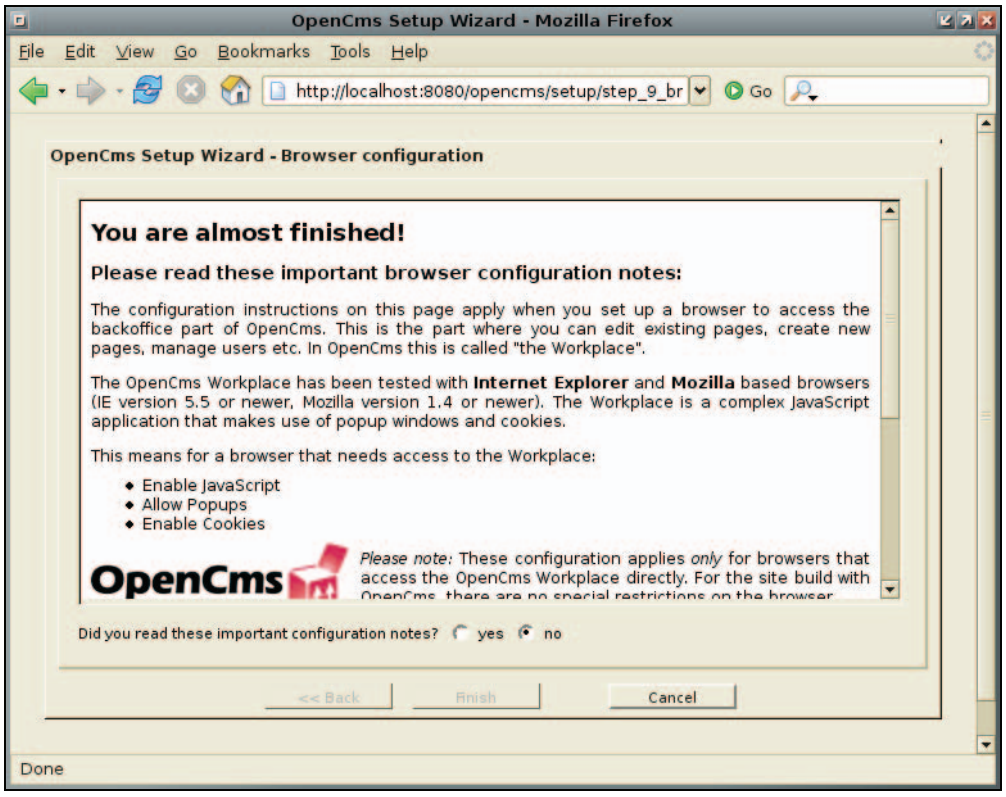
The next field instructs you to Enter the URL of your OpenCms site. It will attempt to generate a URL for you, usually with the domain name `localhost`. If your machine has a domain name, you may want to substitute that instead of `localhost` (e.g. `http://myhost.mydomain.com:8080`). The following field is labeled Enter a name for your OpenCms server. This is used primarily for tagging log messages. If you are planning on running multiple instances of OpenCms on the same server, you should set this to a value that will identify this instance of OpenCms (e.g. `MattsOpenCms`), otherwise, you can leave this at its default.

Click Continue to run the module installer.

For the next several minutes, OpenCms will install and configure all of the modules you chose a few screens ago. This is processor, disk, and memory intensive, so the entire system will likely experience slowing.

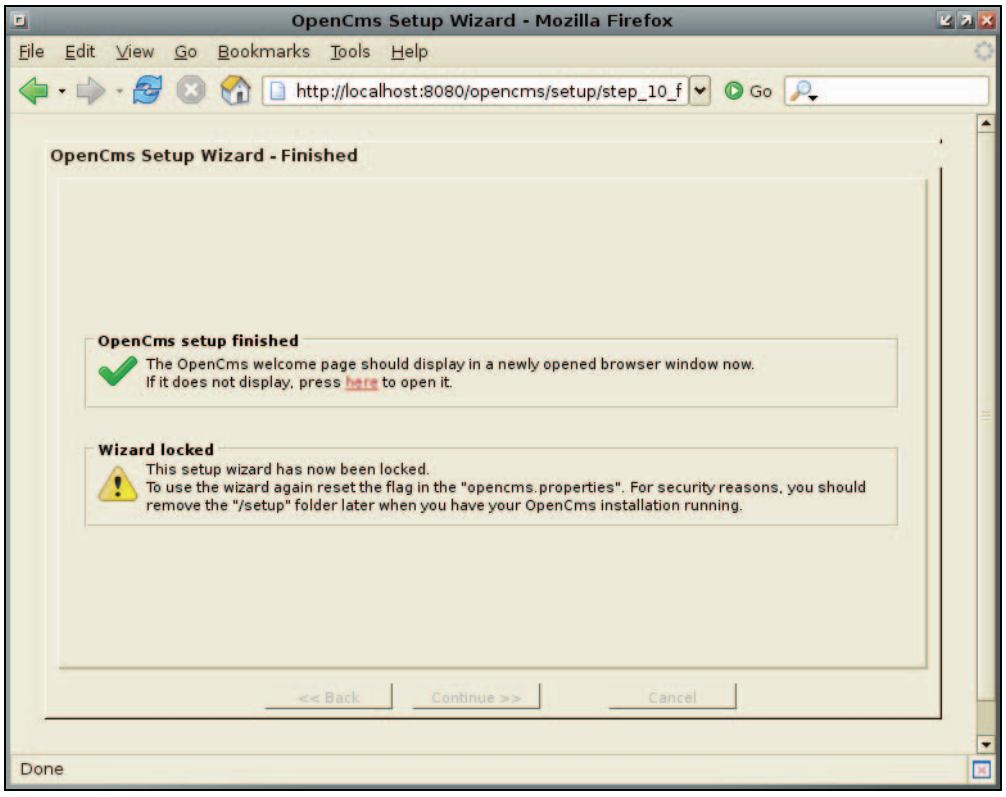


Once the modules are installed, the Continue button at the bottom of the screen will be activated, and you can click it to go on to Browser configuration.

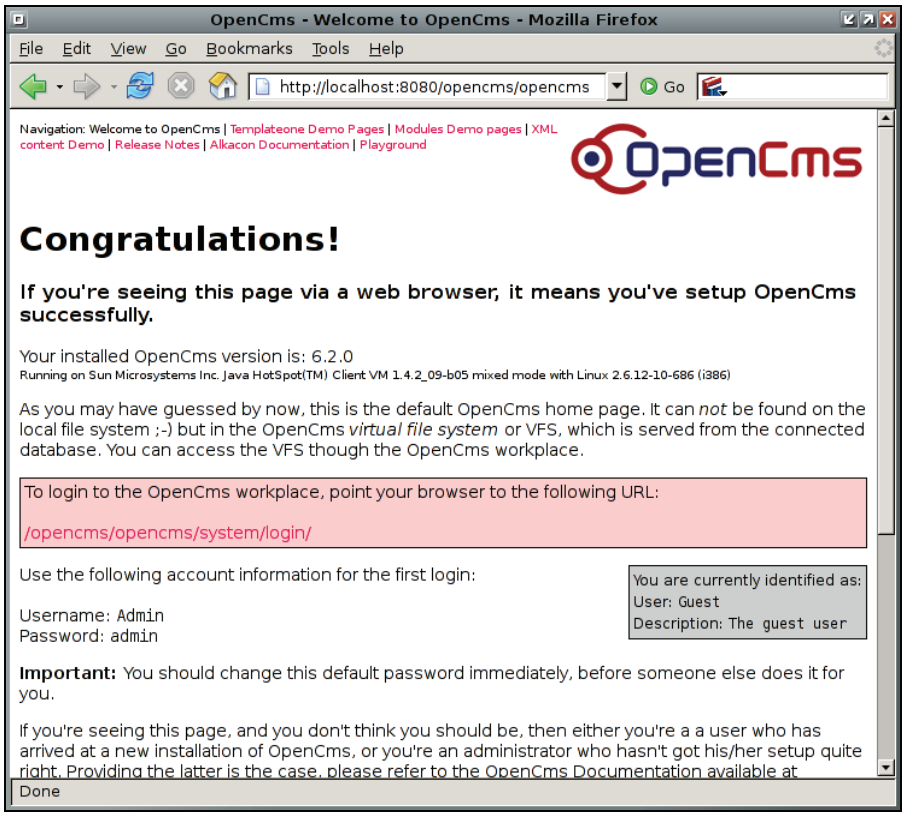


The information box describes the requirements for browsers that are going to access the online OpenCms Workplace (content authors, editors, and managers, as well as administrators will use the Workplace). OpenCms supports MS Internet Explorer 5.5 and up, as well as all browsers based on Mozilla 1.4 and up (that includes the Firefox browser). But you must enable JavaScript and cookies in your browser before you will be able to use Workplace. Additionally, you will need to allow popups from the OpenCms server.

Select the yes radio button and click Finish. This will take you to the final screen of the installation wizard.



Firstly, when this screen loads, a new window should pop up as well—one that looks like this:



If the window does not pop up, you will need to change the settings to your pop-up blocker in order to allow OpenCms to open new windows.

Secondly, you may notice the warning message that says that the installer is now locked. That means that you (or anyone else, for that matter) cannot run the installer again without first unlocking it.

> If for some reason you need to reinstall, you will need to unlock the installer. To unlock the installer, you will have to edit the file `$CATALINA_HOME/webapps/opencms/WEB-INF/config/opencms.properties` and change the value of the property `wizard.enabled` (located at the very bottom of the file) to `true`. Only enable this if you must reinstall OpenCms, since it is a major security risk to have this flag enabled. At all other times, leave the value set to `false`.

Installation is now complete. You can safely continue on to the next chapter to learn about the Workplace. The remainder of this chapter contains information on advanced configuration and installation troubleshooting.

Manually Configuring Settings

On occasion you may need to set or alter some of operating parameters of OpenCms manually. This section briefly describes the configuration files that OpenCms uses.

The OpenCms configuration files are stored in a common directory: `$CATALINA_HOME/webapps/opencms/WEB-INF/config/`. The `opencms.properties` file conforms to the Java standard for properties files. For the most part, the `opencms.properties` file contains database pool information, but it also contains the MAC address settings, the server name, and the property that determines whether the installation wizard is enabled (see above).

The rest of the files in the `config/` directory are in XML. Here is an overview of what each file contains:

- **opencms.xml**: This file contains information on which Java classes OpenCms should load when it is configuring the server (at startup).
- **opencms-system.xml**: Many of the main configuration parameters for OpenCms are stored in this file, including information on which locales are available, parameters for cache tuning, and site configuration parameters. (It is possible in OpenCms 6 to host multiple sites on one instance of OpenCms.)
- **opencms-modules.xml**: We already had our first encounter with the concept of modules during installation of OpenCms. Every module has its own configuration directives. This file is where all of the settings for all installed modules are stored. After a clean installation, my `opencms-modules.xml` file is over 2000 lines. Needless to say, it is tedious to edit this file. Fortunately, it is also rare that this file needs editing.
- **opencms-workplace.xml**: The OpenCms Workplace is the "back office" component of OpenCms. This configuration file stores settings for the Workplace.
- **opencms-vfs.xml**: OpenCms stores content in the database, but in such a way as to mimic a standard file system. Since this system looks and acts like a file system, though it is not part of the server's actual file system, it is called the Virtual File System (VFS). The `opencms-vfs.xml` file stores configuration parameters for the VFS.
- **opencms-search.xml**: OpenCms has a built-in search engine—one based on the fantastic Lucene Open Source search library. This file contains configuration parameters for OpenCms's search engine.
- **opencms-importexport.xml**: Like any good content management system, OpenCms can import and export its contents (this is useful for upgrading, backing up, or even migrating to another platform). This file makes it possible to fine tune aspects of that process.

While OpenCms should function fine without any modifications to any of these files, there are a few changes to one file that are worth making at this time.

Edit the `opencms-system.xml` file. In order to take advantage of email notifications in the Workplace, you will need to configure a few parameters in this file. Find the `<mail/>` element. It will contain several mail configuration elements:

```
<mail>
  <mailfrom>nobody@nowhere.com</mailfrom>
  <mailhost name="my.smtp.server" protocol="smtp" user="username"
    password="secure"/>
  <mailhost name="alternative.smtp.server"/>
  <mailhost name="another.alternative.smtp.server"/>
</mail>
```

Set `<mailfrom/>` to the username you want OpenCms to send from. This address will show up in the `From:` field of outgoing email messages. The `<mailhost/>` element contains information about the SMTP mail server. The first attribute of this element is `name`. Set the value to the domain name of the SMTP server that it should contact. On Linux and UNIX machines that run Sendmail, this value can usually be set to `localhost`. The next attribute is `protocol`, which should be set to `smtp` for Simple Mail Transport Protocol—the standard mail protocol used on the Internet. If your SMTP server requires authentication before a message can be sent (and this is increasingly common), you can use the `user` and `password` attributes to set the appropriate account information. If you want to set up "failover" mail hosts (so that if the first server is down, OpenCms can try another), you can simply create additional `<mailhost/>` elements and configure them accordingly. Otherwise, you should delete the extra `<mailhost/>` elements, since they will contain dummy data.

Any time you change one of these files, you will need to restart Tomcat in order for OpenCms to pick up the changes.

Installation Troubleshooting

Not all installations go flawlessly. In this section, we will look at the most common installation problems and see how to fix them. Additionally, the OpenCms website—particularly the developers' mailing-list archives—is a good resource for troubleshooting information.

Crashes During Module Installation

Occasionally, some browsers will hang (stop responding) during the lengthy module installation process. If this occurs, do not close the browser. Wait for twenty minutes (ample time for the process to complete) and then see if the browser will respond. You can often find out whether the installation is complete by checking the server's resources.

If Java and MySQL are consuming significant CPU and memory resources, then the installation is still in progress.

Often the browser will regain consciousness after the install completes. You may need to hit the Reload button if the Continue button does not become active, but usually that is not required.

If the browser will not respond, you may have to restart the browser and try again (from the beginning of the web-based installation).

Restarting Tomcat versus Reloading OpenCms

Tomcat includes an application for reloading individual servlets without restarting Tomcat. However, it is unclear whether all of the OpenCms classes are garbage-collected. Many users have reported on the `opencms-dev` list that simply reloading the OpenCms servlet causes strange errors, so it is always best to restart Tomcat rather than reload the individual servlet.

MySQL User/Password Changes

If you change the username or password for the MySQL database user that OpenCms uses, you will need to edit the following file:

```
$CATALINA_HOME/webapps/opencms/WEB-INF/config/opencms.properties
```

Change the value for the `pool.default.user` and `pool.default.password` variables, and restart Tomcat after saving your changes.

Finding More Installation Help

For more help on installation issues, try the OpenCms developers' mailing list and archives available at:

```
http://www.opencms.org/opencms/en/development/mailinglist.html.
```

Summary

At this point, OpenCms is up and running inside the Tomcat Servlet container with a MySQL database, and configuration is complete. In the next chapter, we'll take a detailed look at the OpenCms Workplace, the primary tool for content management.

3

The OpenCms Workplace

The OpenCms Workplace is the primary interface for managing the contents of your OpenCms repository. It includes tools for editing, project management, and server administration. This chapter will cover creating, editing, and publishing content. We will look at:

- How to log in to the Workplace
- The toolbar and the Preferences panel
- The explorer view
- The virtual file system (VFS)
- Creating folders and pages
- Editing content
- Publishing
- Creating galleries

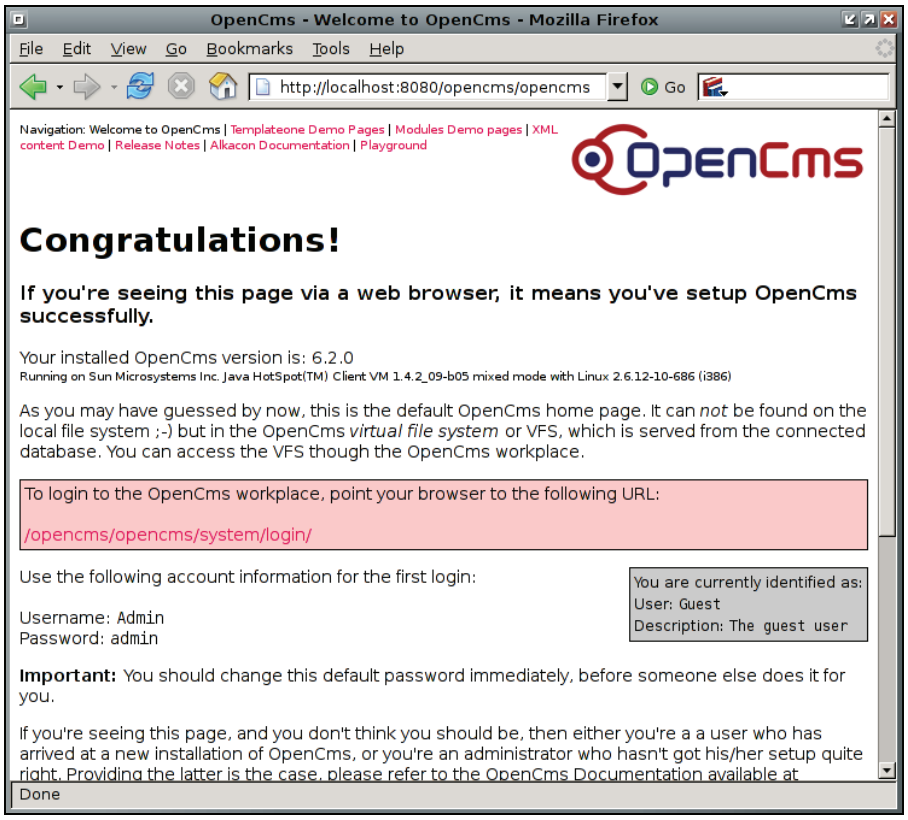
What is the OpenCms Workplace?

The Workplace is the centre of OpenCms. Editors will use the Workplace to author their documents. Project managers will use it to create workflows, structure the site, and control publication of material. The vast majority of technical administration is done through the Workplace, and the system administrator will use the Workplace to manage the caches, check logs, and install new modules. All of these tasks can be done using this one web-based interface.

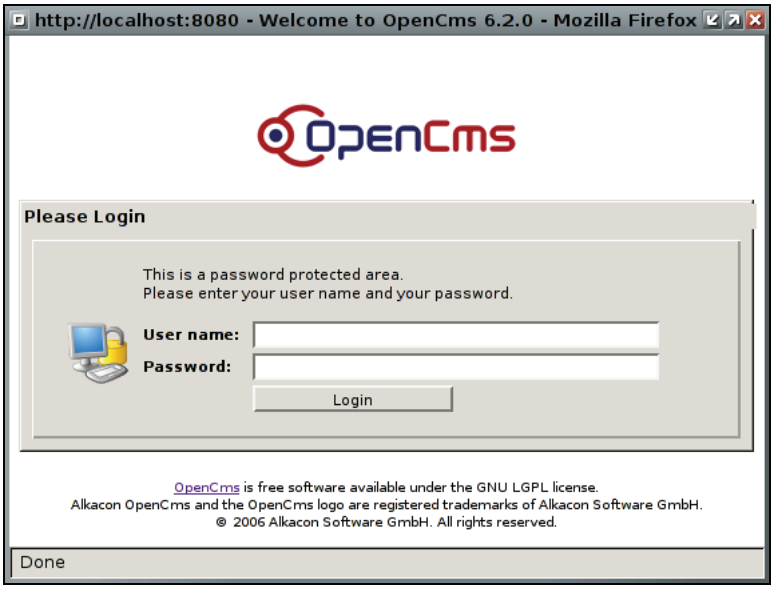
The OpenCms developers have made an effort to create an enterprise-class interface, and while it takes a while to get used to the Swiss-army-knife approach to interface design, experienced users appreciate its power and compactness.

Logging In

To begin, point your browser to `http://localhost:8080/opencms/opencms/`, the OpenCms URL. This is the main OpenCms page. You should see the default index page:



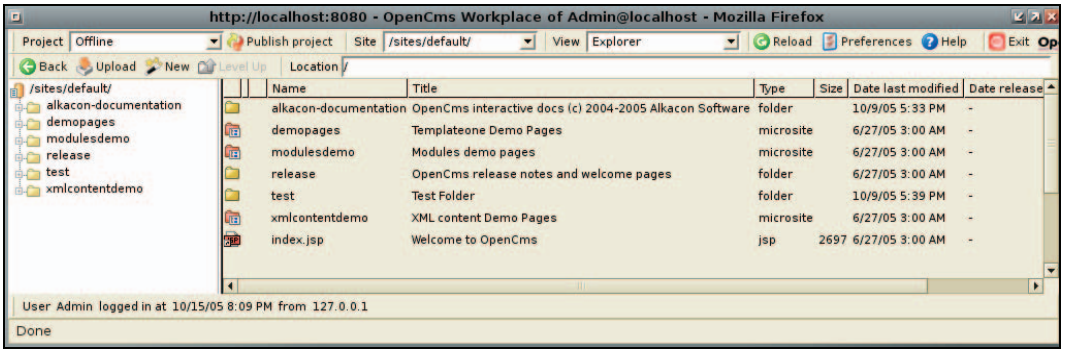
The link in the red box in the centre is the URL for the Workplace. If you click on it, a log-in window should appear.



The default login is `Admin` (note the uppercase 'A'), and the default password is `admin` (the 'a' is lowercase this time). We will be changing that shortly. After clicking on the Login button, you should see a new window pop open. If not, make sure that pop-up windows are not being blocked by your browser. Depending on your browser, you may also see a dialog asking if you want to close the calling window. If this happens, click OK.

Welcome to the Workplace.

> In the OpenCms Workplace, the normal browser navigation is hidden. This is because using the standard back and forward commands can cause unexpected results in OpenCms (as in many dynamic web applications). When using the OpenCms Workplace, it is best to use the OpenCms navigation.



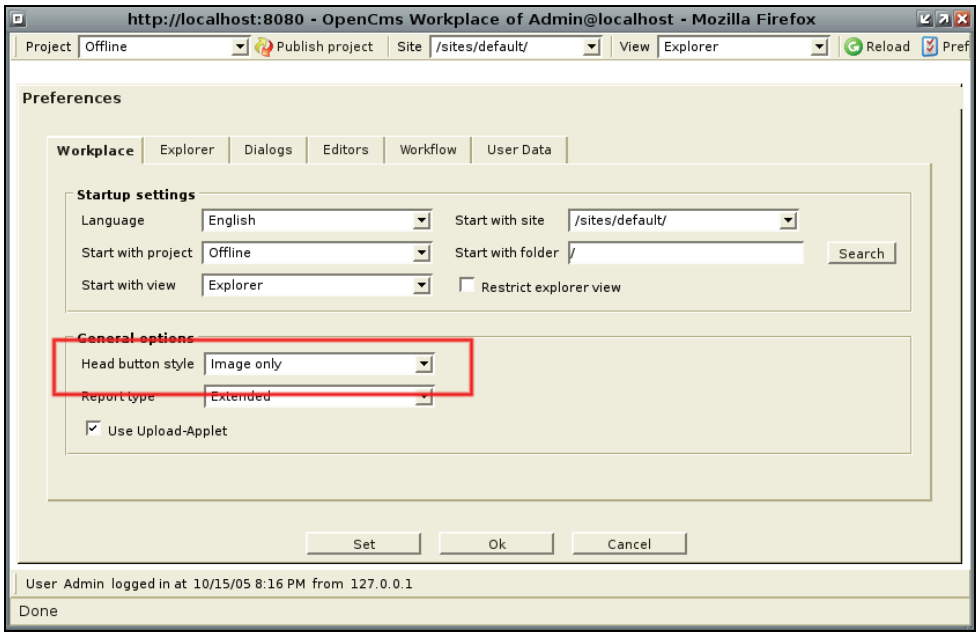
The first screen to display after logging in is the OpenCms Workplace explorer view. The explorer view shows the contents of the Virtual File System (VFS). OpenCms stores content in the VFS. While the VFS looks like a regular file system, it actually stores the files in the database, not in the server's regular file system. We will talk about this in more detail later in this chapter.

If your Display is Not Big Enough...

By default, the Workplace is configured to display on a screen that is at least 1024 pixels wide (and, optimally, larger than that). Attempting to display on a screen with fewer horizontal pixels will result in a partially obscured toolbar, and some very important buttons will be completely hidden from view.

To work around this problem, temporarily decrease the font size. (On Mozilla Firefox you can do this by simultaneously pressing `Ctrl` and `-`.) Now you should be able to see at least a few more buttons. Click the Preferences button on the toolbar at the top.

On the Workplace tab (the tab that comes up by default), under the subsection entitled General options, look for the drop-down list labeled Head button style. Change it from Image and text to Image only. Then press the Ok button so that your changes will take effect.

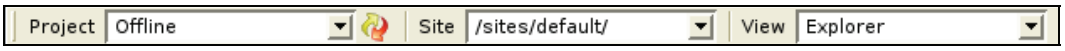


When you return to the Workplace, all of the toolbar buttons should be visible (though without any text). If you hover the mouse pointer over a button, the button name will display in a floating tool-tip box.

The Toolbar

Along the top of the workplace window is the **toolbar** (sometimes referred to as the header strip).

The left side is composed of three drop-down lists:



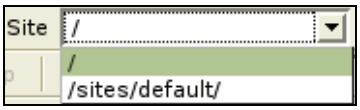
The Project drop-down list allows you to choose the project with which you will work. By default, there are two projects: Online and Offline. The **Online** project contains files currently published. No documents in the Online project can be edited.

The **Offline** project contains all of the files in the VFS. In a way, it is the master working copy of the VFS. Other projects (we will create one in the following section) are essentially sub-projects of Offline. That means that while they are working in the same part of the VFS, users in that project have access to only a subset of the files that are in the Offline project. In Chapter 4 we will examine creating projects and assigning project managers to those projects.

To the right of the Project drop-down list is the Publish project button.

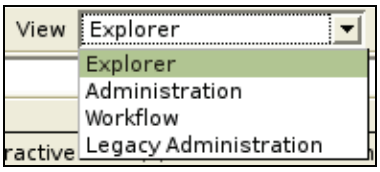
Clicking on this button will publish all of the resources in the given project. In the Offline project, this button publishes all of the documents in the repository. In the Online project, it is always marked inactive (gray), as in the image on the previous page, since the Online project is where things go when they are published. This may seem a bit confusing at the outset, but we will return to projects and publishing several more times during the course of this book.

Next to the Publish project button, is the Site drop-down list.



This lists all of the different websites that are stored in this instance of OpenCms. The items in this list are actually folders in the Virtual File System (VFS). By default, there are only two: `/`, which is the root of the VFS (and is not really its own website), and `/sites/default`, which is the default website. Right now, we want to be in the `/sites/default` site.

The next drop-down list is the View list.



The View drop-down list contains four items: Explorer, Administration, Workflow, and Legacy Administration. These correspond to the Workplace's four views:

- The explorer view is the current view. Most editorial work is done using the explorer view.
- The administration view provides access to all of the technical management tools, and is used mainly by the system administrator.
- The workflow view provides a project manager with a task-based interface for handling projects, rather than individual pieces of content.
- The legacy administration view provides the same content as the administration view, but in the layout familiar to OpenCms 5.0.x users.

For the time being, we will focus on the explorer view, but we will return to the administration and workflow views during this tutorial.

On the right half of the toolbar is a series of buttons.

> If you had to follow the instructions in the section entitled "If your display is not big enough", then your toolbar buttons will not have labels.

The first button on the toolbar is the Reload button.

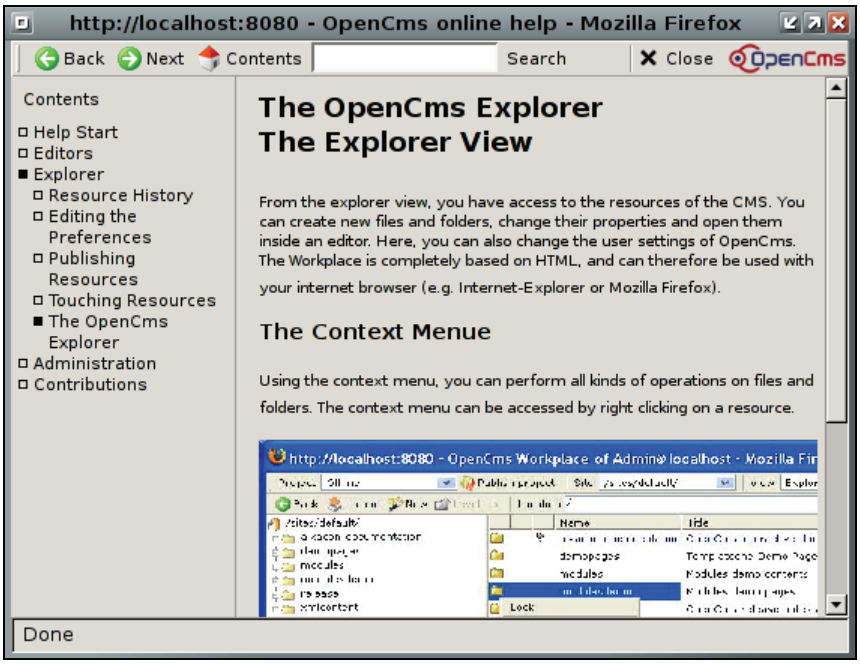
As the name suggests, it reloads the page. In fact, it works identically to the reload/refresh button in your browser.

Next to the Reload button is the Preferences button.

As we have already seen, this button brings up the Preferences panel. Personal preferences and user information (including the password) can be changed here. We will be looking at the Preferences panel in detail in the next section.

After the Preferences button comes the Help button.

Clicking on this will open a new window with context-sensitive help information.



The final button along the tool bar, labeled Exit, is the log-out button.

Clicking on this button will clear all of your log-in information and bring you back to the log-in screen.

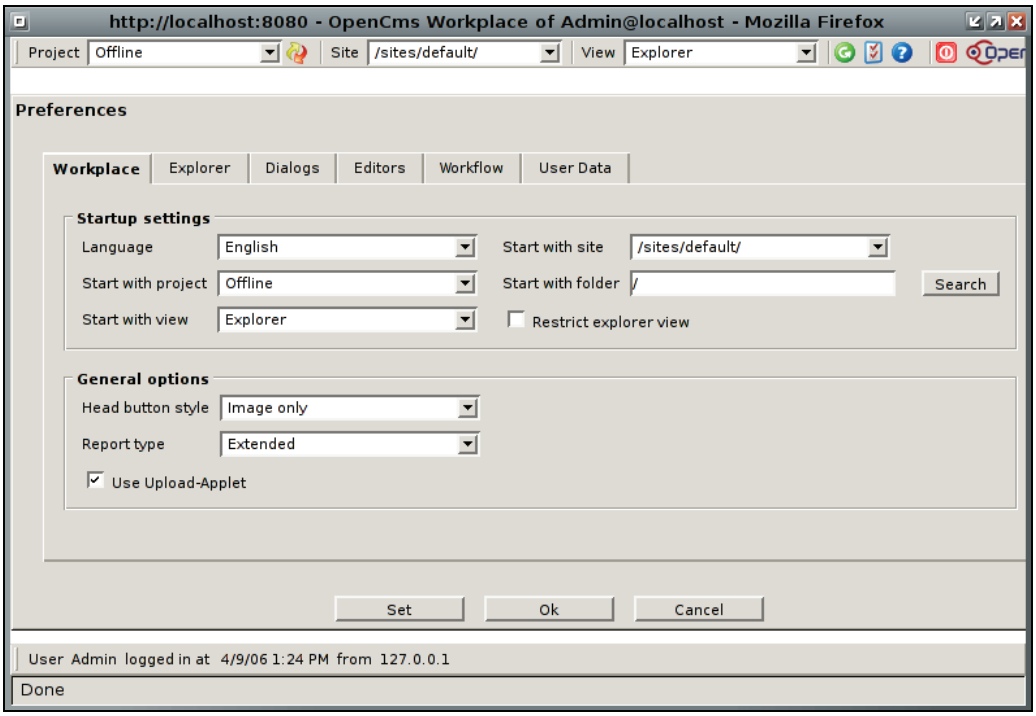
Do not log out. Instead, click on the Preferences button—let's personalize.

The Preferences Panel

The Preferences panel is arranged into six tabs worth of settings. We will start with the first tab and work through each.

The Workplace Tab

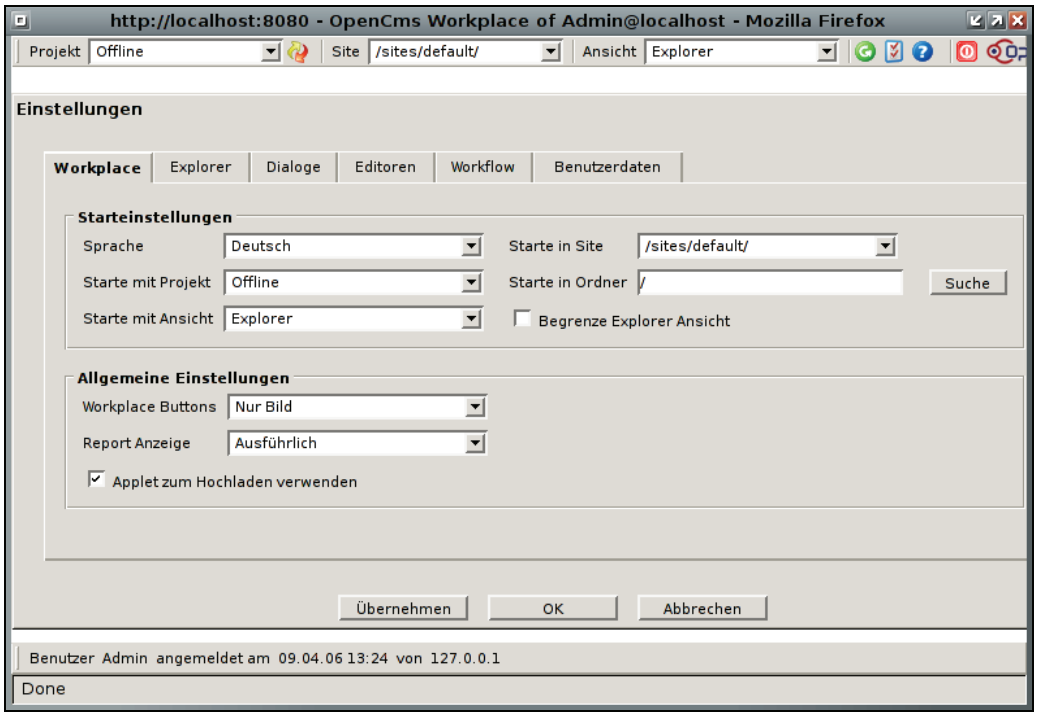
The first tab, Workplace, looks like this:



The Workplace tab is concerned with the overall configuration of the Workplace. It is divided into two sections: Startup settings, which determines how the Workplace will be configured when you log in, and General options, which determines the look and feel of the Workplace.

The Startup Settings Section

There are six fields in the Startup settings section of the Workplace tab. The first is the Language setting drop-down list. This is used to set the default language in which the text within the workplace will be displayed. For example, if you change the language from English to German, then the label for this box will change from Language to Sprache. (All the other labels in OpenCms will be changed as well.)



However, changing the language for the workplace will not have any effect on the languages in the pages created by OpenCms.

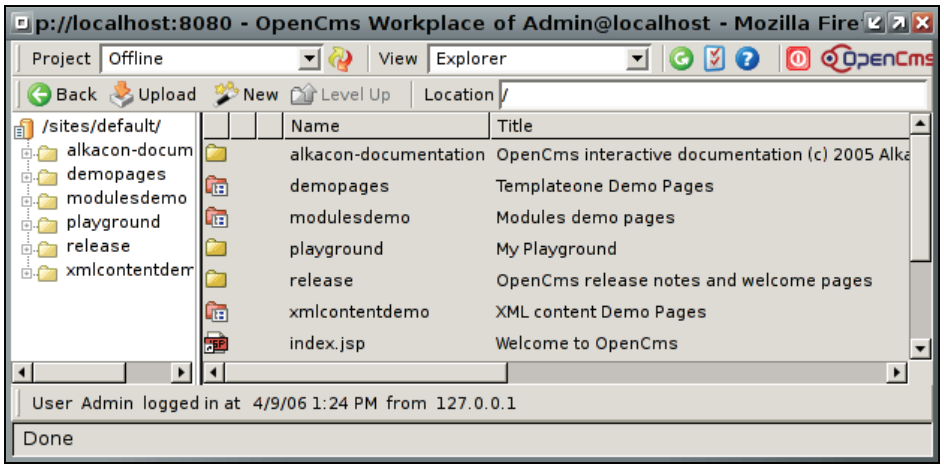
Below the Language Setting drop-down list is the Start with project drop-down list. In the beginning of the section entitled *The Toolbar*, I briefly introduced the concept of the Online and Offline projects. Since editing work in OpenCms is done entirely in the Offline project, you may want to set that to be the default project.

The last item in this column is the Start with view drop-down list. The four options in this drop-down list correspond to the options in the View drop-down list in the toolbar. This setting determines which view you will see when you first log in. The default option, Explorer, is best for most users.

In the second column of the Startup settings of the Workplace tab, the first item is the Start with site text field. We discussed the concept of sites in the previous section, and the selections in the drop-down list here mirror the selections in the Site drop-down list in the toolbar. For our purposes, `/sites/default` is best.

Beneath that is the Starts with folder text field. This determines which folder *within the default site* (as set above) the explorer view should show when you first log in.

The last item in this column is a checkbox labeled Restrict explorer view. Checking this box will restrict the explorer to viewing only the currently selected default site. In this screenshot you can see the impact of checking this box:



Notice that the Site drop-down list has been removed from the toolbar. This means you can no longer navigate in the VFS to a level above the current site (`/sites/default`). Effectively, this limits you to viewing only one particular site—and it prevents you from accessing the root folder `/`, as well. Such a setting may be very beneficial to editors and other workplace users, but if you check it now, you may find yourself very frustrated when we talk about working in the root folder or creating other sites.

The General Options Section

The second section of the Workplace tab is called General options. This section only has three items.

You may already be familiar with the first item, the Head button style drop-down list. This lets you determine how the buttons in the toolbar will look. You can configure them to appear with icons and text, just icons, or just text. If you are pressed for space, you may want to select Icons only.

Beneath the Head button style drop-down list is the Report type drop-down list. When you publish a resource from the Offline project to the Online project (for example, by clicking the Publish button in the toolbar), OpenCms will display a detailed list of all of the steps it is performing in the process of publishing by default. This is the Extended option. If you choose the Simple option instead, the lengthy report will be replaced with a few simple icons indicating the status of the publishing cycle. The Simple option does not provide as much debugging information, and many users may find that attractive.

Finally, the last item in the General options section is a checkbox called Use Upload-Applet. The **Upload Applet** is a Java applet that assists with uploading content from your local hard drive to the OpenCms server. It can be tremendously useful if you have the Java plug-in or control installed. Some environments do not support Java. In such a case, you can uncheck this box. A very simple HTML upload form will be used in place of the Upload Applet.

> In order to use the Upload Applet, a client (web browser) will need to have the Java Runtime Engine (JRE) or the Java Developers Kit (JDK or JSDK) installed.

The Explorer Tab

The second tab in the Preferences panel is the Explorer tab. This tab allows you to set preferences for the explorer (file browser) view in the OpenCms Workplace. This tab has two sections: General options and Display options.

> For more about using the explorer view of the OpenCms Workplace, see the section *The Explorer View* later in this chapter.

The General Options Section

There are two drop-down lists in the General options section. The first, Explorer button style, determines how the explorer buttons are displayed. By default, they appear as text and images.

The Explorer button style drop-down list has three options: Image only, Text and image, and Text only.

Beneath this is a second drop-down list called Number of entries per page. This determines how many files (including folders) will be shown per page in the explorer view. For instance, suppose we have a folder with fifty-two items. If we leave this setting at the default (50), then the explorer view for that folder will be broken into two pages: one with fifty items, and the other with two items. In such a case, a new item will be added to the explorer toolbar: a drop-down list for selecting which page to view.

The Display Options Section

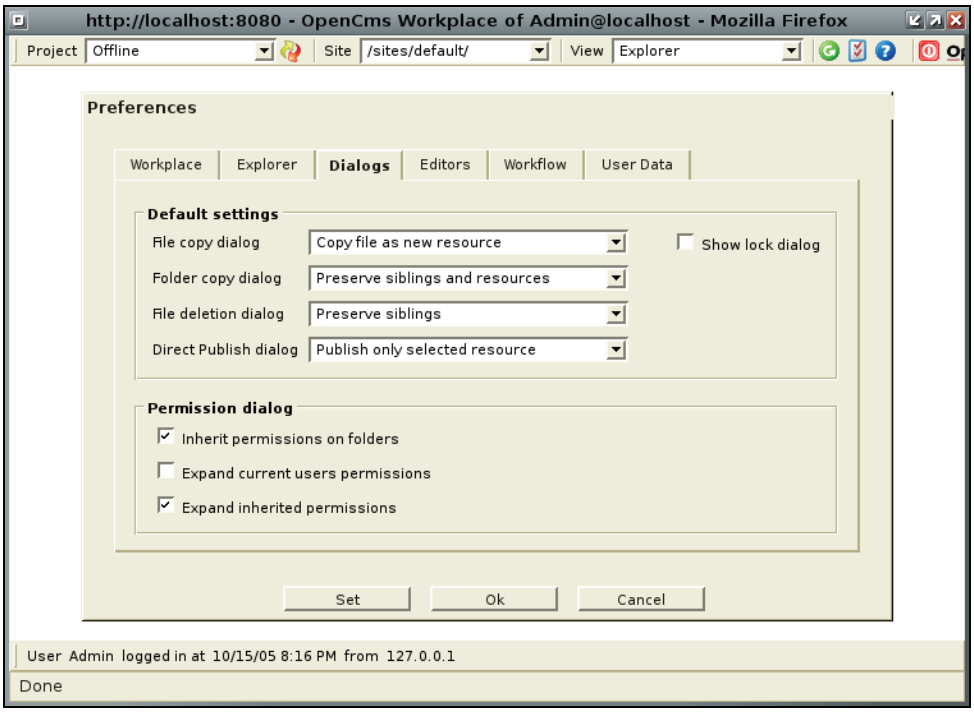
This is the second section on the Explorer tab. It has twelve checkboxes: Title, Type, Size, Permissions, Date last modified, Last modified by, Date created, Created by, Date released, Date expired, State, and Locked by. These determine which fields are displayed in the tabular list of files in the main explorer view. For example, if you check only the Title and Type checkboxes, then the explorer file view will look like this:

Name	Title	Type
alkacon-documentation	OpenCms interactive docs (c) 2004-2005 Alkacon Software	folder
demopages	Templateone Demo Pages	microsite
modulesdemo	Modules demo pages	microsite
release	OpenCms release notes and welcome pages	folder

Note that there is no option for hiding the icon or the name.

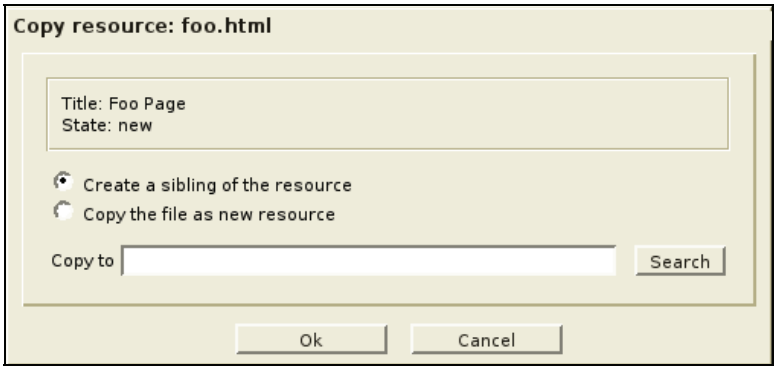
The Dialogs Tab

There are a number of different dialog windows that are displayed in OpenCms, and several of them are configurable in the Dialogs tab of the Preferences panel. This tab is broken into two sections: Default settings, which deals with most of the configurable dialogs, and Permission dialog, which provides several settings for one particular dialog.



The Default Settings Section

The Default settings section of the Dialogs tab has five fields. The first is the File copy dialog drop-down list. The file copy dialog appears when you right-click on a file in the explorer view and select copy. This dialog always gives you two choices for copying: Create a sibling of the resource and Copy the file as a new resource.



If you choose the first of these, the original file and the copied file are coupled together making them into **siblings**. The "sibling" relationship was added in OpenCms 6.0 and provides a way to group files together for editing and publishing. If two or more files are siblings, their lock states are always the same so that, when an editor locks a file for editing, no other editors will be able to access the other—it will automatically be locked. Locks are discussed in more detail in the section below entitled *The Explorer View*, and siblings will also be covered later in this chapter.

In addition, when a file is published, you have the option of publishing all its siblings. Choosing this option will publish all files copied as siblings of that file. Copying as a new resource simply makes another copy of the file, unrelated to the original in any way.

The File copy dialog setting drop-down list determines which of these two options will be selected by default. However, you will still be prompted to choose whenever you copy a file.

The next item in the Default settings section is the Folder copy dialog drop-down list. This is similar to the File copy dialog drop-down list, but it applies to folders. There are three options that can be set as the default option for the dialog.

- Copy resources as siblings: This option will make all resources in the newly copied folder into siblings.
- Preserve siblings and resources: If there are siblings in the original folder, this option will preserve their relationships in the copies. However, resources that are not siblings will remain unrelated in the copied folder.

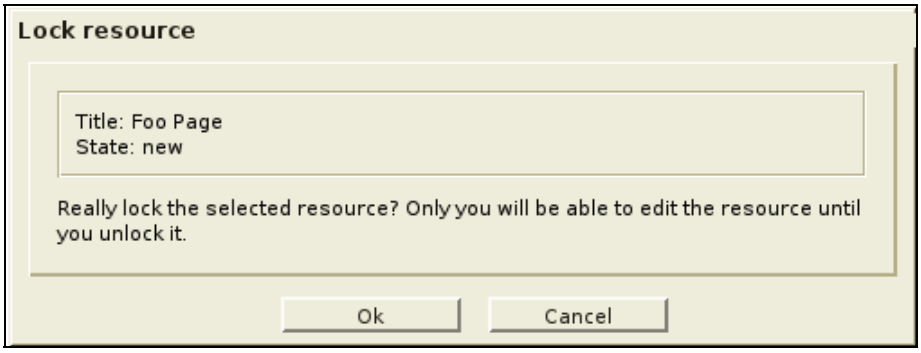
- Copy resources as new resources: No sibling information from the original files will be preserved in the new copy.

As with the File copy dialog drop-down list, this drop-down list only determines which option will be selected by default. When you copy a folder, you will still be presented with all three options.

The next item is the File deletion dialog drop-down list. This determines which of two options will be selected by default in the file deletion dialog. If the first option, Delete siblings, is selected, the file *and* all its siblings will be deleted. If the second option, Preserve siblings, is selected only the file will be deleted; all siblings will remain untouched.

The fourth item in the Default settings section is the Direct Publish dialog drop-down list. As mentioned above, when you choose to publish a file, you are presented with a dialog that contains a checkbox called Publish all siblings. If this box is checked, then the sibling files will be published along with the selected file. Its default state can be chosen using the Direct Publish dialog drop-down list. If you choose the Publish all siblings option from the Direct publish dialog drop-down list, then the box will be checked by default. If you select the Publish only selected resource option, then the box will not be checked by default.

Finally, to the right of these four drop-down lists, there is a checkbox called Show lock dialog. If this is checked, then whenever you click on a file icon and choose lock or unlock, you will be prompted with this warning dialog.

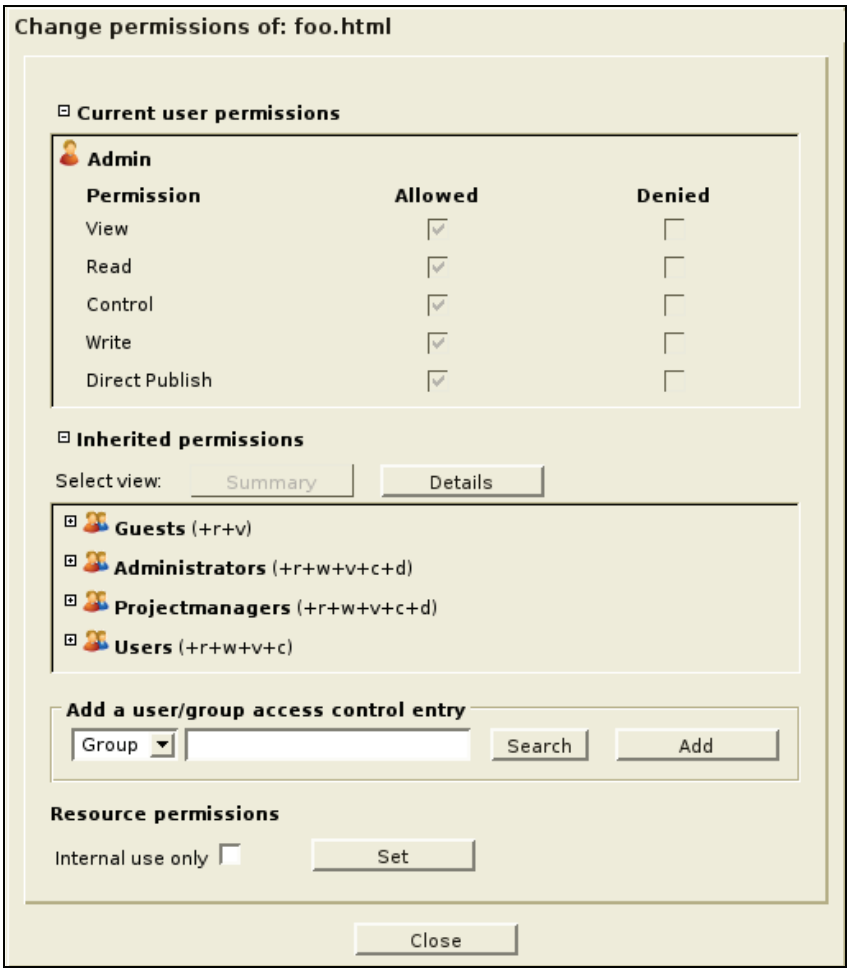


However, if the Show lock dialog box is unchecked, you can change the lock state without receiving a warning. Unless you have (or one of the system's users has) trouble remembering what locking does, the warning is usually an inconvenience and an annoyance.

The Permission Dialog Section

The second section of the Dialogs tab is concerned with the permissions dialog. We will start by looking at the permissions dialog itself.

To access the permissions on a particular file, you will need to go back to the explorer view. When you click on a file's icon in the explorer view and then choose permissions, the permissions dialog will appear.

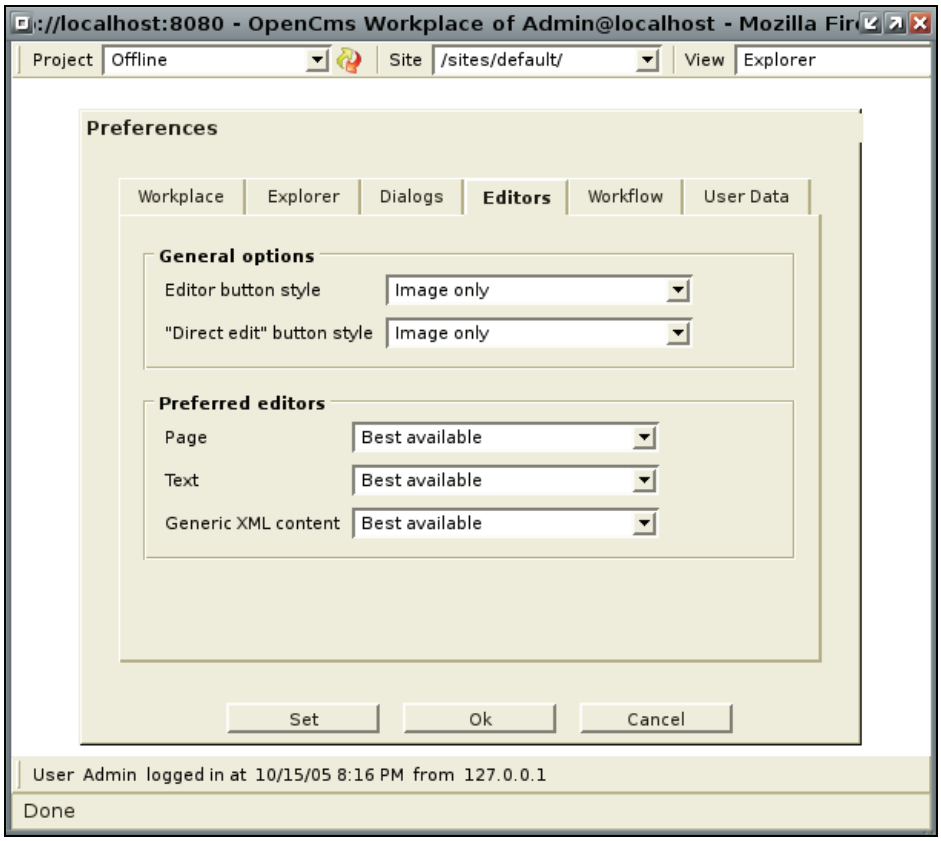


The three checkboxes in the Permission dialog section of the Dialogs tab determine how this dialog will work. The Inherit permissions on folders checkbox determines whether or not the permissions for folders are inherited from the parent folder.

The other two settings, Expand current users permissions and Expand inherited permissions, determine whether the two expandable sections are expanded when the dialog appears, by default (as they are in the image above).

The Editors Tab

OpenCms 6 supports a number of different content editors for use when creating content. This tab provides customization for editing. There are two sections on this tab: General options, which applies to all editors, and Preferred editors, which allows you to specify which editors you want to use.



The General Options Section

There are two options in this section. The first is the Editor button style drop-down list, which offers the now familiar choice of options that determine how buttons appear—Image only, Image and text, or Text only—this time it's for the buttons in the content editors.

The second item, the "Direct edit" button style drop-down list, provides the same choices for the Direct edit button—a button that appears at the bottom of a content section in a published page to users who are logged in and have permission to edit the page.

Clicking this button will open the selected content section in an editor.

The Preferred Editors Section

This section of the Editors tab provides the user with the ability to choose specific editors for specific kinds of content. It has three drop-down lists: Page, Text, and Generic XML content.

Page content is HTML content, and there are three WYSIWYG editors that can edit it. Of these editors, FCKEditor and HtmlArea work in browsers other than Microsoft Internet Explorer, but the Microsoft DHTML editor only works in Internet Explorer. The FCKEditor is the default, and is almost always the best choice.

Text content is plain text with no markup. It can be edited using LeEdit OCX editor, which is an Active X control, but this editor is not much better than a basic bare-bones text editor.

Generic XML content contains XML data.

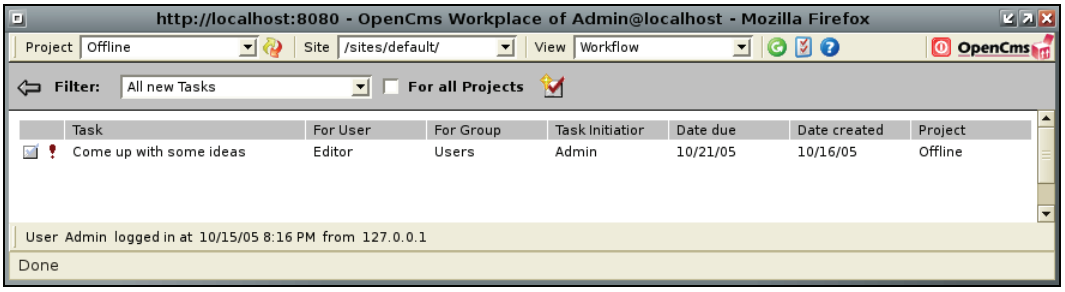
In all cases, choosing the Best available option is fine. In this case, OpenCms will attempt to determine which of these editors will best fit your environment.

The Workflow Tab

The fifth tab in the Preferences panel is the Workflow tab. Workflow is the process involved in taking a piece of content from creation to publishing. This tab has two sections (just like the other tabs): General options, which applies to the workflow view in general, and Defaults for new tasks, which applies to the creation of a task in the workflow view.

The General Options Section

The General options section of the Workflow tab has two fields. The first of these, Startup filter, sets a filter to determine what types of workflow tasks will be displayed when you switch to the workflow view. The workflow view looks like this:



Notice the first drop-down list in the dark gray toolbar. The Startup filter field determines which of those options will be selected by default.

To the right of the Startup filter field, is a checkbox called Show all projects. We've discussed the Online and Offline projects already, but you can create custom projects as well. By default, the workflow view shows only one project at a time (as determined by the Project drop-down list in the toolbar), but if the Show all projects box is checked, then all of your tasks in all projects will be displayed on the workflow screen.

The Defaults for New Tasks Section

The Defaults for new tasks section has four checkboxes: Message when accepted, Message when forwarded, Message when completed, and Inform all role members of the task.

These correspond to four identically named checkboxes in the Create New Task dialog in the workflow view. Checking one of the boxes will cause the corresponding box to be checked by default in the Create New Task dialog in the workflow view.

The User Data Tab

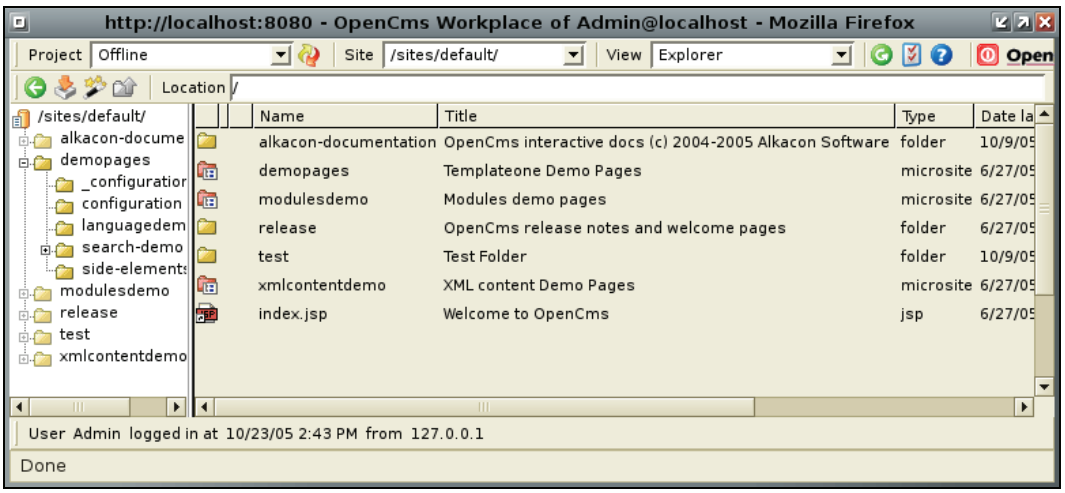
The last tab in the Preferences panel is the User Data tab. On this tab, you can view information about a particular user. You cannot, however, change the information from here. Only administrators can change this information, and they do so through the administration view (as we will see in Chapter 4), but you can change your password.

> If you are the Admin user and you are using the default password, `admin`, you should change it immediately by clicking the Change Password button.

This completes our tour of the preferences panel. Next, we will look at the OpenCms Workplace's explorer view.

The Explorer View

By default, when you log into the OpenCms Workplace, you will start with the explorer view. It is similar to the Windows Explorer in that it is designed to help you navigate through a file system.



In this section, we will walk through the features of the explorer view.

The Button Bar

Just under the toolbar is the explorer view's **button bar** (sometimes referred to as the button strip or the nav bar). It contains buttons specifically related to exploring and editing content.

As in the Windows Explorer, the Back button (the green circle with a white left-pointing arrow) changes the location to the previously visited folder.

The Upload button (an orange arrow pointing down into a gray box) will bring up the file upload screen from which you can load a file from your local disk into the OpenCms VFS.

Clicking the New button (a magic wand with yellow stars above it) will bring up the file creation screen. This is used for creating *any* type of file, including a folder.

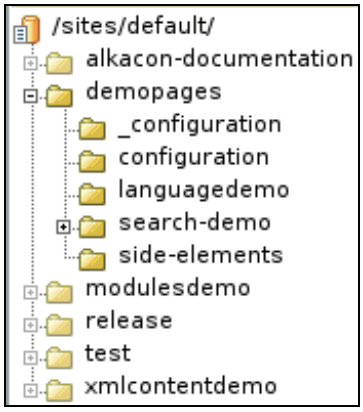
Finally, the Level Up button (yellow folder with a blue arrow pointing up) will change the location to the parent directory—for example, if you are looking at the contents of the `/release/` folder (technically, the `/sites/default/release/` folder), clicking the Level Up button will get you to the `/` folder (technically, the `/sites/default` folder).

Finally, the Location text field displays the current location (relative to the site). If you type in a directory path and press the Enter button on your keyboard, the contents of that directory will be displayed

When typing a path into the Location text field, do not type a file name at the end. End the path with a folder name. For example, `/release/` is OK, but `/release/welcome.html` will be ignored.

The Virtual File System

On the left-hand side of the view there is a file tree display that shows all the file folders in the **Virtual File System** (**VFS**):



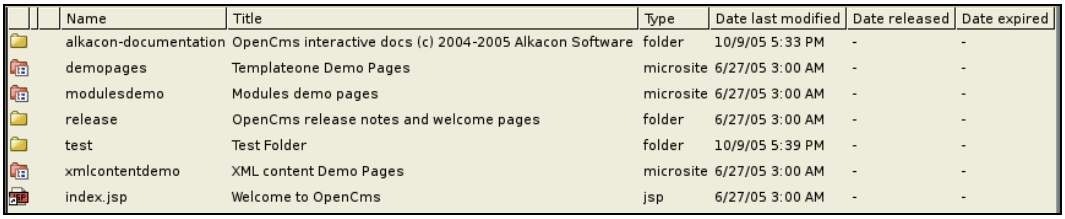
The VFS works like a normal file system, but stores the data inside the database. For that reason, you will not be able to find any of these files in the VFS in your regular file system. Because the VFS looks and acts like a file system, VFS paths are expressed like URL- or UNIX-style paths. The root folder is `/`, and other folders are noted by a name followed (optionally) by a slash. By default, the explorer view starts in the folder `/sites/default/` (as you can see in the window above). Inside this folder there is an entire subtree of folders: `alkacon-documentation`, `demopages` (with its own set of subfolders), `modulesdemo`, and so on.

> In some cases, the files in the VFS can be copied into the regular file system. During publishing, OpenCms can publish resources into the regular file system. These files are *copies* of those inside OpenCms. Be aware that editing these copies is not the same as editing the content of the VFS, and your changes will not be saved in the CMS. To change the content inside the VFS, you must use OpenCms or configure the experimental synchronization feature discussed in Chapter 4.

Clicking on any of the folders in the left-hand pane will display the contents of that folder inside the center pane of the window. Only folders show up in the left-hand file tree; folders, files, and links are all displayed in the center frame—the file detail display.

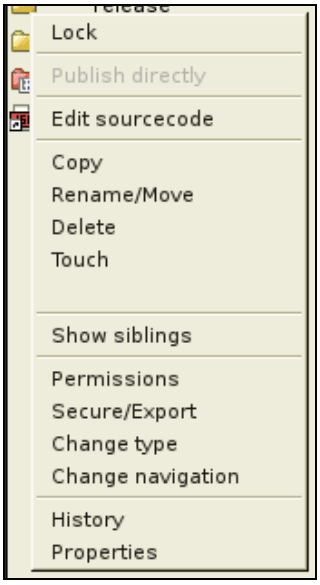
The File Detail Display

The central pane of the explorer view displays a detailed look at the files stored in the currently displayed location in the VFS.



Name	Title	Type	Date last modified	Date released	Date expired
alkacon-documentation	OpenCms interactive docs (c) 2004-2005 Alkacon Software	folder	10/9/05 5:33 PM	-	-
demopages	Templateone Demo Pages	microsite	6/27/05 3:00 AM	-	-
modulesdemo	Modules demo pages	microsite	6/27/05 3:00 AM	-	-
release	OpenCms release notes and welcome pages	folder	6/27/05 3:00 AM	-	-
test	Test Folder	folder	10/9/05 5:39 PM	-	-
xmlcontentdemo	XML content Demo Pages	microsite	6/27/05 3:00 AM	-	-
index.jsp	Welcome to OpenCms	jsp	6/27/05 3:00 AM	-	-

As in Windows Explorer, the icon at the beginning of each line indicates what type of data a file contains. Left-clicking on the icon will display a context menu of actions. For instance, clicking on the page icon next to `index.jsp` opens a context menu like this:



Note that some of the values are grayed out, indicating that they are not available. They will become enabled when they can be used—for example, the publish directly item will be activated when the file has been edited but not yet published.

> OpenCms uses a left-click rather than a right-click to open context menus. Using a right-click will open the web browser's context menu. While this change in default behavior can be confusing at the outset, it lets you take advantage of the browser's context menu.

During the course of this book, the function of each of these menu items will be explored fully; for now, I will provide a brief summary of each:

- Lock or Unlock creates or removes, respectively, a lock on the file. When you lock a file, a blue icon will appear between the file icon and the document name. This is to indicate that, for you, the file is unlocked and that, for all others, the file is locked. Likewise, if another user locks a file, you will see another icon ( ) that indicates that the file is locked for you so that you cannot edit it.
- Steal lock will take the lock away from another user and give it to you. This is sometimes useful (for example, where a user forgets to unlock a file before she or he leaves on holiday), but it is extremely bad CMS etiquette to steal a lock from a user without asking permission first.
- Publish directly publishes just this file and any resources on which it depends, such as images or templates. Also, if the file has siblings, the siblings will also be published. This is a useful alternative to publishing everything with the Publish project button in the toolbar. This item becomes active when the file in the current project (Offline in this case) differs from the version that is available to the public. Pages that are available to the public are stored in the Online project, and publishing a page puts the current working version of the page into the Online project.
- Edit page opens the page in an editor. This is the main tool used to create content. By default, it uses a WYSIWYG editor similar to a desktop word processor, so you do not have to write HTML code by hand. If you prefer to write HTML code by hand, you can configure it to use a plain textbox instead. Note that this tool is only available for files of type Page.
- Edit *item* provides a way to edit the contents of the particular resource. For example, Edit sourcecode opens the page in the sourcecode editor so that you can edit the HTML, XML, and Java Server Pages (JSP) code within OpenCms. (We will use this editor in Chapter 6.) Other options include Edit for special types and Edit link for link types.
- Copy copies the file.
- Rename/Move changes the name of the file or moves the file to a different directory.

- Delete removes the file. Note that it will still be visible with a strike-through mark until it is published (which will remove it from both the Online and Offline projects).
- Touch changes the time stamp on the file, effectively marking the file as modified. The next time the project is published (moved to the Online project where site visitors can see it), this file will be re-copied to the Online project.
- Undo changes rolls back to the previous published version of the file. This option is only available on files you have modified since the last time you published.
- Show Siblings displays all the files that are declared siblings of the currently selected file. Note that when a file is locked, all of its siblings are also locked, and by default, when a file is published, all its siblings are published too.
- Permissions changes the access controls for the file. You can determine who has permission to do what (read, write, control, direct publish, view) the file. You can set these permissions for groups and particular users, allowing you to set fine-grained permissions on your files. There is one extra checkbox in the Change permissions window: Internal use only. This is for pages that are used only in the Workplace, and should not be published. Managing permissions is discussed more in Chapter 4.
- Secure/Export provides access to security and export settings. You can determine whether or not a file is internal (see above), and whether or not a file should be exported when published or requested.
- Change type allows you to change the type of the file. For example, you may want to change the file from a plain Text type to a JSP type. *Use this with caution*—changing a type will not alter the content of the file; it will only alter how the file is treated by OpenCms. For example, changing from type Text to type JSP will not change the contents of the text file into JSP code.
- Change navigation changes the way the file appears in the site navigation in the published website. OpenCms builds the navigation (menus and breadcrumb trails) automatically. When you create a new file, you may choose to include it in navigation, which is built dynamically for each directory. This option allows you to change the navigation settings created along with the file.
- Edit controlcode provides direct access to the XML format in which the file is stored. This feature can be useful for troubleshooting.
- History shows all the changes that have been made to a particular file, and allows you to roll back to *any* earlier version of the document in the repository. Using this option, you can see how many times the file has been published and what has changed. As of OpenCms 6.2, you can even compare two versions and get a line-by-line analysis of the changes.

- Properties gives you access to data about the document—for example title, keywords, position in the navigation. We will look at these properties in Chapter 4. If a file is locked by another user, you will only have read-access to the properties. If you own the lock or if the file is not locked, you can edit the properties.

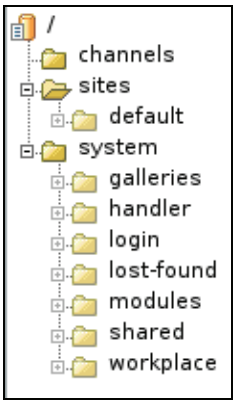
Some of these options will not be available for some types of file. For example, only a few options are available for links. Also, when a file is locked, only a few options (such as Steal lock, Show siblings, and Properties) are available for its siblings (which will also be locked).

Exploring the VFS

You may notice that you cannot go from `/sites/default` to `/sites`–in other words, you cannot traverse *up* the hierarchy beyond the level at which you began. The Level Up button is grayed out and inactive, and the file tree display shows `/sites/default` as the top of the tree. This feature is designed to prevent editors from accidentally getting outside of the location where their content belongs. Unless an administrator has configured your account otherwise, you can move to a higher level of the VFS file tree by changing sites.

The Root Folder

At the top of hierarchy is the root folder. To get to this folder, you must choose the path `/` (a single slash) from the Site drop-down list in the toolbar. The root folder looks like this:



The root folder is the base of the Virtual File system so that, when you publish the Offline project, you are publishing everything inside the root folder (including all of the contents of the `/sites` folder). There are three folders inside the root folder: `/channels`, `/sites`, and `/system`. Each of these folders has a special purpose.

> Do not delete any of these three folders. Doing so can cause serious damage to OpenCms, even rendering it unusable. Also, you should not create additional folders or files in this directory unless you know what you are doing.

The Channels Folder

The `/channels` folder is for storing information about OpenCms **channels**. Channels are an advanced feature used for organizing content. Because the use of channels requires some custom software development, channels are not covered in any detail in this book.

The Sites Folder

The `/sites` folder contains a separate subdirectory for each site that is to be served by this instance of OpenCms. A **site** is an organizational unit in OpenCms. Usually, each site is considered to be a stand-alone unit (that is, not reliant on other sites), and will have its own domain name (e.g. `http://mysite.example.com`), as well as its own set of editors or content owners.

Content should only go inside of a subfolder in the `/sites` directory. Usually, this means that content should only go inside of the `/sites/default` folder, unless you are hosting multiple sites on your OpenCms server.

> Content should absolutely *never* be placed in the root folder, or in the `channels/` or `system/` folders. Doing so can result in unexpected behavior, and in some cases can compromise the security of your system.

Items in the `/sites/default` folder are published to the root of the OpenCms URL. For instance, a file named `foo.txt` in the default site folder would be published to `http://localhost:8080/opencms/opencms/foo.txt`. As another example, the `/sites/default/release` folder contains OpenCms release notes. If you point your browser to `http://localhost:8080/opencms/opencms/release/` you can access the release notes.

The System Folder

The `/system` folder contains items that are functionally important for OpenCms, and each of its subfolders has a particular function.

- `galleries`: This folder contains shared resources (such as images or commonly-used fragments of HTML). These are grouped together into **galleries**. Each of the galleries is stored in a subfolder of the `/system/galleries` folder. While it is possible to maintain the galleries by working directly with these subfolders,

there are tools in the Administrator view that are much better suited to the job. We will look at galleries at the end of this chapter.

- `handler`: Under certain circumstances, OpenCms executes different **handlers**. Supporting files for these handlers are stored in the `handler` directory. For example, error pages are stored in the `/system/handler/contents` folder.
- `login`: The `login` folder contains files for managing user logins (typically the login screen for the OpenCms Workplace).
- `lost-found`: If a page should become "lost" in the VFS, and its location cannot be determined for some reason, then it will be placed in the `/system/lost-found` folder. This is similar to the concept of the `lost+found/` folder typical in Linux file systems.
- `modules`: The `modules` folder is where new OpenCms modules are placed when they are installed. Modules are used to extend the basic function of OpenCms. For example, help text is packaged in a module, as are functional add-ons such as calendar widgets, search engines, and guest books. In Chapter 4, we will look at modules in more depth.
- `shared`: The next subfolder, `shared`, contains a couple of templates that are shared by many resources across multiple sites.
- `workplace`: This final folder contains all of the files—templates, images, and content—that make up the very Workplace itself. Developers can customize the Workplace (through the Workplace) by modifying these files. As you may have guessed, editing these files can be dangerous, and mistakes can lead to disaster. If you decide to edit these files, do so on an installation that does not contain important data.

The contents of the root folder are usually only used for administration and development purposes. Most content editors will never need to work directly in the root folder (or in the `/channels` or `/system` subfolders). Content should be restricted to subfolders of the `/sites` folder.

Now that we have examined the main root folder, we will go back to the `/sites/default` folder (use the Site drop-down list in the toolbar) and continue working from there.

Creating and Editing Content

Thus far, we've examined how to navigate through the VFS using the explorer view. Now that we are comfortable with the navigation, it is time to create and edit some content.

To start, let's create a working area in which we can do some experimenting. All content should be placed inside a site folder (and not in the VFS root folder). So the first thing to do is choose `/sites/default` from the Site drop-down list in the toolbar. The Location bar should now show that the path is `/`, which indicates that you are in the base directory of the `/sites/default` folder.

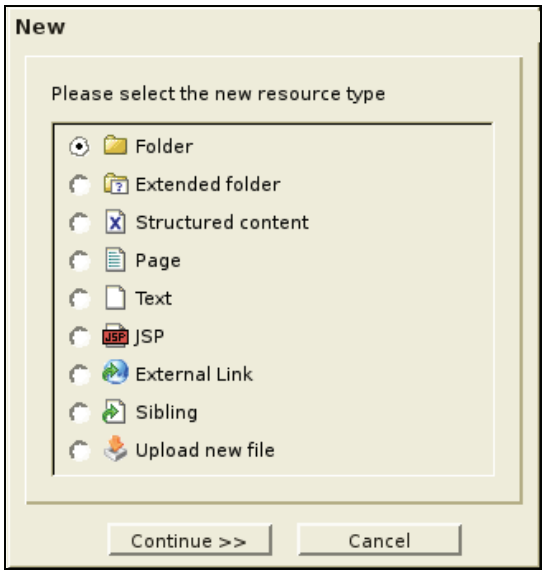
> The Location bar always shows the path relative to the current site. So the path `/` in the location bar does not refer to the root of the VFS unless the Site drop-down list is also set to `/`. When Site is set to `/sites/default`, then the value `/` in the Location bar refers to the directory `/sites/default`.

If you are not already in the root folder of the default site (that is, if the Location is not `/`), click on the /sites/default/ icon in the left-hand file-tree pane. This will take you to the root folder for the default site. The central pane of the explorer should now show a couple of folders and the `index.jsp` file. We now see how to create files and folders.

Creating a File

To create a file, click on the magic wand icon:

This will open a file creation wizard:



File Types

The first step in creating a new file is to choose the file type. The following file types are listed by default on the first screen of the file creation wizard:

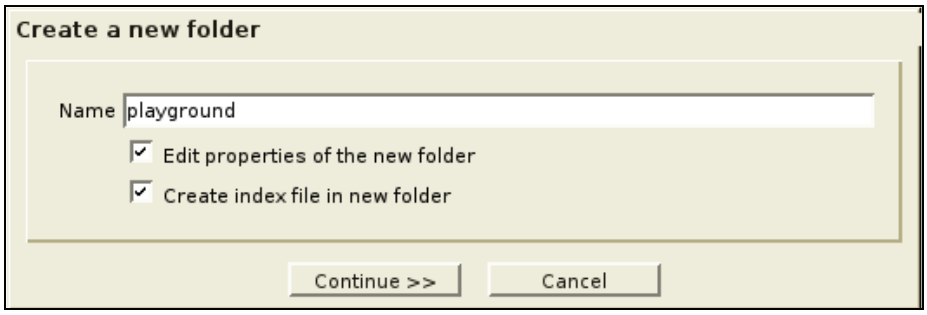
- Folder works just like a folder in the file system.
- Extended Folder is a variation on the regular Folder type. It adds additional features to a folder. There are half a dozen types of extended folder, ranging from Image Gallery to Microsite. If you choose Extended Folder, you will be prompted to choose which of these extended types you want.
- Structured Content is used for creating documents that have specialized fields or specialized content. For example, an event might have specialized fields that normal pages would not have, such as time, a location, and registration information. As with Extended Folder, if you select this type, you will have to select an extended type as well.
- Page is an HTML document that can be edited in the WYSIWYG editor. We will create one of these soon.
- Text (called plain text elsewhere in the Workplace) is a generic type for any text-only content. While it is usually used to store formatted text-only documents like a "read me" file, it may sometimes be used to store data for a program.
- JSP is used to create dynamic content written as a Java Server Page (JSP).
- External Link creates a pointer from this location to a document outside of OpenCms. You will need to supply a full URL to the external resource (for example, `http://www.packtpub.com/`).
- Sibling will create a new sibling copy of an already existing file. As we saw earlier, the "sibling" relationship provides a way to group files together.
- Upload new file provides a method for selecting a file (which can be of almost any type, including images, text, or binary files) on your own machine and loading it into the CMS, creating a new file in the VFS.

> In earlier versions of OpenCms, there was also an XML Template type. As of version 6, OpenCms no longer uses XML templates. If you need XML template support, you will need to install the OpenCms Legacy module.

Creating a Folder

We will start by creating a new folder, or directory, so you should leave the Folder icon checked and click the Continue button. You will then get the Create a new folder dialog, which is used to set the new folder's properties. We will need to give the folder a name.

The name should contain only letters, numbers, dashes, and underscores. Since this is just an area for us to play around, let's name this folder `playground`. This will become the folder's name in the VFS.



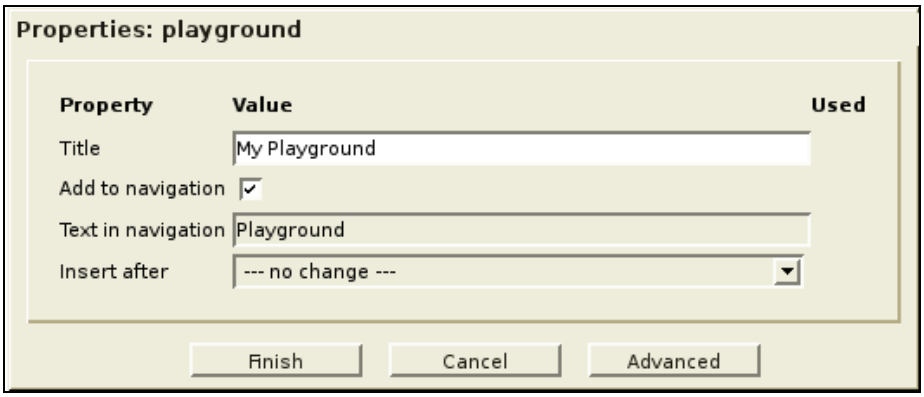
There are two checkboxes on the Create a new folder form. The first is called Edit properties of the new folder. If this is checked, you will have the opportunity to edit additional folder properties. (In a moment, we will look at the Edit properties screen.)

If the second checkbox, Create index file in new folder, is checked, OpenCms will automatically create a new index page for the folder, and you will be prompted to edit the properties for this page. It is good practice to always create an index page for a new folder (even if it is blank), since it prevents visitors from getting errors or directory listings when trying to load the folder. It is therefore generally a good idea to leave this box checked.

Now click the Next button to create the folder.

Setting the Folder's Properties

Since we left the Edit properties box checked on the last screen, we will be prompted to edit the folder's properties:



The first property is Title. A folder title differs from a folder name. While the name is used as a part of the path in the VFS (and in OpenCms URLs), the title is used in user-friendly displays.

The Add to navigation checkbox determines whether the folder should be included in automatically generated navigation menus. For this folder, we want to leave it checked. Note that if you uncheck it, the next two fields will become inactive and you will not be able to edit them.

> In OpenCms 6.0, the Text in navigation text field and the Insert after drop-down list have a gray background. This does not mean that the forms are deactivated. They are still active, and you may edit them.

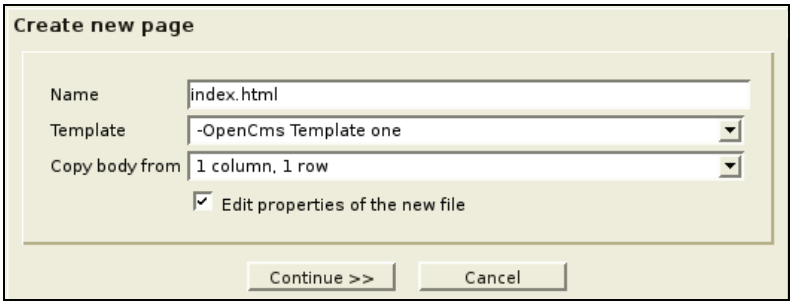
Like the Title text field, the Text in navigation text field is used to generate user-friendly interfaces. Specifically, OpenCms uses navigation text when it dynamically generates navigation. Why have the Text in navigation field when we already have Title? Because items in the navigation menu tend to be short (to conserve space), while titles are often longer and more descriptive.

The Insert after drop-down list determines *where* in the navigation this item will appear. You can set it to place the new folder immediately after one of the existing navigational items, or you can set it to appear at the beginning or end. For now, we'll leave this at the default.

We left the Create index file in new folder box checked when we created the folder, so clicking Finish will run the Create new page wizard in order to create an index file in our `playground` folder. It is good practice to always create an index page for a new folder. It prevents visitors from getting errors or directory listings when trying to load the folder.

Creating a Page

Since we left the Create index file in new folder box checked when we created the folder, clicking Finish takes us to the Create new page wizard to create an index file for our `playground` folder:



The first item on the Create new page dialog is the Name text field. This is for the new index page's file name. It is automatically set to `index.html`, which is what we want.

The Template drop-down list contains the names of OpenCms templates. We will leave this set to the default, -OpenCms Template one. This will use the site (or microsite) properties to determine which template should be applied to the content of the `index.html` file.

> *What is a template?* When you create a new page in OpenCms, you only need to create the basic text. The header, footer, and navigation (as well as formatting information) are stored separately in template files. OpenCms automatically builds the entire document by "applying" the template to your page and creating a completely formatted document.

The Copy body from drop-down list allows us to select the basic page layout from pre-defined formats (in the form of HTML code snippets). While it is possible to create your own (arbitrarily complex) HTML snippets, the default formats provide several common layouts. For now, we'll use the default 1 column, 1 row layout.

At the bottom of the form is a checkbox labeled Edit properties of the new file. If you leave it checked, then clicking Continue will take you to properties editor. Usually, you will want to leave it checked, as there are a few properties, such as title and description, that almost always need to be set.

Setting the Page's Properties

There are sixteen fields on this page:

- Title: This is for a user-friendly title (as opposed to the Name field on the last form, which specifies the file name). The page template will automatically put this title at the top of the document.
- Description: This field is used to provide a human-readable summary of the page's contents.
- Add to navigation: This works in the same way as for the properties of a folder. (We don't need to include the `index.html` file in navigation since we have already included the folder in navigation and both URLs will load the same content, so we will leave this box unchecked.)
- Show in Head Navigation: If this is checked, then the title of the page will appear in the main site navigation as well as in the regular context navigation.
- Show Head Image: This determines which image is shown in the header section of the document when it is displayed. There are three radio buttons from which to choose: Default, Individual, and Disable. Default uses the image

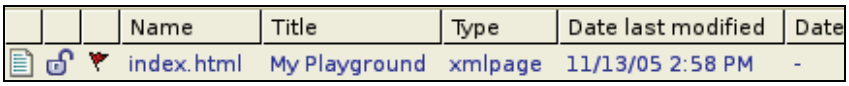
that is defined in the Template one configuration. Individual allows you to set a special image just for this page. Choosing Disable will result in no image being displayed in the document.

- Image Uri and Image Link: These are only enabled if Show Head Image is set to Individual. In this case, you can choose which image (Image Uri) to display and provide a link (Image Link) that will load when an end user clicks on the image.
- Show Head Navigation and Show Navigation Tree: These both have the same set of radio boxes, Default, Enable, and Disable. They determine whether the main navigation (Show Head Navigation) and the context-sensitive navigation tree (Show Navigation Tree) will be displayed. If these are set to Default, then the Template one settings for the site (or microsite) will be used.
- Element Left and Element Right: These are used to add other files as content in the left and right sides of the page.
- Center Layout and Right Layout: These can be used to add dynamically generated lists to the center and right-hand sides of the page. **Structured Content** types (events and news, for example) are used to populate the list. For example, if you have several event items (one of the built-in Structured Content subtypes), then you can display the latest five in the right-hand column by selecting List of five latest events from the Right Layout drop-down list.
- Configuration path for templates: This determines which Template one configuration should be used for the page. If this is blank (the default), OpenCms will search the parent folder, then the grandparent folder, and so on, until it finds a configuration or arrives at the site folder (in which case it uses the site configuration).

Once you have set the desired properties, click Finish to save the properties and return to the explorer view.

Editing a File

You should now be back in the explorer view in the `playground` folder. You will see one file in the directory—`index.html`. The explorer view looks something like this:



The first icon indicates what type of file this is (page—an HTML document that uses a template). The lock icon indicates that the file is locked—if the lock is open, you own it, and if it is closed, another user owns it. The little red flag indicates that this resource has

not yet been published. This means this file does not exist (yet) in the Online project, and visitors to the site will not see (or have links to) this document.

If you click on the name of the file (`index.html`), OpenCms will pop up a preview window, into which it will load the contents of `index.html`. In this case, we have not yet created content for this file, so the preview will just show the header, footer, and navigation.

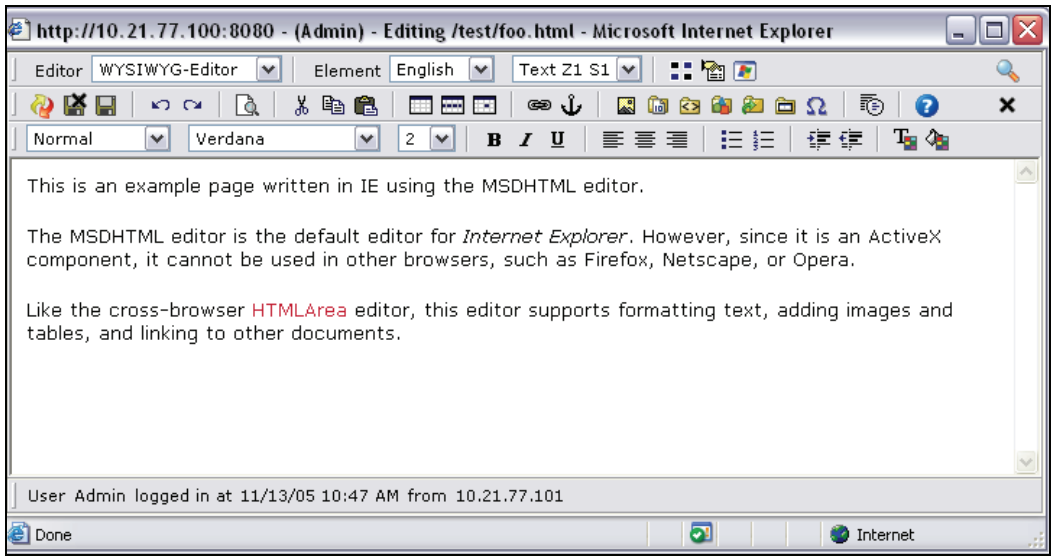
To edit the file from the explorer view, left-click on the page icon and choose Edit page from the menu. This will open the file for editing in one of the three WYSIWYG editors.

The WYSIWYG Editors

OpenCms 6 supports three WYSIWYG editors. The default editor (new in OpenCms 6.2, though it was available as a module in OpenCms 6) is the FCKEditor. The other two are the Microsoft DHTML editor and the HTMLArea editor.

The Microsoft DHTML Editor

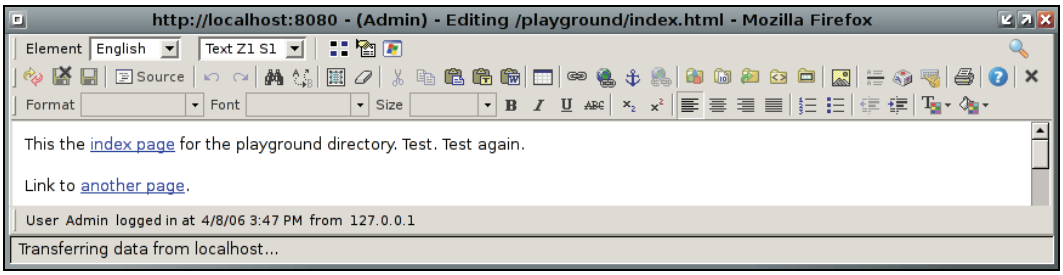
As in previous versions of OpenCms, the **Microsoft DHTML editor** (which is an ActiveX control) is available with OpenCms, though it only works with Internet Explorer. In earlier versions of OpenCms (version 5 and earlier), this was the default editor. This is no longer the case.



This editor is not as feature-rich as the FCKEditor, and it may not be included in future versions of OpenCms.

FCKEditor

The **FCKEditor editor** is the new default editor and runs in both Internet Explorer and Mozilla-based browsers (including Netscape 7, Firefox, and the Mozilla Suite). It is a standards-compliant open source editor. The FCKEditor editor has been customized specifically to work with OpenCms, and the OpenCms developers recommend using it.



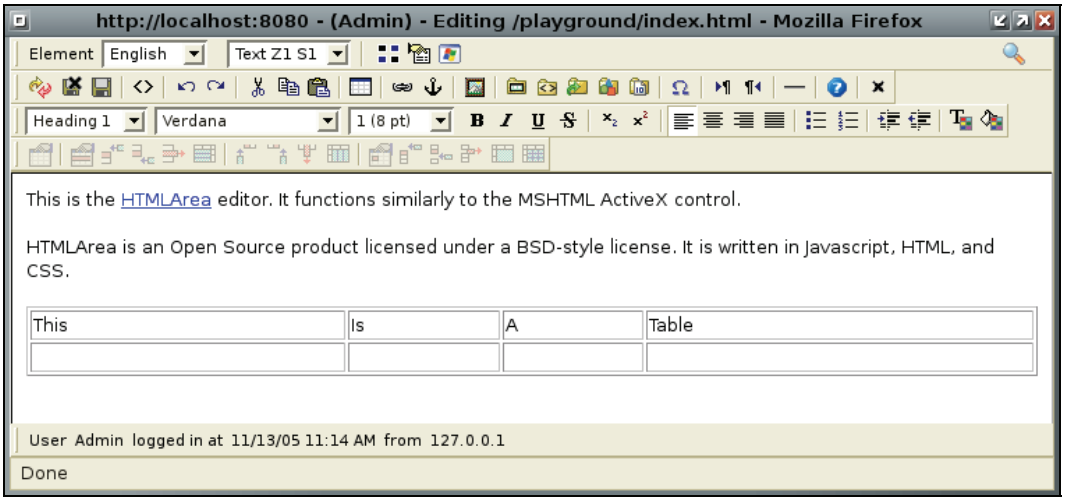
> FCKEditor is an open-source online HTML editor implemented in JavaScript, HTML, and CSS. It is freely available, customizable, and easy to integrate into existing web applications. Should you be interested in using it in other web applications, you can download it from `http://www.fckeditor.net`.

The HTMLArea Editor

The third option is the **HTMLArea editor**. It has similar features to FCKEditor and also runs in Mozilla-based browsers. The HTMLArea editor is included for backward compatibility, but it may be removed in future versions of OpenCms.

> HTMLArea is an open source online text editor implemented in JavaScript, HTML, and CSS. It is freely available, customizable, and easy to integrate into a web application. Should you be interested in using it in other web applications, you can download the stand-alone version from `http://www.dynarch.com`.

Both editors provide the same functionality, and the OpenCms developers have given the two almost identical user interfaces, which are similar to a standard word processor or HTML editing tool:



Most of the buttons are self-explanatory, but I will explain a few of the less intuitive ones here.

The first drop-down list, Element, lets you select which language you are going to write in. By default, only the English and German language packs are installed, though there are many others that can be added. (See Chapter 4.)

You can write content for each language. That is, you are not choosing one language per page (as was the case in OpenCms 5.0) but can write one page in multiple languages. Select English and type in some English text. Save. Select German and type in some German text. Save. Now you have a page available in two languages. Based on the browser's language settings, the end user will be served whichever language they prefer!

The second drop-down list and the button with four small blue and black squares are both related to the text body. Just as a document can contain multiple languages in one page, it can also contain multiple body sections in one page. (In fact, it can contain multiple body parts in multiple languages.)

For example, if you want two sections of text on the same page, one on the right side and one on the left, click the button with four squares (according to the button's tooltip, this is the Elements button). In addition to the already checked Text Z1 S1 box, check the Text Z1 S2 box. This will add a second column. If instead of two columns you should want two rows, check the Text Z2 S1 box instead.

> Z indicates the row number, and S indicates the column number. I would speculate that this has to do with the German terms for rows and columns: *die Zeile* and *die Spalte*. Or perhaps that is but happy coincidence.

The next button, Properties, provides a way to get to the properties screen discussed in the section above.

The last button in the top row, Clean Up HTML, is useful when you paste text from external HTML generators, some of which use non-standard or overly bloated HTML code. These generators include unwanted HTML tags and CSS (Cascading Style Sheets) information. You can eliminate the HTML bloat by highlighting the desired text and clicking this button.

Another set of particularly useful buttons are the five gallery icons, which are small folder icons with various objects on top:

From left to right, these icons do the following:

1. The first provides access to the Tables gallery, where you can browse through the table formats stored in OpenCms. When you locate one that you like, you can insert it into the document with the click of a button.
2. The second provides access to the HTML gallery, where you will find HTML snippets stored in OpenCms.
3. The third provides access to the External Links gallery, which has links to external resources.
4. The fourth may be the one that is most often used and provides access to the Image galleries. You can browse all of the image galleries in OpenCms to find the image or images that you want.
5. The fifth button provides access to the Downloads gallery. You can browse this gallery for downloadable packages (such as PDF files or executables). When you choose a download file, a link to that file is placed on your page.

Finally, there are a few buttons for handling, saving, and closing the document:

The first of these buttons, composed of a yellow curved arrow above a red curved arrow, is the Publish Directly button. Clicking this will do three things in sequence. First, it will save the document. Then it will exit the editor. Then it will publish the page. Typically, this is not the best way of doing things. You will usually want to save the content, then preview it to make sure everything is correct before you publish it to the Online site (allowing end users access to it).

> You can preview your page directly from the editor by clicking on the Preview button, a magnifying glass icon, in the upper right-hand corner of the editor.

The second button, the disk with a black X on it, is the Save and Exit button.

The Editor drop-down list allows you to switch between editors.

The third icon, the yellow floppy disk, is the Save button. It is a good idea to save periodically.

Finally, on the far right side of the same row of buttons is a black X icon. This is the Exit button. It will exit without saving (though it will ask you if you are sure you want to close before throwing out all your work).

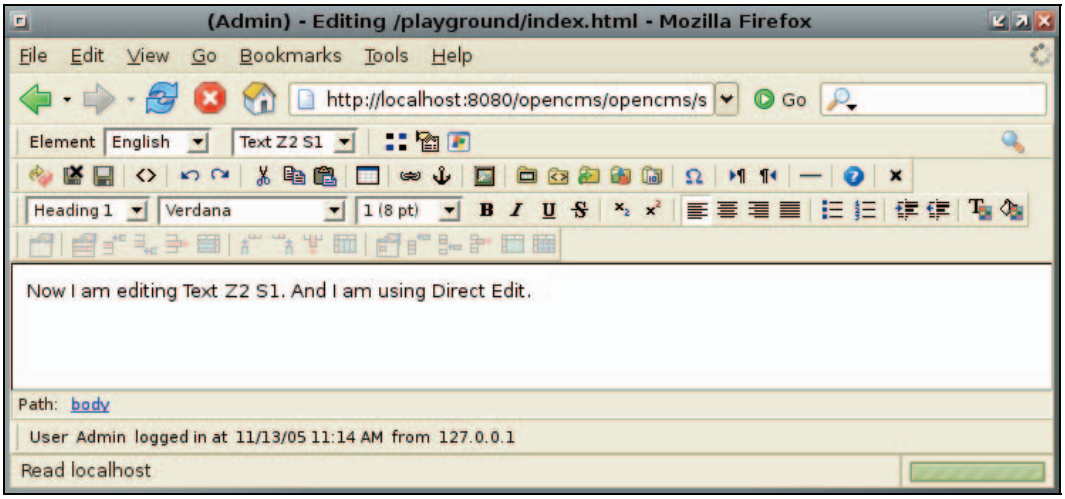
> The disk icon with the black X ( ) often confuses new users, who think that it means 'do not save'. In fact, it is the 'save and exit' button. The plain black X on the far right is the 'exit without saving' button.

Starting the Editor from a Page

In the last section, we navigated through the explorer view to the file we wanted to edit. But you can also start the editor when viewing the page you want to edit. For example, the `playground/index.html` page has some very basic content right now.

The page has two sections, or elements, one in Z1 S1, and one in Z2 S1. You may notice that each element shows up in a gray box that has an OpenCms target icon on the far right.

This only shows up if you are logged in as a Workplace user. What is it? It is the Direct Edit button. Clicking this button will open the element in an editor, where you can edit and save the HTML.



The Direct Edit button can be a time saver since it makes the process of editing and previewing documents much quicker.

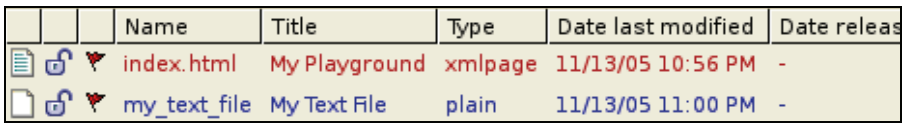
In the next section, we will cover the other editor: the sourcecode editor.

The Sourcecode Editor

The **Sourcecode editor** is usually used for text files and JSP pages, but you can use it for page files as well (though you will have to write HTML by hand).

Once again, our starting point will be the `playground` subfolder of the `/sites/default` folder.

Create a new text file. (Click on the New button and choose Text from the list of available types.) The explorer view should look something like this:



To edit this file with the Sourcecode editor, click on the text file's icon and choose Edit sourcecode from the menu.

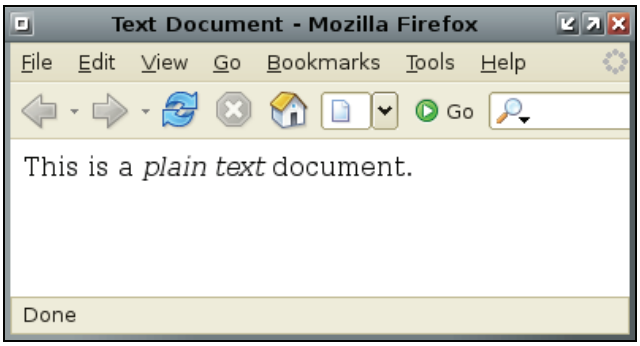
> For a page (HTML) document, clicking Edit sourcecode will load the WYSIWYG editor, but you can edit the source code from by clicking on the Toggle to HTML Source button (the icon is pair of angle brackets: < >).

The Sourcecode editor is a stripped-down no-frills editor. While this editor is basic, it contains all of the tools necessary for developing text resources—both content, which we will cover here, and executable scripts, which we will look at later.

A text document can contain any type of text, including HTML. By default OpenCms serves text files with the MIME type set to `text/html` so that the browser will try to display the document as HTML. Here is an example of a text file:

```
<html>
<head>
<title>Text Document</title>
</head>
<body>
This is a <i>plain text</i> document.
</body>
</html>
```

This is as generic an HTML document as I could devise. After saving and closing the document, the preview in the browser looks like this:



However, as you may have noticed, plain documents do not use templates and cannot take advantage of link checking, galleries, and the other features of page documents.

What we've done here is one use of plain text documents in OpenCms. However, more often plain documents are used to store data for dynamic applications.

The Controlcode Editor

One special case of the sourcecode editor is the **Controlcode editor**. The contents of page documents are stored within a larger XML document. On rare occasions, you may need to edit the XML directly, instead of changing the contents in the WYSIWYG editor. To do this, find the file in the explorer view, click the file's icon, and select Edit Controlcode.

A very simple file containing only one line of content, "This is a simple file.", will look like this in the Controlcode editor:

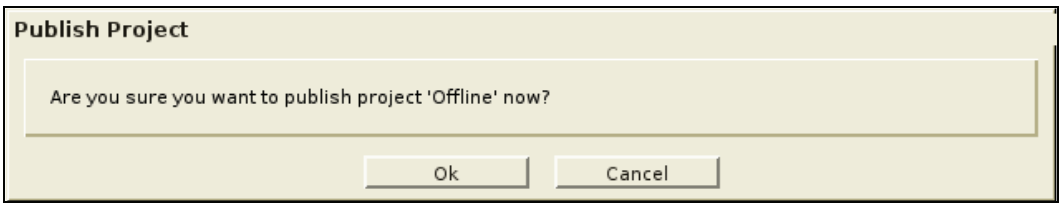
```
<?xml version="1.0" encoding="UTF-8"?>

<pages xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation=
      "http://www.opencms.org/dtd/6.0/xmlpage.xsd">
  <page language="de">
    <element name="text1">
      <links/>
      <content/>
    </element>
  </page>
  <page language="en">
    <element name="text1">
      <links/>
      <content><![CDATA[This is a simple file.]]></content>
    </element>
  </page>
</pages>
```

The line containing the content has been highlighted in the above example. The rest of this file is XML describing the content. As languages and elements are added to the document, this file gets larger and more complex.

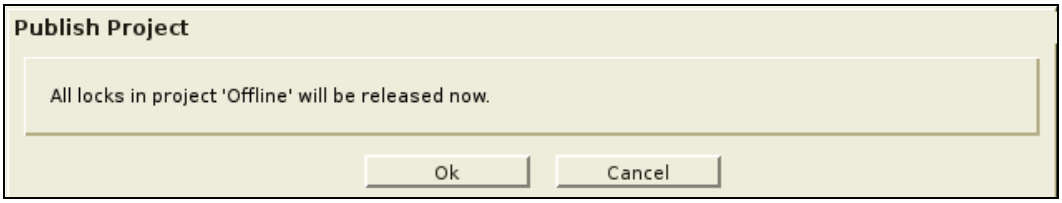
In the next section, we will walk though the process of publishing our new documents to the Online project, thereby making them available to those who visit our site.

Publishing Your Changes

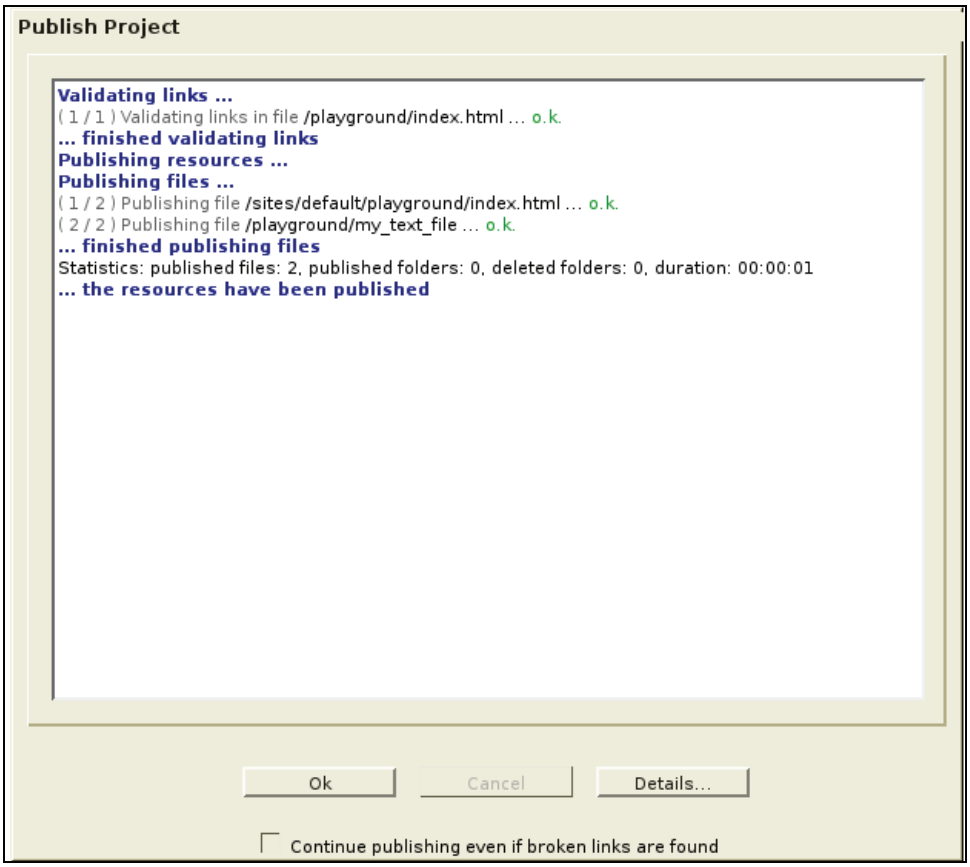
There are two ways to publish the changes we've made. One is to publish the entire project (the Offline project, in this case). This will copy all of the changes in the current project to the Online project. To do this, simply press the Publish icon in the main toolbar. You will be asked to confirm:



If any resources are locked, you will be prompted to allow OpenCms to release the locks. (You can turn this behavior off in the Preferences panel.) Choosing Ok will automatically remove all the locks and then continue the publishing process:



After this, OpenCms will conduct a series of checks, including making sure all siblings are published, checking for broken links, and making sure that there are no broken dependencies.



If the checks turn out OK, then all the modified or newly created files will be moved to the Online project, where they can be accessed from the public side of the site.

You may want to publish only a small set of changed files rather than the whole project. To do this, left-click on the icon next to the resource you wish to publish and select Publish directly. Note that this option will be marked inactive if the file has not been changed since the last time it was published. Try publishing the `playground` directory directly. That will publish the folder as well as all of its contents. After asking you to confirm the publish request, it will display progress information as it publishes all the resources to the Online project.

> If you publish a locked file, the lock will be removed before the file is published. This is different than the behavior in OpenCms 5, where a locked file could not be published.

During the course of a direct publish, OpenCms runs the same checks as it does during a full publish (as discussed above). For this reason, publishing only a few pages may still take a long time.

Which Method of Publishing Should be Used?

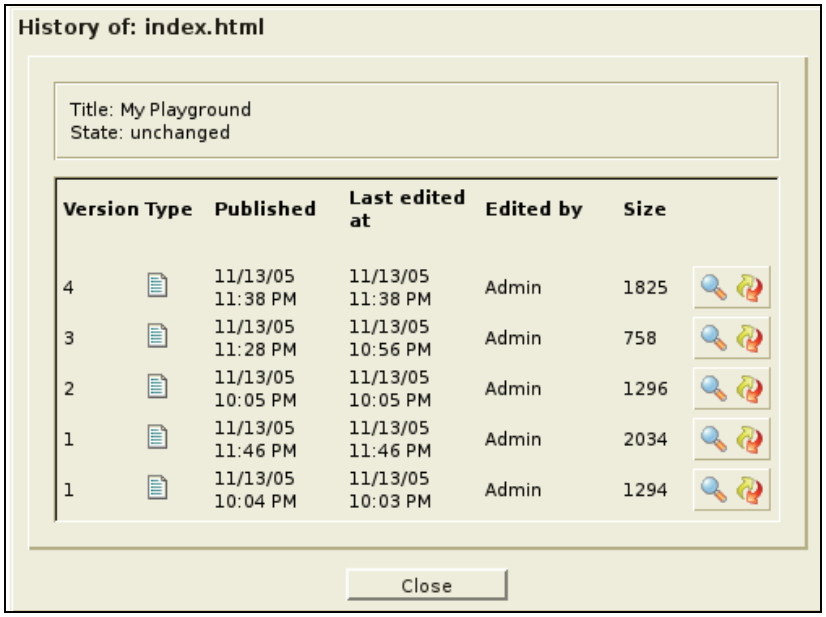
So how do you determine which sort of publishing method to use, and when? Generally, site-wide updates (by their very nature) tend to involve publishing a large number of files all at once. Publishing the entire project is often the best way of pushing these sorts of updates to the Online project. But it is much safer to publish individual pages directly, as there is less risk of accidentally publishing material that is not ready to be seen by the viewing public. Most pieces of content are therefore published directly, rather than as part of a project-wide publish.

There are more advanced tools that help in complex publishing scenarios, including custom projects and sibling relationships.

Versioning

Now that the new resources are published, OpenCms will keep version information on these files. To view a file's history from the explorer view, left-click the file's icon and select History from the menu. Every change made to the file is recorded. If a mistake is made, we can open the History menu, select a version, and roll back to that version of the file.

To do this, go back to the Offline project, and lock and edit the `/playground/index.html` file. Once you have changed a line or two and saved it, left-click on the page icon for `index.html` and choose History. You will see an entry for the last published version.



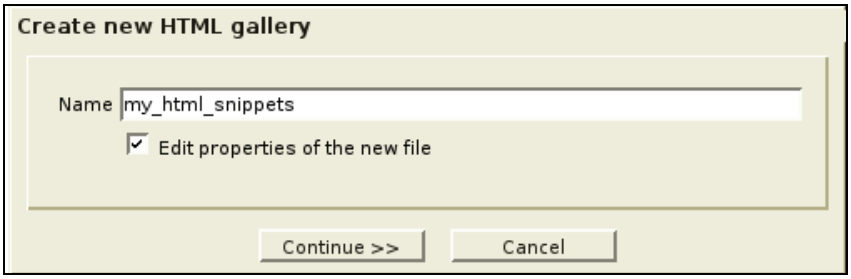
There are two buttons for each entry in this list. The magnifying glass, View version, will display that particular version in a browser window. The second button, the familiar publishing button, will publish that particular revision of the document, thereby replacing the current version with this older archived version.

As you work with a file, making changes, publishing, and then changing it again, OpenCms will build up a version history for that file. Using the history, you will be able to track the file's changes for every version since it was first published. In Chapter 4, we will look at some administration tools that work with file history.

Galleries

In a previous section we looked briefly at the Extended Folder type. In this section, we will use the Extended Folder type to create a new gallery.

To start with something simple, let's create an HTML gallery. To create a gallery, click the New button and select the Extended Folder type from the list of types. Click Continue to choose the gallery type. This will bring up the Create new HTML gallery dialog. As with a folder, the Name field determines the name of the folder (in the file system and URL). In the screenshot overleaf, I've named my gallery `my_html_snippets`.



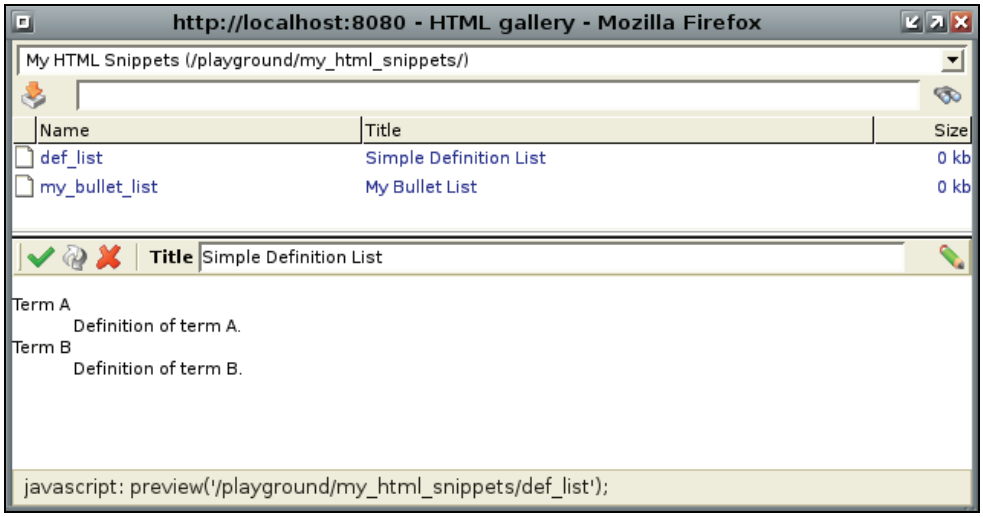
Since we left the Edit properties of the new file box checked, we will get the Properties dialog. Title is the only important property on this dialog (and there are a lot of properties here—this is the generic Properties dialog, and it shows *all* of the properties that could be set, even though most don't apply to our folder).

HTML snippets are stored in plain text documents. Let's create a new HTML snippet for constructing definition lists. First, click the New button and select the Text type. When you hit Continue, you will be prompted to give the file a name. We'll call it `def_list`. The only property we need to set in the Properties dialog is Title, which should be a clear description of what this HTML snippet is for—in this case, "Simple definition list".

At this point, we are inside the new `my_html_snippets` HTML gallery, and we have one file named `def_list`. Click on the file's icon and choose Edit sourcecode from the list. This will open the `def_list` file in the Sourcecode editor. Here is what it looks like:

```
<dl>
  <dt>Term A</dt>
  <dd>Definition of term A.</dd>
  <dt>Term B</dt>
  <dd>Definition of term B.</dd>
</dl>
```

That's all there is to creating an HTML snippet. Now we can use these snippets in the WYSIWYG editor. Browse back to the `playground` folder and edit `index.html`. Click on the HTML Gallery button (the folder with the angle brackets). When the new window opens, you may have to select the appropriate gallery from the drop-down list. Once you've done that, you should see something like this:



(You may notice that I've already added a second snippet for creating bulleted lists.) To insert our definition list snippet in the current page, just click the green check button. Adding a table gallery proceeds in the same way.

Image and Download Galleries

Image galleries and download galleries both require you to upload the content into OpenCms. With the new Upload Applet, this process is much easier than it used to be.

First, create a new extended folder of subtype Image Gallery. (The previous section describes how to do this.) Let's name it `my_images`. In the explorer view, navigate to the new `my_images/` folder. Click on the Upload button to upload a new image from your local drive to the OpenCms VFS.

Uploading a Gallery

By default, OpenCms 6 uses a Java applet to upload files. For this applet to work, you must have a fairly recent Java JRE or JSDK installed (the official requirement is J2SDK 1.4 or greater). The first time the applet loads, it may take a minute or so, but subsequent use of the applet should be much faster. The applet streamlines the process of browsing the local system and then transmitting the files to the server.

Once you have located the image you want to upload, click the Ok button, and the image will be transmitted to the server. When the transmission is complete, the explorer view will return.

Not all browsers support Java applets, but you can also upload files via a web-based form. To access this form, you will need to go to Preferences (in the toolbar) and uncheck Use Upload Applet.

Using a Gallery

Now we can navigate to the `playground/index.html` file and view our new image gallery. Open the index file in the WYSIWYG editor, and click on the Image Gallery button (the folder with the colored shapes). You will probably need to select the appropriate gallery from the drop-down list at the top. Once you have selected it, you should be able to see the newly uploaded images.

Clicking the green check mark Insert button will add the image to your current page.

There will be more about galleries in Chapter 4.

Summary

In this chapter we looked at the explorer view of the OpenCms Workplace. We learned how to navigate through the Virtual File System (VFS) and how to create and edit the basic forms of content in OpenCms.

In the next chapter, we will turn from these editorial tasks to managerial and administrative tasks. The next chapter we will look at administration and project administration.

4

OpenCms Administration

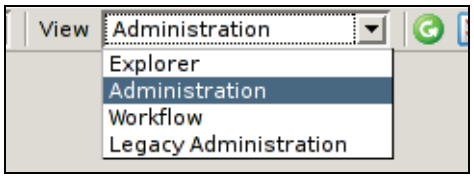
In the previous chapter, we used the explorer view of the OpenCms Workplace to create, edit, and publish content. In this chapter, we will examine the administration view. The administration view provides tools for managing many aspects of your OpenCms server. The workflow view provides project management capabilities and task tracking for OpenCms projects. This chapter will cover:

- The administration view and the legacy administration view
- Managing projects
- Accounts, users, webusers, and groups
- Managing the database
- File history settings and clearing the file history
- Link validation
- Managing galleries
- Managing the indexes used for searches
- Scheduled jobs
- Flex cache administration
- Content tools
- Workplace tools
- Managing modules

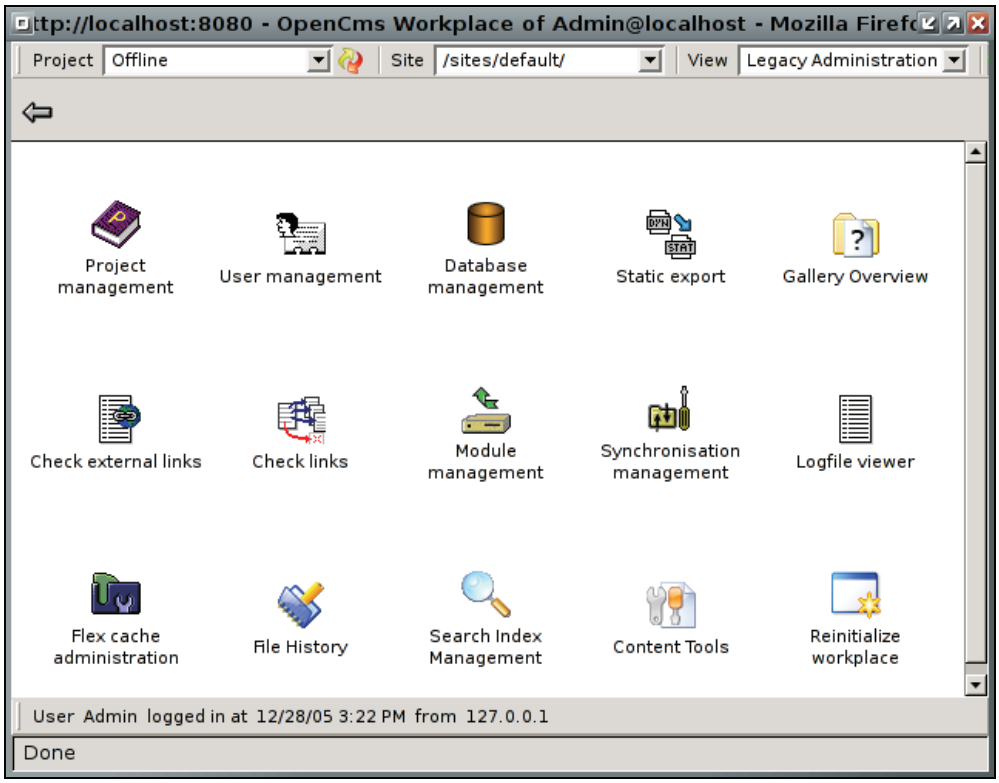
The Administration View

The **administration view** provides access to the tools you will need to administer the OpenCms server. We will look at these tools in this chapter.

By default, the OpenCms Workplace loads the explorer view. To switch from this to the administration view, select Administration from the View drop-down list in the toolbar.

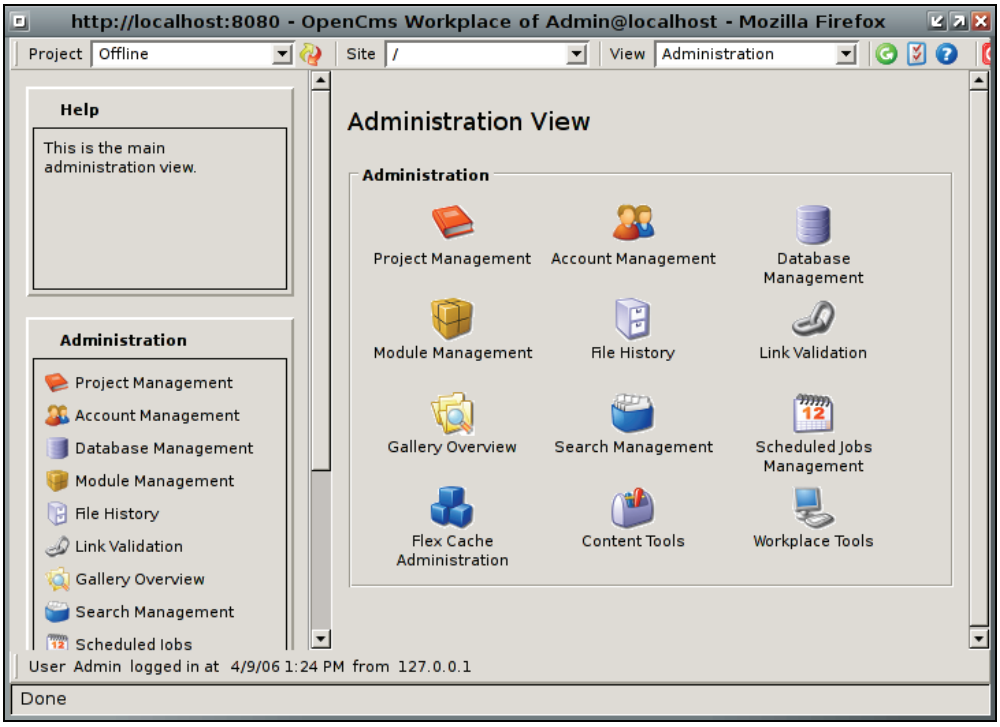


In the View drop-down list, there is also an item called Legacy Administration. Choosing this item will load an alternative version of the administration view that uses an icon layout scheme similar to Windows 3.1.



The **legacy administration view** shares the look and feel of the OpenCms 5.0x administration view. While it uses less JavaScript than the OpenCms 6.0x administration view, it is slower and may not be supported in subsequent releases of OpenCms. In this chapter, we will focus on the new administration view, but everything covered here is also present in the legacy administration view.

The administration view has two panes. The pane on the left contains help text and the main navigation for the administration view. All the administration tools will load in the right pane, though initially the right pane contains navigation icons that match those in the navigation menu in the left pane.



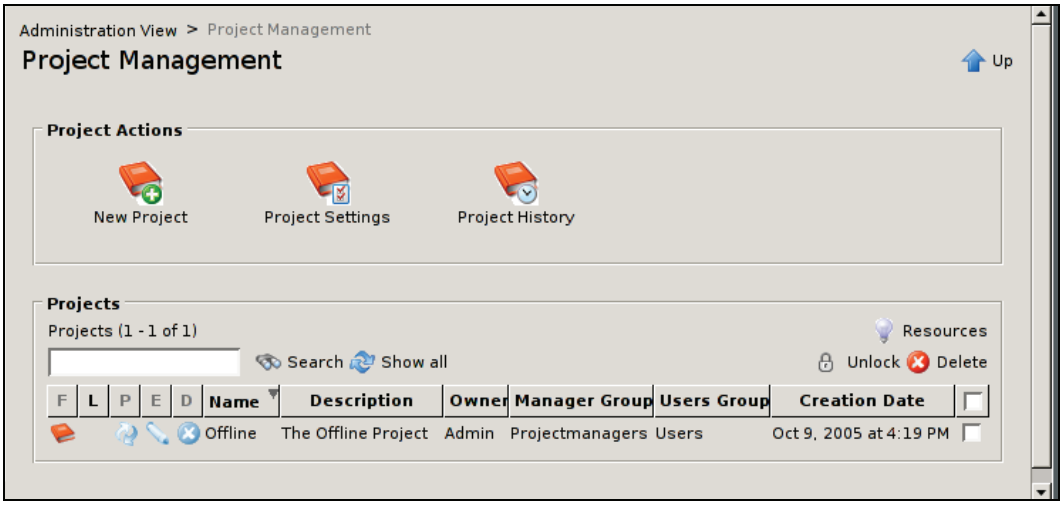
When the mouse cursor moves over a navigation item (either on the menu in the left pane or over one of the icons in the right pane), a brief summary is displayed in the Help box in the left pane. You can collapse the help section (thereby hiding the help text) by clicking on the word Help. Likewise, you can hide the navigation in the left-hand pane by clicking on the word Administration.

Project Management

The first item in the administration menu is Project Management. This is a category containing all of the administration tools that have to do with configuring OpenCms projects. A project, in OpenCms, is an organizational unit. A project provides access to a particular collection of documents, and is often used to provide some group of users with access to just a particular section of a site. As we saw in Chapter 3, by default, OpenCms includes two projects—the Offline project, which contains the current versions of all of

the files in the repository, and the Online project, which contains all of the published versions of the contents in the CMS. Both of these projects serve a specific purpose in OpenCms, and are thus indispensable. However, you may find it convenient to create additional projects to help organize your content. Custom projects always contain a subset of the files in the Offline project.

Click on the Project Management icon to view the project management tools. This will load the Project Management screen in the right pane.



Along the top of this page there are a few common elements that will be repeated in all of the administration tools. The breadcrumb menu, Administration View | Project Management, provides a representation of the position of the current tool in the administration view hierarchy. In this case, the Project Management tools are direct children of the main menu of tools (here called Administration View). You can click on items in the breadcrumb menu to take you to that specific screen. Clicking Administration View (which is the first item in every one of these breadcrumb menus) will take you to the main menu for the administration view. Beneath the breadcrumb menu, is the title of the current screen—Project Management.

To the right of the breadcrumb menu is a blue up arrow, labeled Up, which will take you one level up the administration view hierarchy. It functions identically to clicking the second-to-last item in the breadcrumb menu. In our current case, the Up button links to the main menu of tools. The breadcrumb menu and the arrow are available on all screens within the administration view.

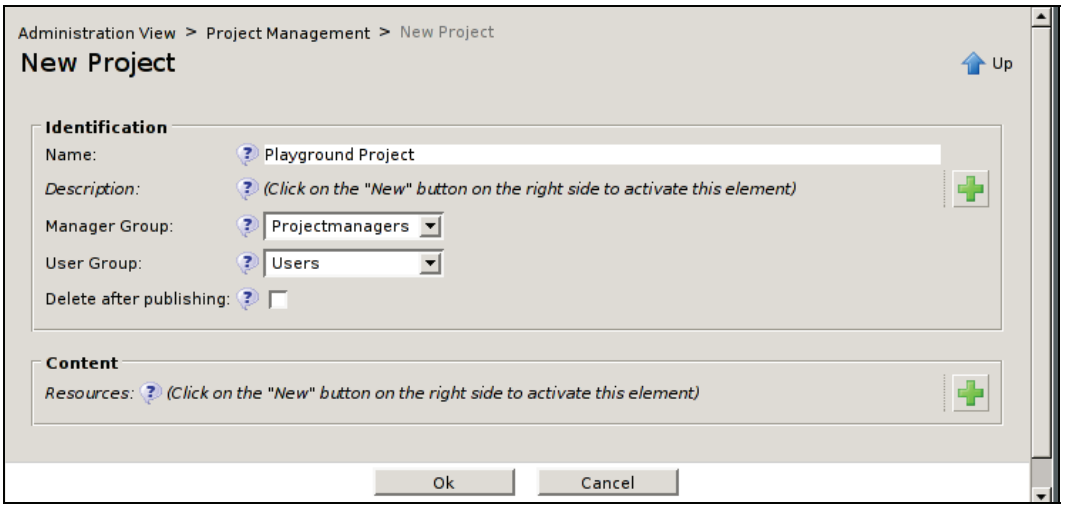
At this point, it is time to look at the project management tools on this page. There are two boxes on this page. The first, Project Actions, contains three icons, each representing a particular action. The second box, Projects, displays a detailed list of all of the offline projects (including the project named Offline). The special Online project, which contains only published content, does not show up here and cannot be manipulated by the Project Management tools.

We will create a new project, which will contain a subset of the documents contained in the Offline project.

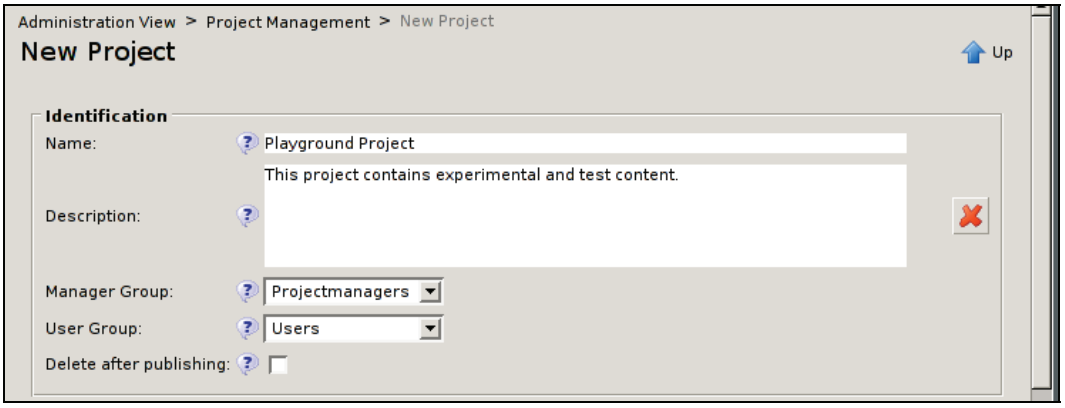
Creating a New Project

We want a project that will include all of the files in the `/playground` folder we created in Chapter 3.

Click on the New Project icon in the Project Actions section. You will be prompted to define the new project.



First, we need to provide some basic information about our project in the Identification section. Name is a human-readable name (spaces are allowed). I have named this project Playground Project to indicate that it is correlated with the `/playground` folder in the Virtual File System (VFS), but in some cases, the name does not provide sufficient information to describe the purpose of the project. If you click the large green plus sign, +, icon next to the Description label, a text area will appear, and you can fill in a more adequate description.



The Manager Group drop-down list should be set to the group that is responsible for overseeing and publishing this project. Later on in this chapter, we will create custom groups, but by default, the available groups are Administrators and Projectmanagers. The Projectmanagers group is a safe selection (since all administrators are also project managers).

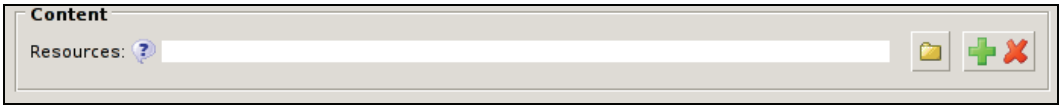
The User Group drop-down list determines which group of users will be using this project within the Workplace. This does not affect who will be able to see the files when they are published, but it indicates who will be editing content in this project. The default group, Users, is the group composed of all the users who can access the OpenCms Workplace. This includes all editors, project managers, and administrators. Later in the chapter, we will look at creating custom groups, but for now, leave Users selected.

If the Delete after publishing box is checked, then as soon as the project is published, the *project* will be deleted, but the *contents* of the project will still be available. None of the files in the project will be deleted (or modified in any way) when the project is deleted, and you will still be able to edit them in the Offline project.

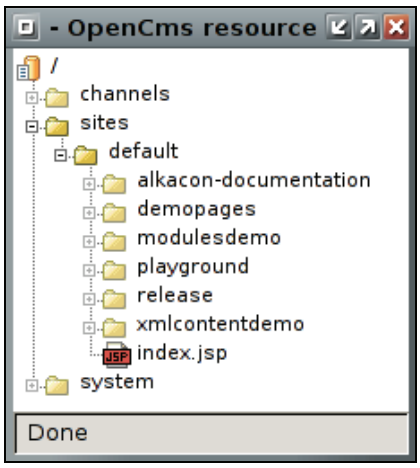
Deleting a project after publishing is useful for short-term projects. You can create a project for a specific set of articles and then assign editors to work on this project until it is complete. When the project is published, all the articles will be copied to the Online project, where they can be accessed by the viewing public, and the project itself will be deleted automatically. There is no need to go back and manually clean up old projects.

But, since we may be publishing our Playground project a couple of times, we will leave this box unchecked for now.

Now we need to identify which content will be part of this project. To select content, click the large green plus sign, +, icon in the Content section. This will add a new row containing a text input box and three icons: a folder, another green plus icon, and a red X icon.



Click on the folder icon to browse the VFS for the right folder:



Remember, by default all content goes in the `/sites/default` folder.

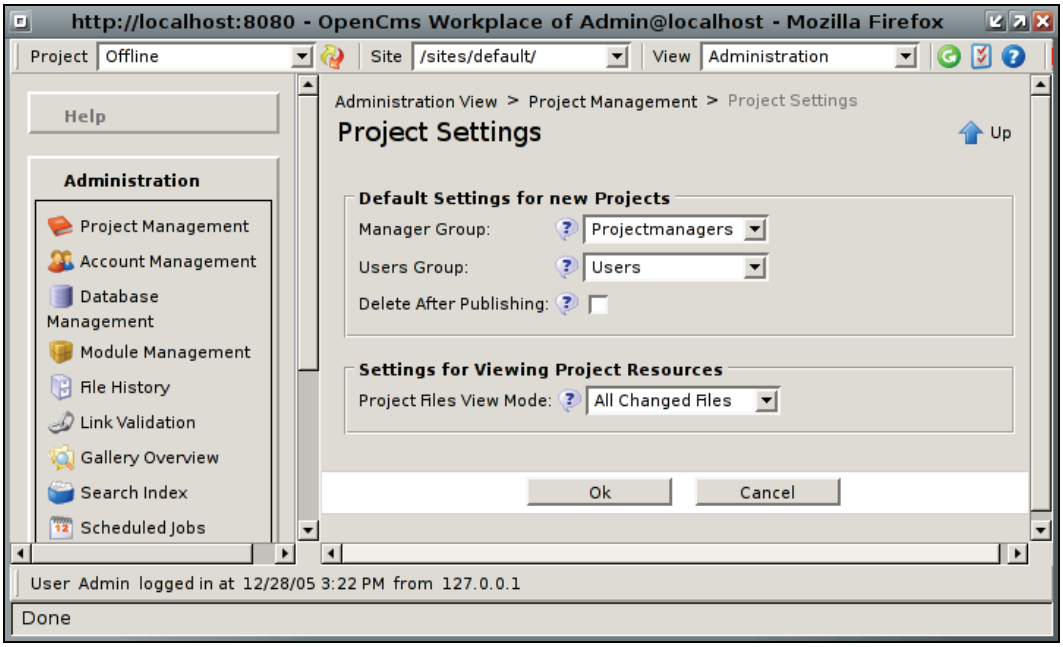
In the navigation tree, clicking on the plus/minus (+/-) boxes will expand or collapse a folder. Clicking on the name of a folder will add the folder to the project text input box and expand the folder. For our project, select `/sites/default/playground`.

To add additional directories, click the big green plus sign, `+`. A new row will appear. You can remove unwanted rows by clicking the red X button.

Now that the form is complete, click the Ok button to create the project. This will take us back to the Project Management screen, and the new Playground Project will show up in the list of projects. Before looking in detail at our newly created project, we will briefly look at the other two items in the Project Actions menu.

Project Settings

The second item in the Project Actions menu of the Project Management screen is Project Settings. The Project Settings screen allows you to configure some of the defaults for project management tools.

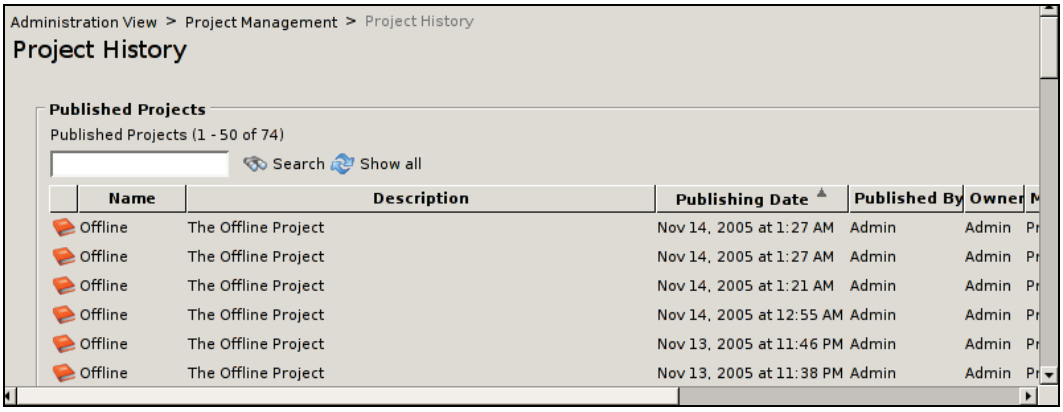


In the Default Settings for new Projects section, you can specify which groups will be selected by default for Manager Group and Users Group. (See the section above for descriptions of these two fields). If the Delete After Publishing box is checked, all new projects will have this box checked by default.

In the Settings for Viewing Project Resources section, you can set which types of file will be displayed by default when viewing a project's files. (To view a project's files, go to the Project Management menu and click on a project title in the Projects file list.) You may choose to have only new files, only deleted files or only modified files displayed there. The default option, All Changed Files, will display files that have been added, modified, or deleted since the last time. Usually, this is the option you want.

Project History

The third item in the Project Actions menu is Project History. The Project History screen displays the publishing history of all offline projects (that is, all projects except for the Online project) and provides quick data on which projects were published and when:



The Project List

We will now take a look at the Projects list in the lower half of the Project Management screen.

A row exists for each project. At the beginning of each row are five icons. When the book icon is clicked, the View Files for Project screen is loaded. This screen has a list of all the files in the project that have been changed since the project was last published. Clicking on the project's name (in the Name column of the table) will also display the View Files for Project screen.

The second column displays information about locks for the project. If the resources for the project are locked, then a lock icon will be displayed here. For example, if I go to the explorer view and lock the `/playground` folder, a lock icon will show up in this second column.

Note that while the project is locked, the publish icon is marked as inactive. A project must be unlocked before it can be published. Clicking the lock icon will unlock the locked resources. You can also unlock one or more projects at a time by checking the box at the end of the desired row or rows and then clicking the icon labeled Unlock located at the upper-right corner of the Projects list.

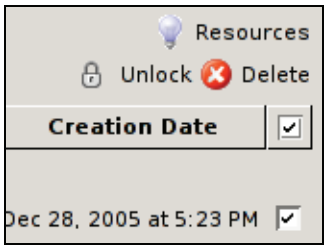
The third icon, a circle made of two blue arrows, is the publishing icon. Clicking it will publish the project. All of the files listed in the View Files for Project screen will be published in the Online project.

The pencil icon, the fourth on the list, is for editing the project information. Clicking it will bring up a screen similar to the New Project dialog. Most of the information entered when the project was initially created can be edited here.

The fifth icon is a blue and white X. Clicking this will delete the project.

The next six columns in the table—Name, Description, Owner, Manager Group, User Group, and Creation Date—provide information about the project. Most of these fields can be edited by clicking on the pencil icon. Finally, at the end of each row there is a checkbox. These checkboxes are used to select rows for bulk tasks.

There are three bulk tasks, two of which are performed on rows that have been checked. Each of these three actions can be performed by clicking on the appropriate icon in the upper-right corner of the Projects list.



If the Resources icon (a light bulb) is clicked, then information about which content is associated with this project will be displayed in the table.

Clicking the light bulb icon again will hide the information. This task is performed on all rows, not just the checked rows.

> Buttons with the light bulb icon will, when clicked, display information about all of the rows in the table. In this case, clicking the Resources icon will show and hide resource information for *all* projects, regardless of which ones are selected.

The icon labeled Unlock is used for unlocking all of the checked projects. This provides a quick way of unlocking numerous projects. Likewise, the Delete icon deletes all the checked projects.

Accounts

Until now, we've done all of our work with only a few built-in users and groups. Obviously, having only a few generic user accounts and groups would be an inconvenience, to say the least, in a real-life situation. The Account Management screen provides tools for creating and managing users and groups.

There are three buttons in the Account Tools section of the Account Management screen: User Management, Group Management, and Webuser Management. We will look at each of these in turn. But first, I will provide a brief overview of how users, groups, and webusers are used by OpenCms.

Users, Webusers, and Groups

The term **user** refers to user accounts in OpenCms that can access the OpenCms Workplace. A user has a login and password. Writers, editors, managers, system administrators, and OpenCms developers are all users.

There are three built-in users. The first is Admin, which we are already familiar with. This is the administration user for OpenCms. The second built-in user is Export. The Export user is used by OpenCms to control exporting content from the OpenCms VFS into the server's file system. While you will rarely (if ever) do anything with this user, you should not delete it or disable it, as OpenCms internals rely on it.

Finally, there is the Guest user. Anyone who is not logged in is treated as the Guest user. The Guest user does not have any special privileges in OpenCms, nor can it log in (since it has no password). The Guest user cannot access the Workplace. However, it can view published pages with public permissions.

It is often desirable to have site visitors who must login in order to see certain "members only" content, but who cannot access the OpenCms Workplace. OpenCms treats these visitors as a special type of user: **webusers**. Webusers have logins and passwords (and other membership data), but still cannot access the OpenCms Workplace. Using OpenCms's access control mechanisms, you can configure certain sections of the published website to be accessible only to logged in webusers.

> Webusers are a special type of user, and they share some of the features of a regular user account. For that reason, you cannot be logged in simultaneously (in the same browser) as both a user and a webuser.

By default, no webusers are configured. Furthermore, simply creating webusers will not automatically give you all of the features of membership-driven site. You will need to set up members-only areas of the site and also create a process for handling webusers. If you want visitors to create their own accounts, you will also have to create the necessary forms for that (which may involved some JSP programming).

Users and webusers can belong to **groups**. A group is an organizational unit used to provide additional classification and functionality to a collection of users. For example, a content development team may have its own group, and each of the editors and writers that are part of that team will be in that group. Permissions to the files for which this team is responsible may then be set according to group, rather than by individual user ID. Access to edit or publish that team's content can be restricted to all users who are part of that team's group.

By default, there are four groups:

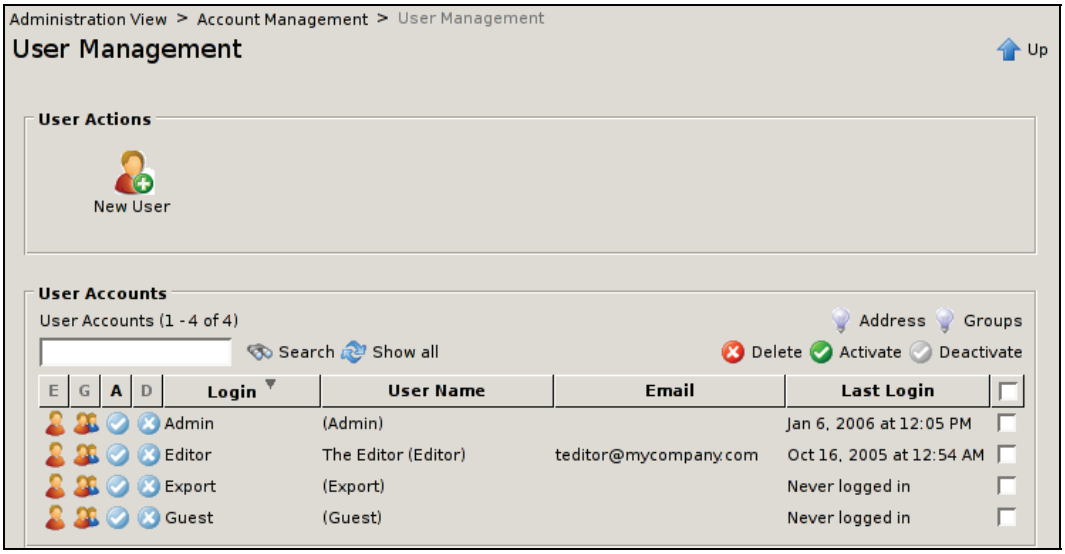
1. **Administrators**: This group is for users that are responsible for overseeing the operation of the OpenCms server. By default, the Admin user is a member of this group.
2. **Guests**: This group is for users who do not log in to the OpenCms workplace. By default, both the Guest and Export users are members of this group.
3. **Projectmanagers**: This group is reserved for users who will act as project managers in OpenCms. They will be responsible, for example, for publishing projects, creating new tasks, and overseeing the creation of new content. By default, there are no users in this group.

4. **Users**: This group is for all non-administrator users that use the Workplace (members of the group Projectmanagers are automatically members of the Users group, too).

Now that we have a basic idea of the role each of these three concepts play in OpenCms, it is time to look at the administration tools in the Account Management screen.

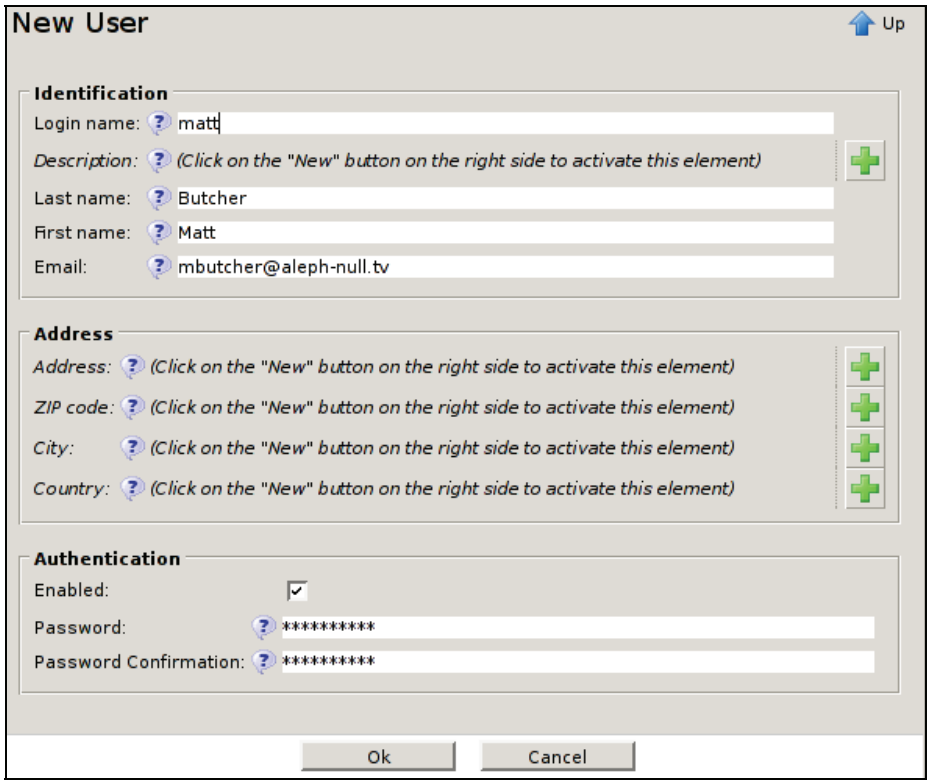
User Management

The first button in the Account Tools section of the Account Management screen is User Management. Clicking on this button will enable you to view, add, modify, and delete users.



At the top of the User Management screen, there is one button, New User. The bottom portion of the screen is composed of the User Accounts list, which contains basic information about all the users in this instance of OpenCms.

Clicking on the New User button will load a form for creating a new user.



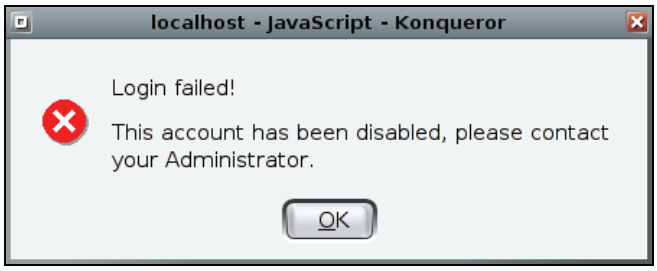
As we have seen in previous forms, the labels for optional fields are in *italics* and to add information to these fields, you will first have to click on the green plus, +, sign. All other fields are mandatory. The Login name text field is for the user ID that the user will type when logging in. The login name cannot have spaces or special characters. The Last name, First name, and Email text fields are all required and are used by OpenCms in a number of places. For example, the email address is used in workflow to send notifications to a user about the state of her or his assigned content.

All the information in the Address section is optional, but it can be useful where users are geographically dispersed.

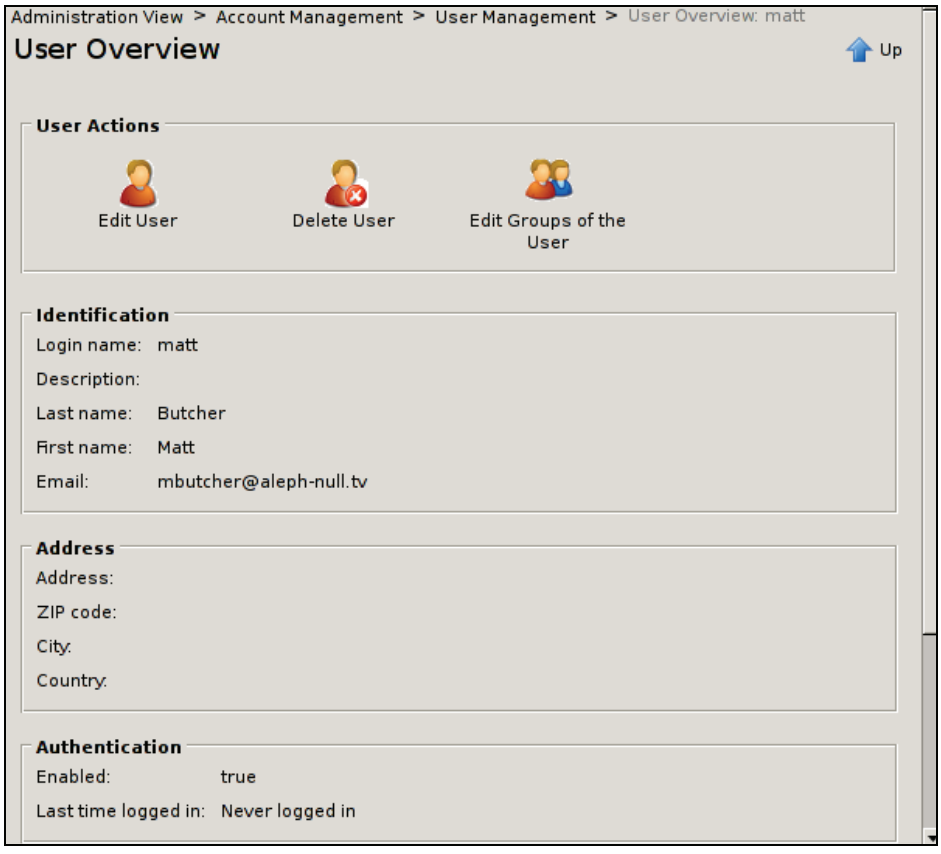
In the Authentication section at the bottom, you can specify whether this new user is enabled. If the Enabled checkbox is checked and a value is set in the Password and Password Confirmation text fields, then the new user will be able to log in. However, if the Enabled checkbox is unchecked, the user will not be allowed to log in to OpenCms.

This provides OpenCms administrators with the ability to disable certain accounts (either temporarily or permanently) without having to delete the account altogether.

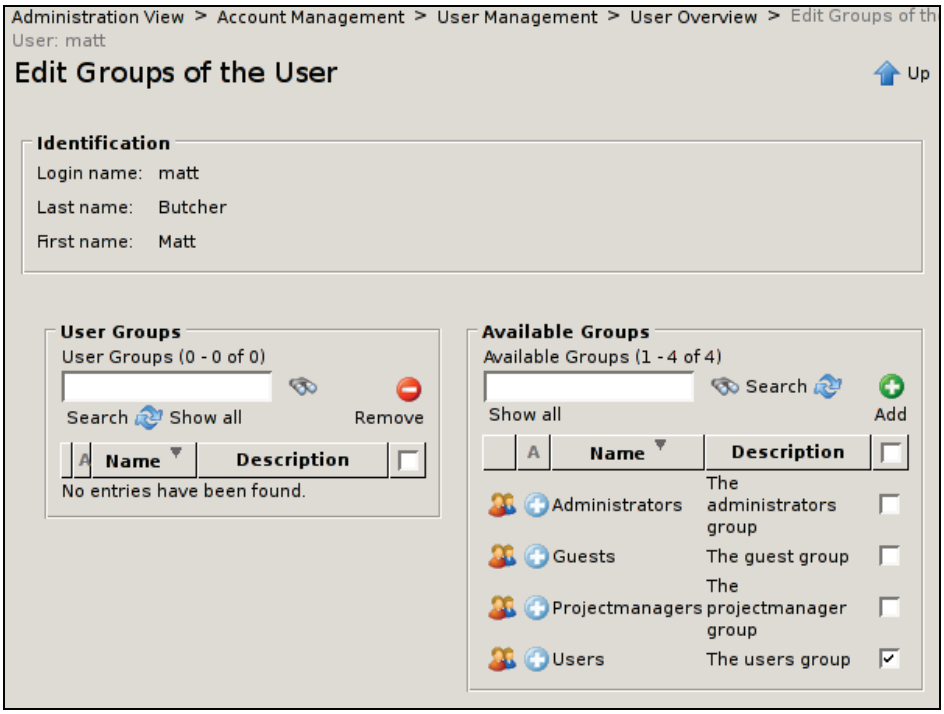
If a user whose account has been disabled attempts to log in, she or he will be notified that the account has been disabled and will be instructed to contact the administrator.



Clicking on the Ok button at the bottom of the screenshot shown on the previous page will check the information you entered, create the user, and load the User Overview screen.



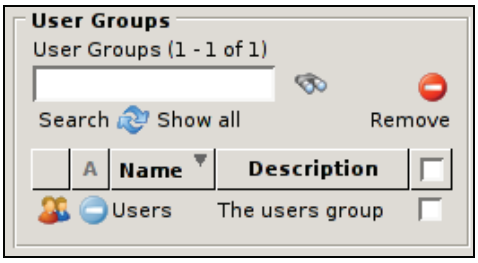
This screen displays detailed information about the user (including, if you scroll to the bottom, information about the groups to which this user belongs). There are three buttons in the User Actions section at the top of the screen. The Edit User button loads the editing screen, which is identical to the New User screen examined above. The Delete User screen will delete the user from OpenCms (prompting you for confirmation first) and return you to the main Account Management screen. Clicking the Edit Groups of the User button will bring up a screen for assigning the user to one or more groups.



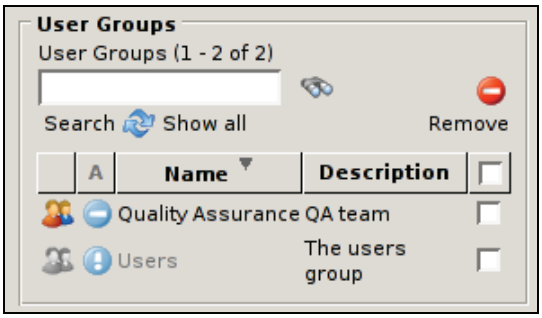
At the top, there is the Identification section for basic information about the current user (matt in this case). On the lower left, the User Groups section displays information about which groups the user is already a member of. By default, a user does not belong to any groups.

On the lower right, the Available Groups section shows information about all of groups on the system, with each group having its own row. If the first item in the row is an icon of two cartoon people, then the user can be added to that group. You can add the user to a group by clicking the blue circle with the white plus sign in the center, or you can add the user to multiple groups by checking all of the desired checkboxes in the last column and clicking the button in the upper-right corner of the box—a green circle containing a white plus sign with the label Add. Usually, all active users should belong to the group Users.

Once the user has been added to a group, the User Groups section on the left will display information about that group. It also has buttons for removing the user from groups.



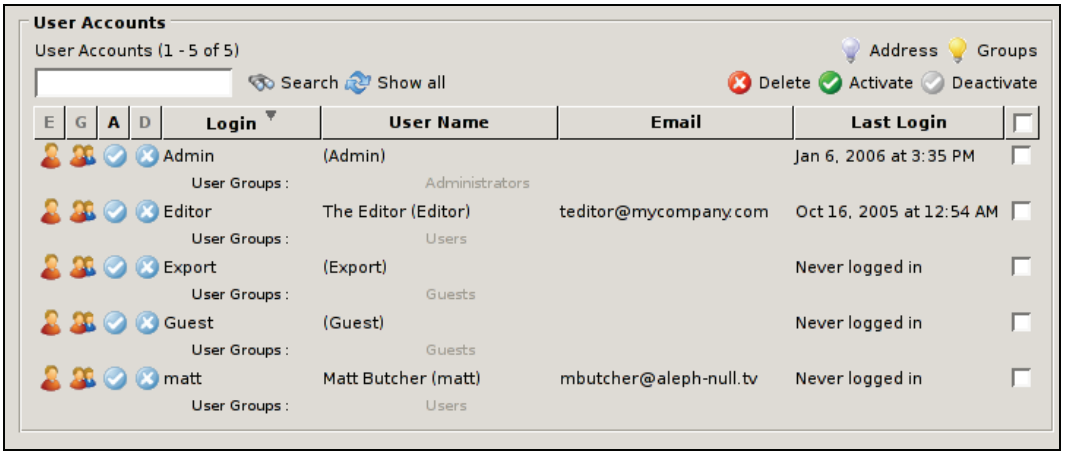
Groups can be arranged in hierarchies (where a group can have a parent group as well as any number of child groups). When a user is assigned to a group, the user also automatically becomes a member of that group's parent group (if it has one). The parent group will also show up in User Groups section, though the user cannot be removed from this group separately. For example, consider the case where we have a group called Quality Assurance. (We will see how to create groups shortly.) If the group Quality Assurance is a child of the Users group and our user *matt* is assigned to the Quality Assurance group, then the User Groups section will look like this:



The icon in the first column is grayed out, and that in the second column, the circle button has an exclamation point instead of a minus sign. This indicates that the user cannot be directly removed from this group. But if the user is removed from the Quality Assurance group, she or he will also be removed from the Users group.

Clicking on the User Management breadcrumb at the top of the screen will take us back to the main User Management screen.

At the bottom of the User Management screen, there is a section called the User Accounts section.



In the upper-right corner, there are two buttons, Address and Groups (both of which have light bulb icons). Clicking on these buttons provides additional information about the items in the list. In the image above, Groups (with the yellow light bulb) has been clicked, so that the information about group membership is displayed beneath each user. Clicking the Address button would display the optional address information for each user (if such information exists).

Beneath these two buttons, there are three bulk-update buttons: Delete, Activate, and Deactivate. When one of these buttons is clicked, the associated action will be performed on all of the rows whose checkboxes (in the far right column) are checked. The Delete button removes the selected users from the database. The Activate button will enable a user account. (This is the same as going to the Edit User screen for the user and checking the Enabled checkbox in the Authentication section.) The Deactivate button will prevent a user from logging in. (This is the same as unchecking the Enabled checkbox on the Edit User screen.)

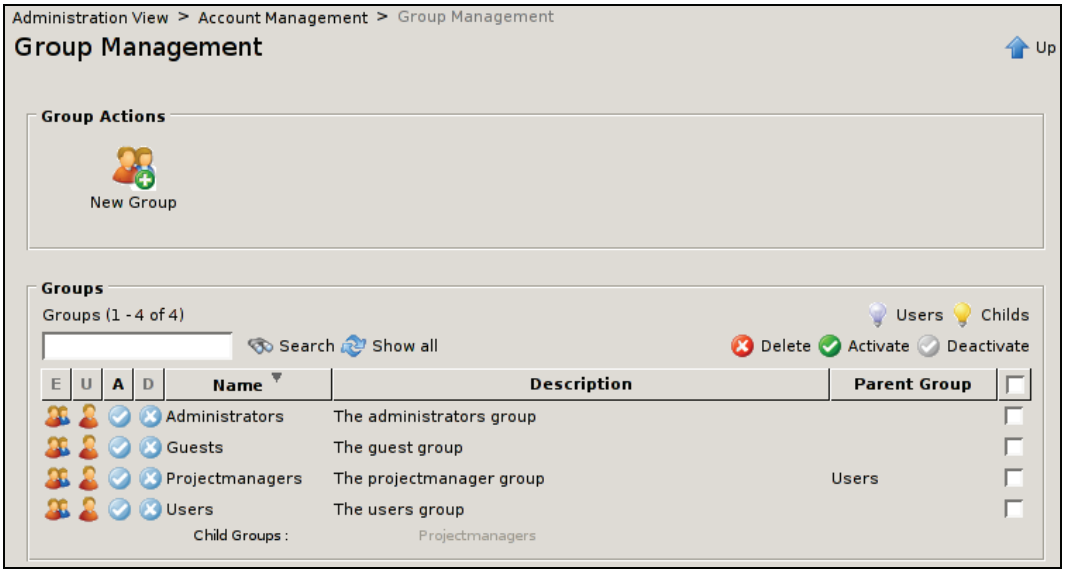
> "Activate" and "deactivate" are synonyms (respectively) for "enable" and "disable".

In the table of users, the beginning of each row has four buttons. The first is a single cartoon person. Clicking this button will open the Edit User screen for that user. The second, two cartoon people, brings up the Edit Groups of the User screen. The third button, a check mark in a circle, can be used to toggle the enabled/disabled status of the account. If the icon is blue, the account is enabled, and clicking the button will deactivate that account. Conversely, if the icon is gray, then the account is disabled, and clicking the button will enable the account. Finally, the fourth button (a white X in a blue circle) is for deleting the account. Clicking on a name in the Login column will bring up the User Overview screen.

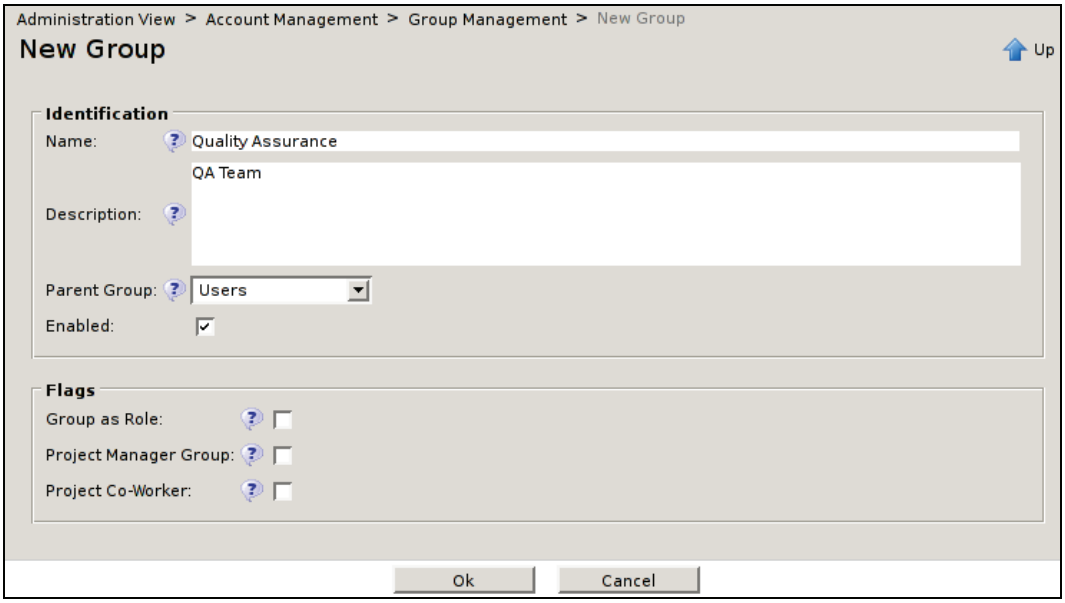
At this point, we have looked at the user tools and seen what OpenCms groups can do. Now, we will switch focus from user management to group management. Use the breadcrumb navigation at the top of the screen to return to Account Management.

Group Management

The second button in the Account Tools section of the Account Management screen is Group Management. Clicking this button loads the Group Management screen, which has two tables, Group Action and Groups.



Clicking on the New Group button in the Group Actions section will load the screen for creating a new group.



As you can see from the image above, creating a group is simpler than creating a new user. The Name text field should be given a human-readable name for the group (spaces and special characters are fine). The Description text box, which is optional (though it is not marked as such), should contain a helpful summary of what the group is for. Since this information is displayed in a number of tables, it is best to keep it brief.

Earlier, I noted that groups can be organized in a hierarchy. The Parent Group drop-down list allows you to make this new group a child of an existing group. In the example above, the Quality Assurance group is a child of the Users group so that all members of the Quality Assurance group will automatically be in the Users group (and be granted the privileges of members of the Users group).

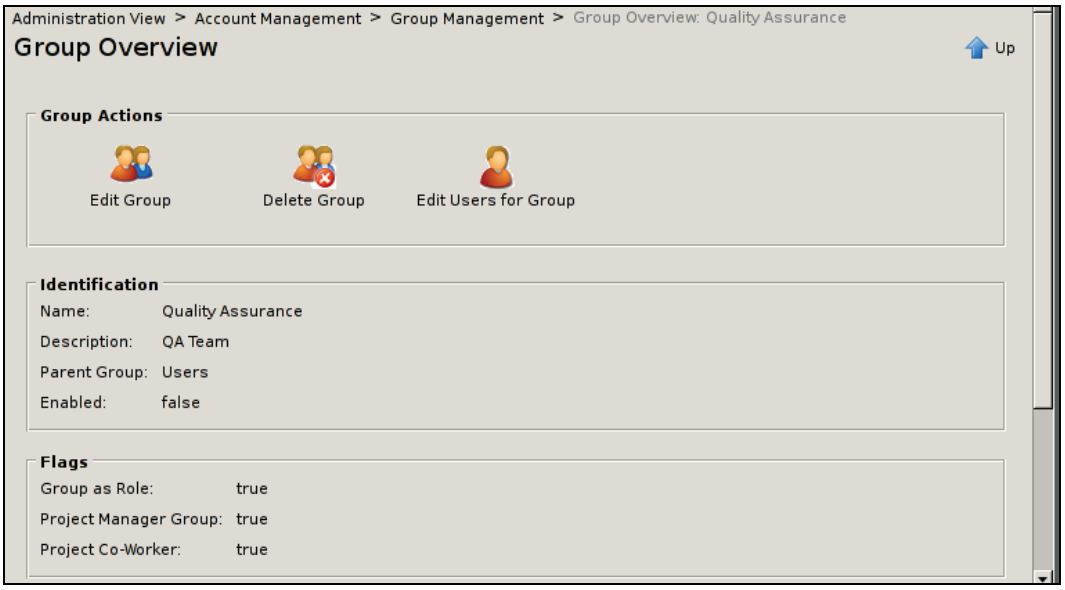
The Enabled checkbox determines whether the group will be immediately enabled. If it is left unchecked, the group will be disabled. If a group is disabled, users can still be added to and removed from the group, but group members may not be able to access content that is restricted to members of the group.

In the Flags section, there are three checkboxes, all of which are unchecked by default. The three boxes are as follows:

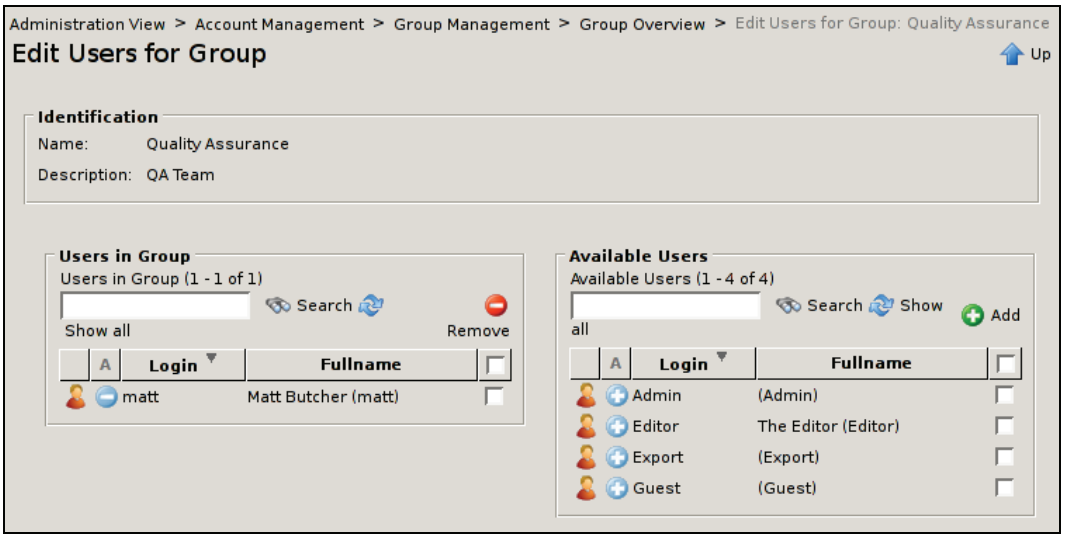
1. Group as Role: If this is checked, project management tasks can be assigned to the group as a whole. (We will discuss this in more detail when we look at the workflow view in Chapter 5.) This is useful when the group as a whole conducts certain tasks.

2. Project Manager Group: If this is checked, all members of the group will have the permissions necessary to manage projects, including publishing and assigning tasks (in the workflow view).
3. Project Co-Worker: This checkbox is also related to the Workflow view. If it is checked, a member of the group will be able to see tasks assigned to other members of the group.

Clicking the Ok button will add the new group and take you to the Group Overview screen. This screen is similar to the User Overview screen and displays detailed information about the group.

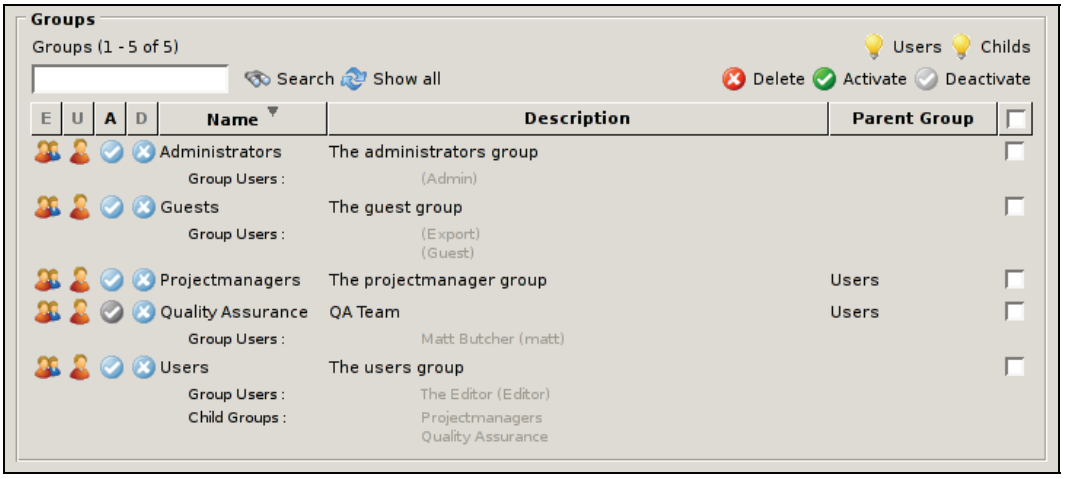


In addition to the information displayed, there are three buttons in the Group Actions section at the top of the screen, Edit Group, Delete Group, and Edit Users for Group. The Edit Group button will load a screen similar to the New Group screen discussed above. You can use this screen to modify group information. The Delete Group button will immediately (without prompting) delete the group from the database. Finally, the Edit Users for Group button will load a screen for managing the members of the group.



This screen is similar to the Edit Groups of the User screen. The top section, Identification, has basic information about the group. The Available Users section on the right displays the list of users that can be added to the group, and the Users in Group section displays the current members of the group. The buttons on this screen work in the same way as those on the Edit Groups of the User screen.

Clicking on Group Management in the breadcrumbs will reload the main screen for managing groups. At the bottom of this screen is a section with information about all the current groups.



In the upper-right corner, there are two buttons with light bulb icons. These buttons turn on additional information about the items in the table. (In the image above, both buttons have been pressed and the additional information is displayed). The first, Users, displays a list of members for each group in the table. The second, Childs, displays a list of the child groups.

Beneath these two buttons, there are three bulk-update buttons, Delete, Activate, and Deactivate. These buttons will operate on all the groups whose checkbox (in the far right column) is checked. Delete will remove all the selected groups from the database. Activate will change the status of all the selected groups to enabled, and Deactivate will disable all the selected groups.

At the beginning of each row are four icons. Clicking the first (two people) will load the Edit Group screen. Clicking the second (one person) will load the Edit Users for Group screen. The third button (a check mark on a circle) toggles the enabled/disabled status for the group. If the icon is blue, the group is enabled, and clicking on it will disable the group. If the icon is gray, then the group is disabled, and clicking on it will enable the group. Clicking the fourth button, a white X on a blue circle, will delete the group.

Finally, if you click the name of the group (in the Name column), the Group Overview screen will be loaded.

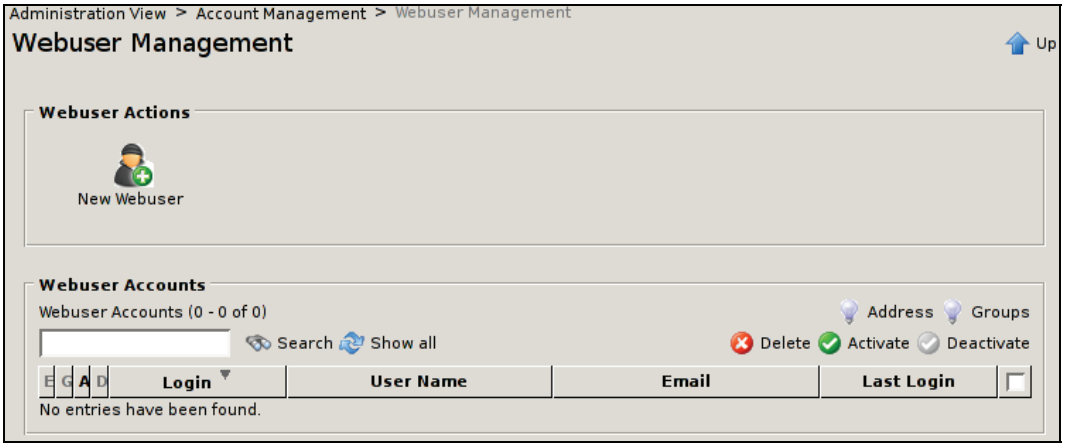
At this point, we have finished looking at the Group Management tools. We will now turn to the final section of the Account Management screen, Webuser Management.

Webuser Management

Earlier in this chapter, we saw how webusers differ from users. The main difference is that webusers cannot access the OpenCms Workplace, though they still have accounts in OpenCms.

It may be desirable to set up a membership-like feature for the published website, which will enable users to visit certain published (but restricted) resources but not to log in to the Workplace. This is the idea behind webusers. However, simply adding some webusers will not automatically provide a complete system for handling a membership-based site. You will have to do some more work to configure OpenCms.

Clicking the last button in the Account Tools section of the Account Management screen will load the Webuser Management screen. Clicking this button will load the main screen for administering webuser accounts.



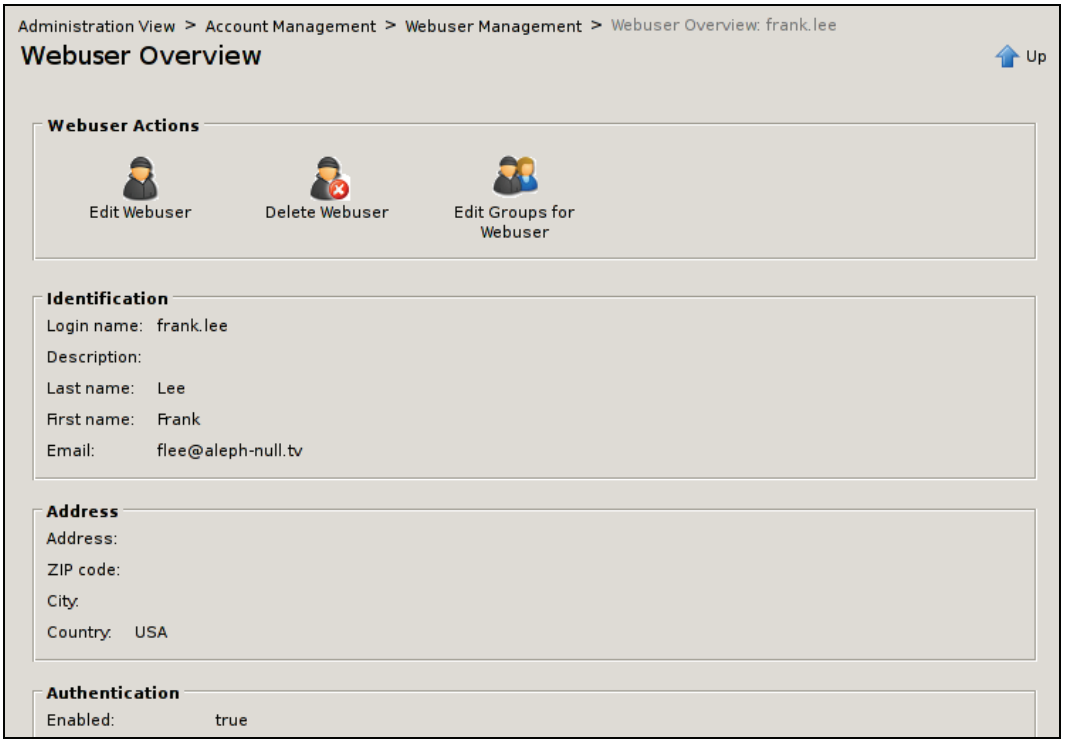
This screen is very similar to the User Management screen (though the cartoon webuser looks like a criminal). Clicking on the New Webuser button in the Webuser Actions section will load the form for creating a new webuser account.

There are four required fields and one optional field in the first box (Identification). These boxes are the same as those in the New User form. The Login name text field must contain no white space, and only some special characters are allowed.

None of the four fields in the Address section are required. In fact, before information can be added, you must click the green plus sign to add the field. In the screenshot opposite you can see that I added the Country field.

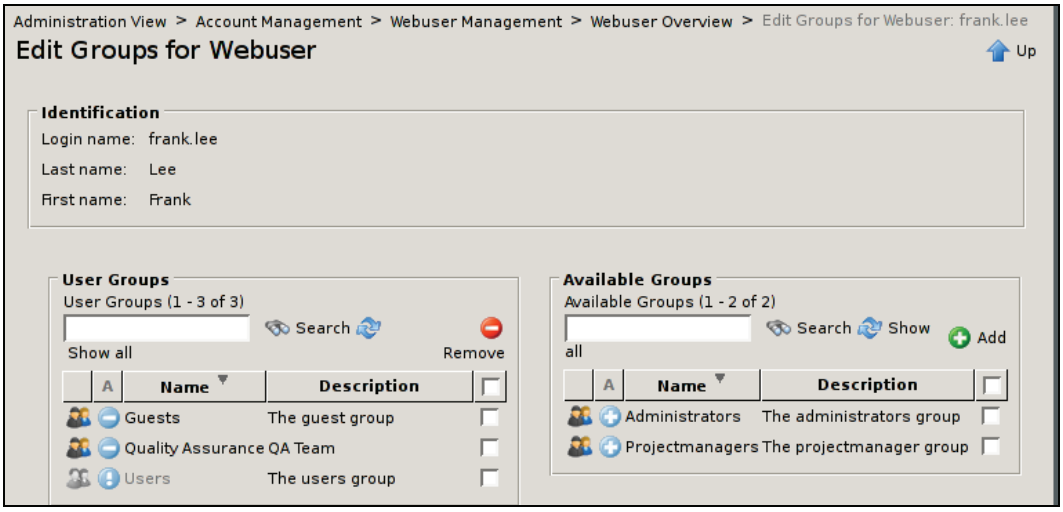
In the last section, Authentication, there is a checkbox and two text fields (Password and Password Confirmation) for the password. If the Enabled checkbox is checked and a password is set, then the webuser will be able to log in (assuming there is a location on the published site for webusers to log in to). If the checkbox is not checked or a password is not set, then the user will receive a message saying that the account is disabled.

Clicking Ok will check the entered data, save it in the webuser database, and load the Webuser Overview screen.



At top of this screen is the Webuser Actions section, which has three buttons: Edit Webuser, Delete Webuser, and Edit Groups for Webuser. Clicking the first button loads the Edit Webuser form, which is similar to the New Webuser form (though the Login name field cannot be changed). If the second button, Delete Webuser, is clicked, the webuser will be removed from the database of webusers. Finally, clicking the Edit Groups for Webuser button loads a screen for adding the webuser to existing groups.

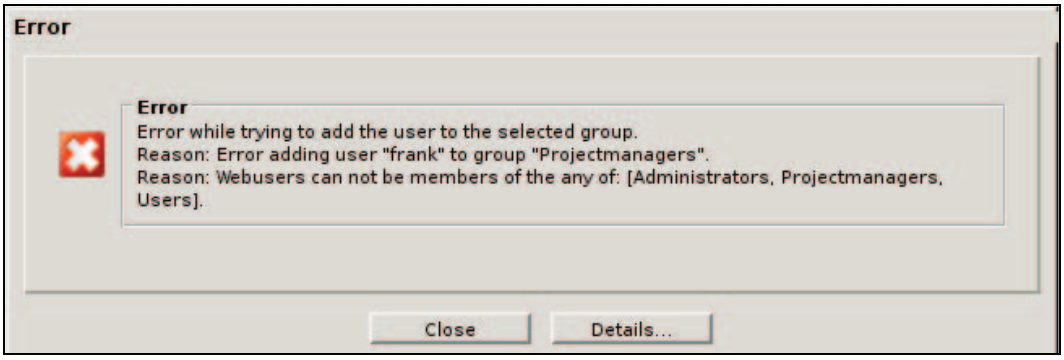
The Edit Groups for Webuser screen works in much the same way as the Edit Groups of the User screen.



At the top of the screen is some basic information about the webuser. On the lower right is the Available Groups table, which lists all the groups. Webusers cannot be added to some of the groups in this list. On the lower left, the User Groups section shows which groups the webuser belongs to. The buttons in both of these tables work in the same way as those in the Edit Groups of the User screen.

All this looks familiar. However, there are some crucial differences. First, all webusers are automatically members of the Guest group. (Regular users are not automatically members of any group.)

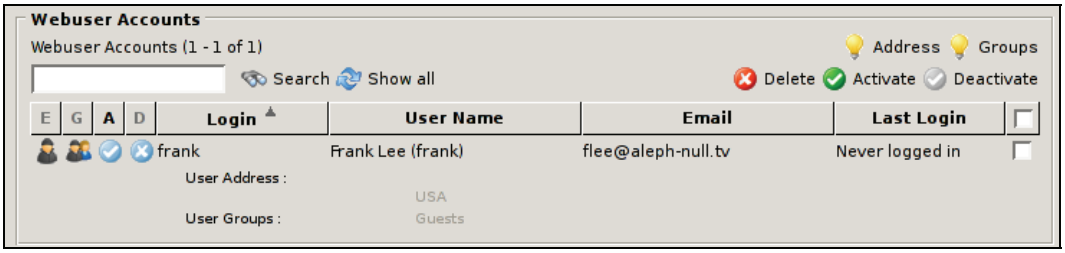
We know already that webusers are not allowed to access the Workplace. Further, they cannot be direct members of the other three default groups, Administrators, Projectmanagers, and Users. Trying to add a webuser to any of these groups will result in an error message that looks like this:



Webusers cannot be directly added to these groups, but they can become indirect members of these groups if they are assigned to a child group of Users, Projectmanagers, or Administrators. (You can see an example of this in the screenshot above of the Edit Groups for Webuser screen.) However, even if a webuser becomes an indirect member of one of these groups, the webuser will not be allowed to log in to the Workplace.

Although there are no known security problems resulting from making a webuser a member of a privileged group, adding a webuser to a privileged group is not a good idea. Modules may make assumptions about users in such groups, and the webuser may inadvertently be granted access to sensitive information. The best policy is to only put webusers in groups that either have no parent or have the Guest group as a parent.

To take one last look at the webuser tools, click on the Webuser Management breadcrumb. At the bottom of the Webuser Management screen, there is the now familiar table of accounts—in this case, it is the Webuser Accounts section.



This section works in the same way as the User Accounts section of the User Management screen. In the upper-right corner, there are two light bulb icons labeled Address and Group. When they are turned on (as they are in the above screenshot), additional information about each webuser is displayed. Beside the text User Address, you can see any optional address information that was specified in the New Webuser or Edit Webuser forms, and beside the text User Groups, you can see a list of groups of which the webuser is a member.

Beneath these buttons are the now-familiar mass-update icons for deleting, activating, and deactivating users *en masse*.

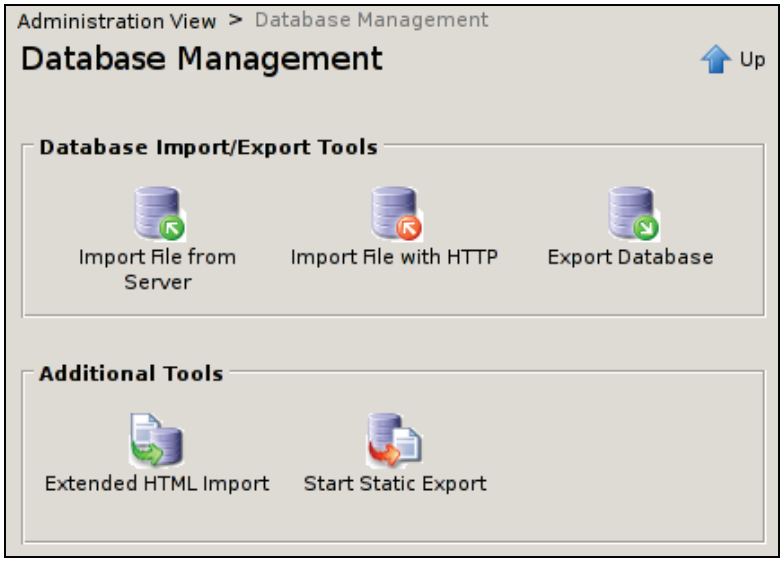
Each row in the table begins with four buttons. Clicking the first opens the Edit Webuser screen, clicking the second opens the Edit Groups for Webuser screen, clicking the third toggles the enabled/disabled state of the user, and clicking the fourth deletes the user.

Database Management

The third item in the administration view is the Database Management screen. OpenCms uses a relational database (in our case, an MySQL database) to store the contents of the Virtual File System (VFS), as well as ancillary data such as user accounts, activity logs, and content history.

Although the VFS is contained entirely within the database, the user sees it as a file system organized no differently than the normal operating-system file hierarchy (as we saw in Chapter 3). Likewise, OpenCms abstracts away from the details of database interaction, so you can perform most tasks without giving a thought to the underlying database operations.

However, as an OpenCms administrator, you may need to perform certain maintenance tasks that deal more directly with the underlying database. The Database Management screen in the administration view contains tools for this purpose.



The Database Management screen has two sections. The Database Import/Export Tools section is used to access tools that move data into and out of the database. These tools come in handy for backing up OpenCms, restoring an older database and even loading content from other OpenCms servers. In OpenCms 6.0, this section has three buttons, Import File from Server, Import File with HTTP, and Export Database.

The second section is called Additional Tools. This section is used to access tools that also operate on the contents of the database but perform more specific tasks. Rather than just moving data to and from the database, these tools conduct additional processing such as

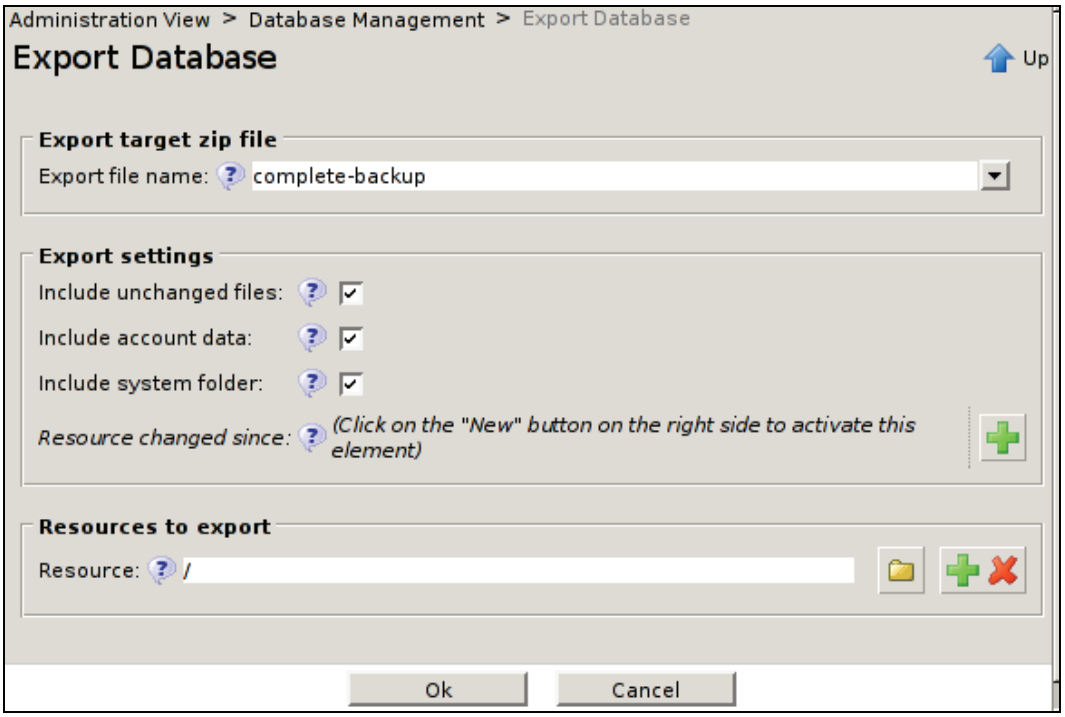
cleaning HTML tags or running Java or JSP code. In this section, there are two buttons, Extended HTML Import and Start Static Export.

Exporting from the Database

First, we will examine the Export Database screen, as we will need to use it before we can do much with the other two tools.

The Export Database screen allows you to selectively export resources from the database. You can export all the VFS content (as well as user and group information), or you can export just a selected portion of the VFS, which is then dumped into a zip-compressed file that is stored in the server's file system.

You can make a complete backup of the entire contents of the VFS, including all of the information in the system (root) folder. The screenshot below shows such an export:



The first text field, Export file name, is for the name for the ZIP file containing the export. The `.zip` extension is automatically appended—for example, I have entered complete-backup, so my information will be stored in a file called `complete-backup.zip`.

The contents of an export are determined in part by the *project* from which the export is run. If the current project is the Offline project, then the export process will look at all the files in that project. If the current project is Online, then only the published files in the Online project will be stored in the exported file. Custom projects (such as the Playground project that we created in Chapter 3) are considered sub-projects of Offline, so exports run from these projects should be the same as exports run from the Offline project.

In the Export settings section, there are three checkboxes and one optional field. The first checkbox is Include unchanged files. If this box is unchecked, then only files that are marked as changed (added or modified) will be exported, but if the box is checked (the default), then files will be exported regardless of whether they are marked as changed.

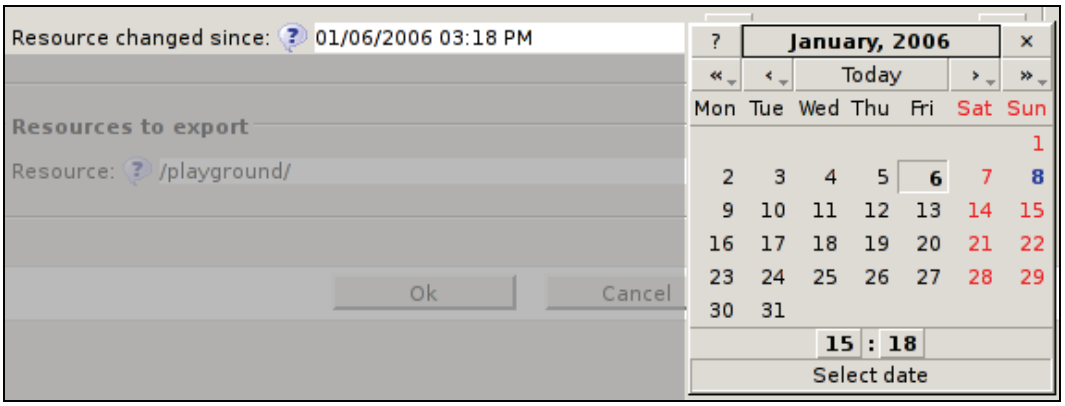
> Information about folders is always exported, regardless of their publishing state. If a project containing only one changed file is exported, information about all of the folders (even those that contain no changed data) is exported.

When the second checkbox, Include account data, is checked, user and group information is exported as well, but webusers are not exported. If you have a large number of webusers, you should not use the Export Database screen to protect their account data.

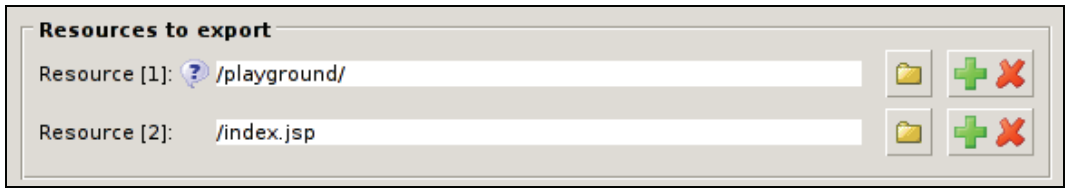
The third checkbox, Include system folder, determines what parts of the VFS will be exported. As you may recall from Chapter 3, content is stored in site folders located in the `/sites` folder. By default, there is only one site, located (aptly) in the folder `/sites/default`. If the Include system folder checkbox is unchecked, then only files in the current site will be exported. However, OpenCms stores valuable data (including OpenCms modules and the Workplace files) in different parts of the VFS. For example, modules are stored in `/system/modules`, and the Workplace files are stored in `/system/workplace`. To export the complete VFS, you must check the Include system folder checkbox and switch sites from `/sites/default` to `/`.

> Checking the Include system folder checkbox is not enough to include the system folder, `/`, in an export. You must also set the Site drop-down list (in the toolbar at the top of the Workplace) to `/` and enter the appropriate path (usually just `/`) in the Resource field of the Export Database screen.

The last item in the Export settings section is the optional Resource changed since field. Clicking the green plus sign will add a text input field with a calendar icon, which allows you to specify a particular date. Only files modified after that date will be exported. You can either type in the desired date and time information (for example 01/06/2006 03:18 PM), or you can click on the calendar icon, and select the desired date and time from the pop-up calendar widget.



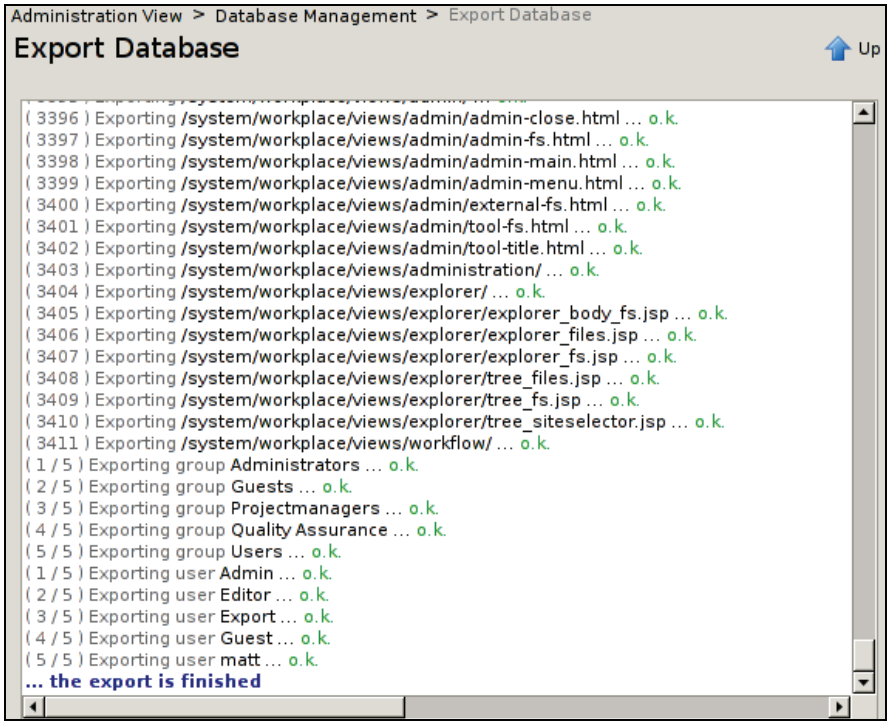
Beneath the Export settings section is the Resources to export section. By default, this has one field, the Resource text field. However, you can add more text entry fields by clicking on the green plus sign to the right this field. For example, below is an image with two resource fields. (The Resource field is relabeled Resource [1], and the new field is labeled Resource [2].)



These fields are used to specify which files and folders to export. You can either type in the path, or you can click on the file folder icon to the right of the text input and navigate to the desired file or folder.

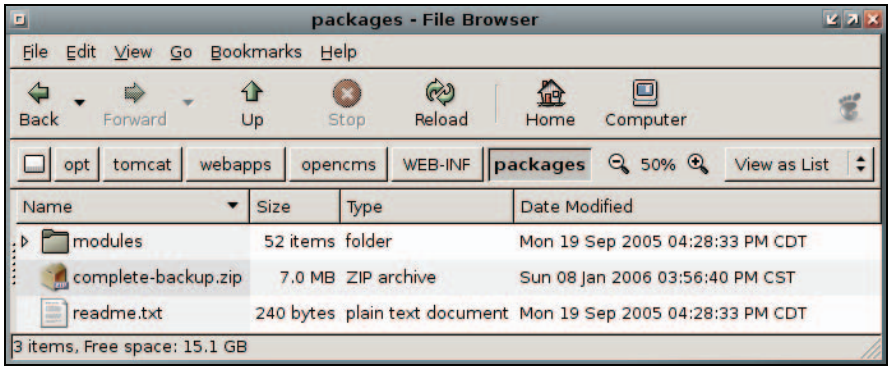
By using various combinations of the Export settings screen's fields, you can be highly selective as to what data are exported. Once the desired configuration is set, clicking the Ok button will begin the export process.

An export of the complete VFS can take a long time, as even a default installation of OpenCms has many megabytes of data stored in the VFS—especially in the `/system` folder.



A complete backup of my relatively empty OpenCms instance (with the Include system folder box checked) still exported over 3,400 files, and the resulting ZIP file was over seven megabytes in size.

Once the export has finished, you can click the Ok button at the bottom of the screen to return to the Database Management screen. But where is the exported file? Exported files are stored in the server's file system in one of the directories within the OpenCms web application. In our case, export files are written to the folder `$CATALINA_HOME/webapps/opencms/WEB-INF/packages`.



If you take a look inside the file, you may be surprised at what you find. At the root of the export folder, there is a file called `manifest.xml`, which contains information about everything (files, folders, users, and all) exported from the database. This file is generated automatically during the export process.

Many of the files appear very differently in the export than in the Workplace. For example, all the HTML files are actually XML files containing not only the HTML generated by the WYSIWYG editor but also additional information about the hyperlinks and images used in the document.

Why do the files appear this way? Because they are exact copies of what is in the database. When you view these files in the Workplace, or when OpenCms displays these files for the browser, the server has already manipulated these files, formatting them for presentation. But when OpenCms exports files, it exports them in their raw format. This is done so that these files can be re-imported into OpenCms without losing any important information.

> The Export Database screen is used to export data so that it can be re-used by OpenCms. If you want to generate processed (ready to view) content, you should use OpenCms's static export tool, which is covered below.

So far, we have looked at the process of exporting the contents of the VFS from the database to a set of files compressed into a ZIP file stored on the file system. In the example above, I showed how to back up the entire VFS. Of course, the export tool is useful for more than just backing up files. It can also be used to move content from one OpenCms server to another.

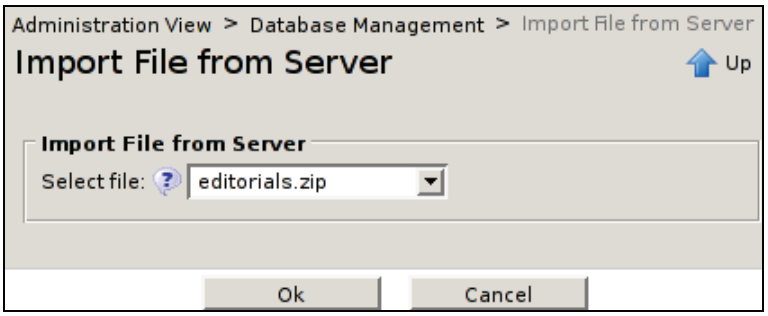
For example, we could create a folder full of content on one OpenCms server and then export just that folder. If we have a folder, `/site/default/editorials`, on one OpenCms server and want to move it to another OpenCms server, we should start by creating a new export containing only the contents of `/site/default/editorials`. Now we have two different ways of loading the content into the desired OpenCms. We can either do a local import or import the file over the network. Both of these tasks can be accomplished from the Database Management screen.

Importing a File from the Server

The first button in the Database Import/Export Tools section is called Import File from Server. The Import File from Server screen is used to look for ZIP files in the export folder `$CATALINA_HOME/webappas/opencms/WEB-INF/packages`. So the first step when importing a file is to place the desired file in that folder.

> The Import File from Server screen is used to import from the same folder that the Export Database screen is used to export to. This is very convenient when it comes to restoring OpenCms from a backup.

The Import File from Server screen has drop-down list with the names of all the ZIP files in the export folder:



Simply select the desired export file and click Ok. OpenCms will then import the files. These files will appear in the VFS organized in the same way as they were on the server from which they were exported.

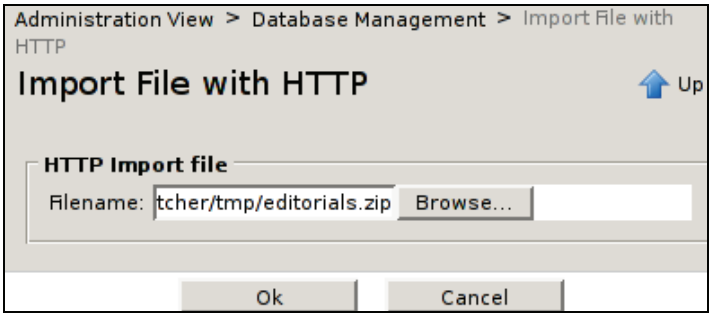
The imported version is considered definitive. Thus, if an imported file has the same name as a file already in the VFS, the imported file will always replace the other file. If an import file is identical to a file already in the VFS, the file will be marked as modified, so importing a file may result in a number of unchanged files being marked as modified.

Once the files have been imported, they work just like all other OpenCms files.

While the Import File from Server screen is indispensable for recovering from failures, it is not always convenient for uploading content from another server. The file must be moved from the remote server to the correct location on the desired server before it can be imported. Sometimes it is more convenient to import content from your local workstation to the server over the network. This can be accomplished with the Import File with HTTP screen.

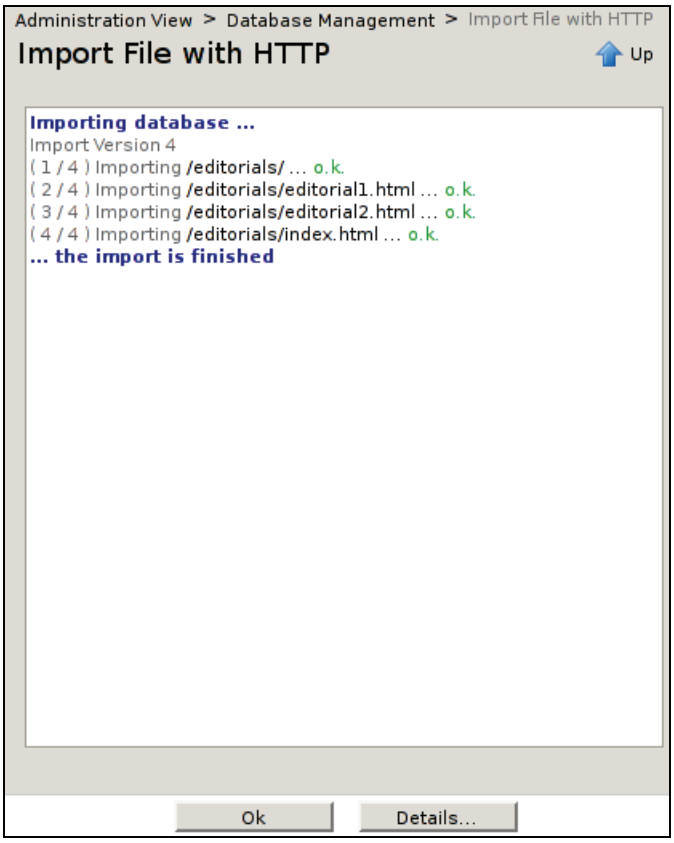
Importing a File with HTTP

The Import File with HTTP screen can also be accessed from the Database Management screen. This screen is used to load a file from the local file system (the file system of the machine on which the web browser is running). It uses HTTP (Hypertext Transfer Protocol—the protocol for requesting and retrieving web pages) to send the file from a local machine to the server. The Import File with HTTP screen looks like this:



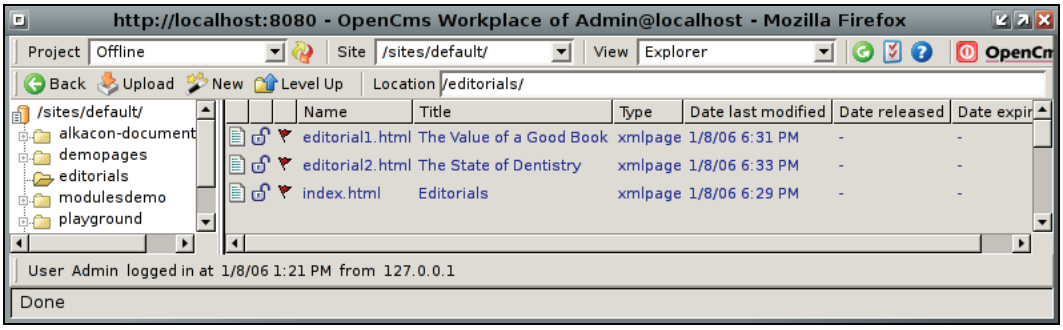
You can either type in the path to the desired file in the Filename text field or click the Browse... button to navigate to the file in your local file system. Once you have found the file, click Ok to upload it to the OpenCms server.

OpenCms will immediately import the file, showing you the status of the import as it goes.



Once the import is complete, the Ok button at the bottom of this screen will become active. Clicking it will return you to the Database Management screen.

Switching to the explorer view, we can see the imported files:



Since these files are all new, they are marked as new and unpublished (blue text, red flag).

We have seen how to export files from the VFS and how to import them using tools accessed from the Database Import/Export Tools section of the Database Management screen. There are two other tools that can be accessed from the Additional Tools section of the same screen, the Extended HTML Import screen and the Start Static Export screen.

Extended HTML Imports

In my experience, one of the biggest headaches in migrating a website to a content management system is getting all of the old information loaded into the new server. OpenCms provides a tool to ease the pain.

The Extended HTML Import screen can be accessed from the Database Management screen in the administration view. It can be used to read through a collection of HTML files stored in the local file system, parse the files and then import the content (images and all) into OpenCms.

The import tool is designed to handle a moderately sized website, but in this example, we will be importing a very small site, which has two web pages, an image and a downloadable file. The site, stored on the server's file system, looks like this:

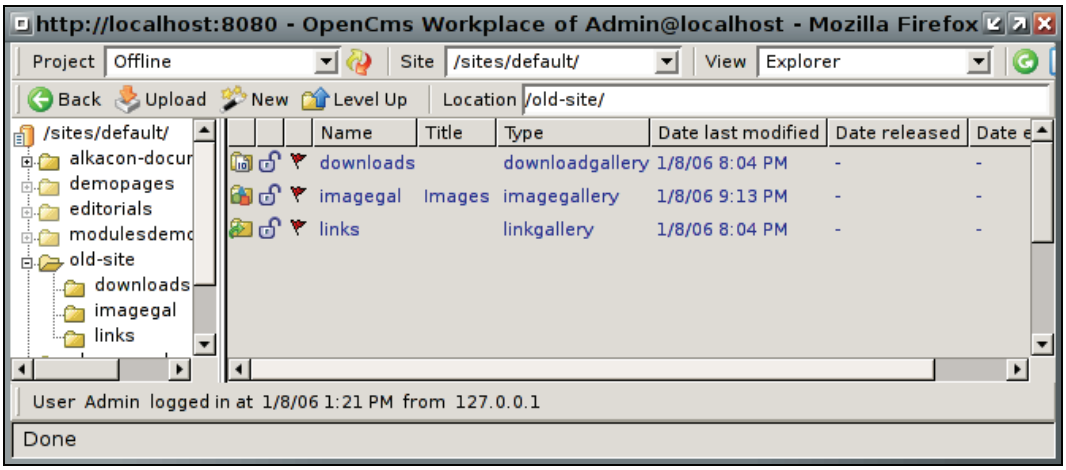
```
mbutcher@locahost:~$ ls -R old-site/
old-site/:
images/  index.html  penguin.html  penguin.zip

old-site/images:
penguin.jpg
mbutcher@localhost:~$
```

I want to import all of this into OpenCms so that I can manage it from the Workplace.

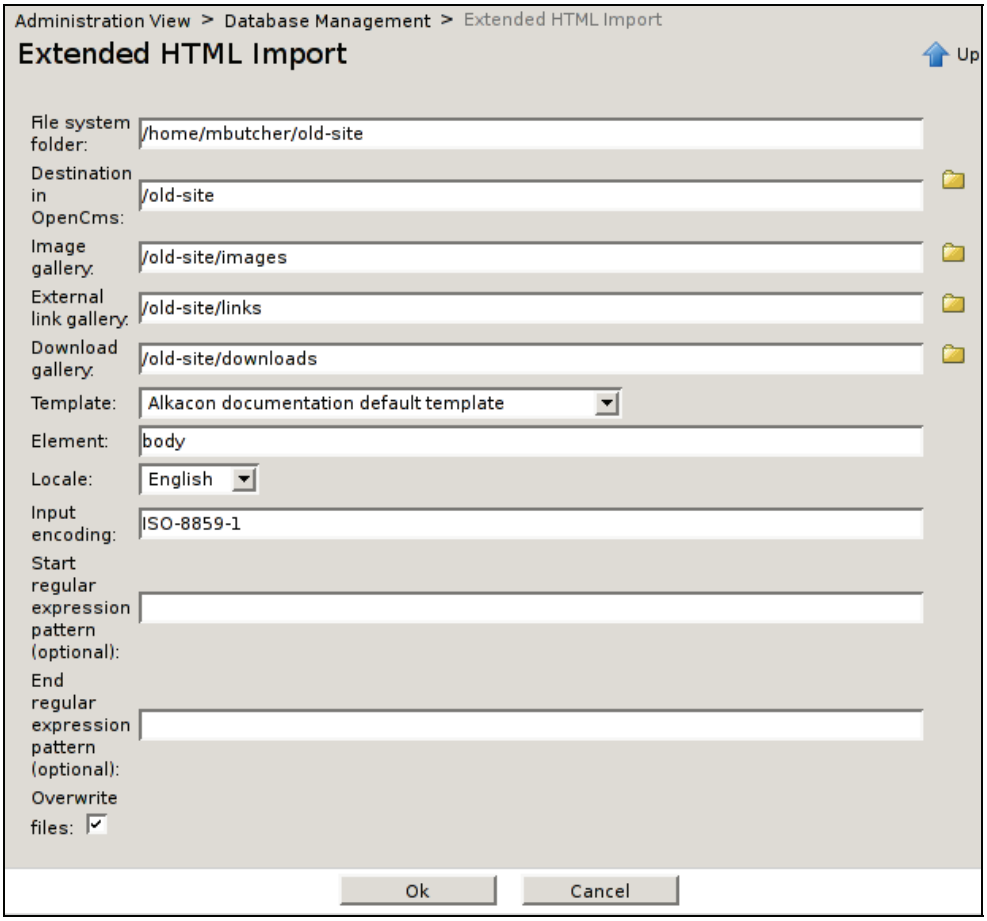
The first thing to do is set up the target folders in OpenCms. I have created a new folder (named unimaginatively `old-site`) in the explorer view. The old site has an image file, a downloadable file and a link to an external website (in `index.html`), so OpenCms will need galleries for each of these content types. For that reason, I have created an image gallery (`old-site/imagegal`), a download gallery (`old-site/downloads`) and an external link gallery (`old-site/links`), all inside of the `old-site` directory.

> In Chapter 3 we covered creating folders and galleries.

The `old-site` folder looks like this:



Now that the VFS is set up to handle the import, we can go back to the Database Management screen in the administration view. Clicking on the Extended HTML Import button loads the import tool:



Type in the complete path to the site that will be imported in the first field on the form, File system folder. The files must exist in the file system of the server running OpenCms. Unfortunately, there is no button for browsing the file system. You will need to type in the exact location (i.e. the absolute path, for example `C:\folder\` in Windows or `/var/www` on UNIX) to the main folder that contains the website.

The second field, Destination in OpenCms, refers to the VFS. When the files are imported, they will be placed in this folder. You can type the path in by hand, but this field (like the next three fields) has a folder icon which, when clicked, brings up a file browser.

The Image gallery, External link gallery, and Download gallery fields should all point to galleries in the VFS. While importing, OpenCms will store images, links, and downloads in these galleries, instead of putting them with the HTML files in the folder specified in the Destination in OpenCms field.

When OpenCms imports HTML files, it will attempt to retrieve the content from the body of the HTML file and save it as a Page type file. When displaying the file, OpenCms will apply a template. (See Chapter 3 for more on file types and templates.) The Template field on the Extended HTML Import form determines which template the imported pages will use.

As mentioned above, the import tool will use a template when rendering the document, and the Element field determines where in the template the content will go. Most templates define an area called body (the default) or content.

The Locale field determines the language for the imported HTML files. This language must already be one supported by OpenCms. You may need to download and install a particular language module to get the desired locale.

The Input encoding field should be set to the character set used by the imported documents. The default, ISO-8859-1 (a.k.a. Latin-1), is the most common character set for English-language documents.

The next two fields, Start regular expression pattern and End regular expression pattern (both optional) provide an advanced method of indicating where content begins and ends within an HTML document. By default, OpenCms will import what it finds in the body element in the HTML file. If the web pages have a complex layout, simply importing everything in the body element may create a significant editing task. To avoid importing unnecessary formatting, you can specify a regular expression (a short piece of code that searches for patterns in a document) to tell OpenCms what to consider as the beginning and the end of the content section.

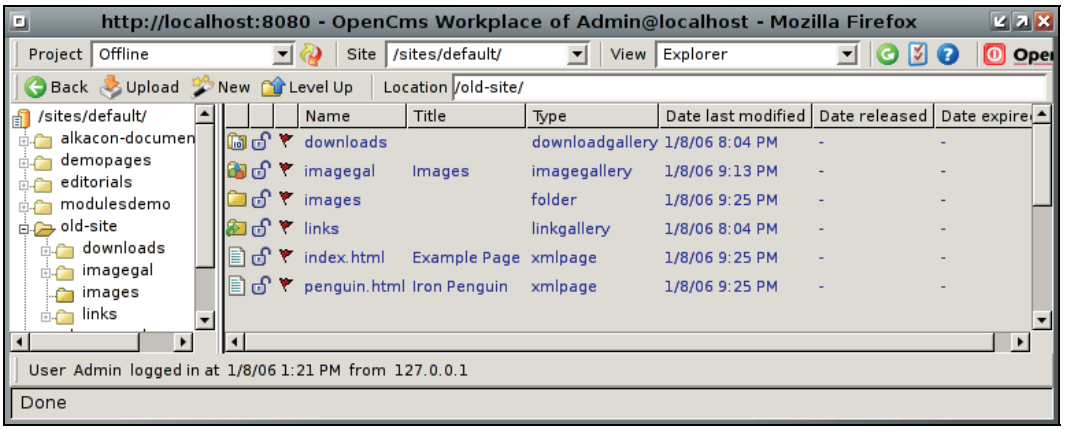
> OpenCms uses the **Jakarta ORO** package to handle regular expressions. This library supports the well-documented Perl5 style of regular expressions. The Jakarta ORO package is an Open Source project maintained on the Apache Jakarta site: `http://jakarta.apache.org/oro/index.html`.

The regular expression tools can be very useful when importing, but for our simple site, they are not necessary.

The last field on the form is a checkbox called Overwrite files. When this box is checked, OpenCms will privilege the imported copy over files already in the VFS. If an imported file has the same name as another file in the destination folder, the imported file will overwrite the existing file. Unchecking this box will prevent imported files from overwriting existing files.

Once the form is completed, you can click the Ok button to begin importing. OpenCms will show you progress information as it imports. Once the import is complete, you can click the Ok button again to return to the Database Management screen.

Switching to the explorer view, we can see the newly imported files in the `/old-site` directory.



Note that the `images` folder was imported, though the only file that was contained in that directory, the `penguin.jpg` image, is now in the image gallery.

The imported files are now pieces of OpenCms content and can be treated just like any other OpenCms file.

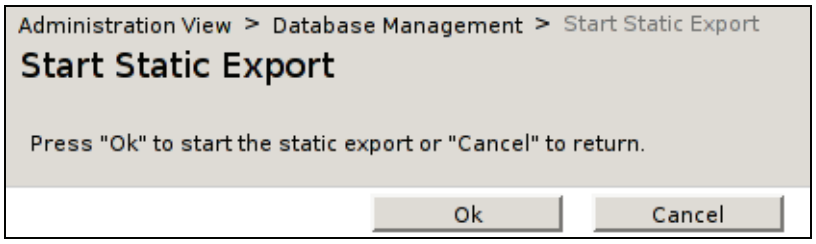
Static Exports

The last item on the Database Management screen is the Start Static Export button. Like the Database Export screen, the Start Static Export screen is used to export the contents of the VFS site to the server's file system. But it does so in a very different way and for a very different purpose—in a static export, files in the VFS are prepared for delivery to a client.

Some resources, such as images and download files, are not processed by OpenCms at all, so it is quicker for OpenCms to export these files to the file system and serve them from there. This is the simplest form of static export, and this type of export is done by default.

In a more complex configuration, the entire site can be exported. Pages are rendered in their templates, code is executed, and the results are formatted for delivery. But instead of delivering these processed files to the web browser (as would be the case in a normal request), OpenCms writes the files to the server's file system. The result is a completely static representation of the site.

The Start Static Export screen has just two items, the Ok button and the Cancel button:



As soon as you click the Ok button, the export will begin. OpenCms scans all the files in the site, determines which files can be exported (according to the current OpenCms configuration) and exports them.

An export can take several minutes or more. You will be able to watch the progress. Once the process is done, the Ok button at the bottom of the screen will become active. Clicking it will take you back to the Database Management screen.

All the exported files are written to the static export directory in OpenCms, usually located in `$CATALINA_HOME/webapps/opencms/export/sites`. When requests come in for these files, OpenCms will serve the exported file rather than the file stored in the VFS. This can speed up the server by avoiding needless processing.

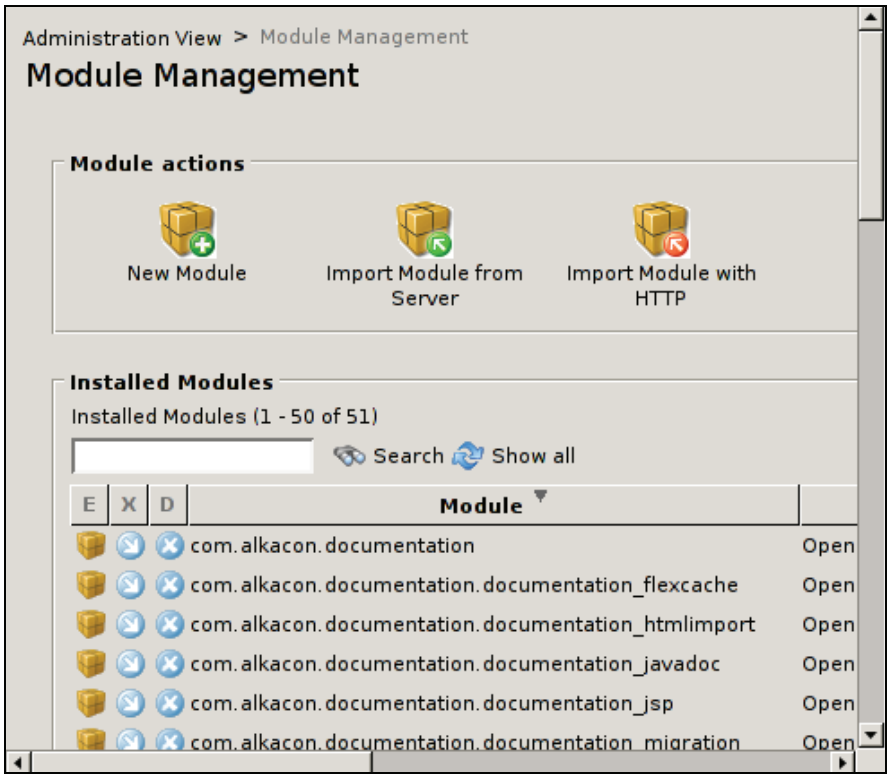
Modules

OpenCms is not a one-size-fits-all content management system. It can be tailored to fit the needs of an organization. In order to provide their users with the ability to fit OpenCms into their own environment, the OpenCms developers implemented a module architecture that can be used to extend OpenCms with new functionality and content. These modules are portable collections of code and content, including JAR files, Java classes, and JSP files.

When we installed OpenCms in Chapter 2, we installed a number of default modules, many of which contained only help information and documentation. But there are many more modules for OpenCms, some provided by Alkacon (the company that oversees OpenCms development) and some provided by third parties.

An OpenCms module consists of a well defined set of folders and files stored in a ZIP archive. A module may contain Java Archive (JAR) files, Java classes, JSP scripts, images, stylesheets, and any other type of content that OpenCms supports.

The Module Management screen in the administration view provides the primary interface for installing, modifying, and removing modules.



This screen can be used to upload and install new modules, as well as creating, editing, and exporting modules. In fact, you can even use it to create and distribute your own modules. At the top of the screen is the Module actions section, which contains three buttons. Beneath it, is the Installed Modules table, which should already contain a number of modules. Mine, which is only the default installation, has over fifty modules.

Here, we will look at module management, including finding new modules, adding and removing modules, and creating your own modules.

Obtaining Official OpenCms Modules

The OpenCms website (`http://www.opencms.org/`) has four different module archives, each providing a particular kind of module. These archives are:

1. **Documentation**: This archive contains supplemental documentation for OpenCms. Most of the documentation is packaged as OpenCms modules, though there are a few stand-alone files, such as a slide show introducing OpenCms. It is found at `http://www.opencms.org/opencms/en/download/documentation.html`.
2. **Localizations**: This archive contains modules that provide localizations for the Workplace. It is found at `http://www.opencms.org/opencms/en/download/localizations.html`.
3. **Modules**: This archive is for stable, production-quality modules. Usually, modules in this category are functional extensions to OpenCms. However, there are still some documentation modules from older versions of OpenCms in this archive. It is at `http://www.opencms.org/opencms/en/download/modules/index.html`.
4. **Module Sandbox**: This archive is similar to the last one, but these modules are considered less stable. Often times, they are under active development, so that new versions are released more often. While many of the modules in this archive are production-quality, you ought to test out a module before you install it on a production server. It is found at `http://www.opencms.org/opencms/en/download/sandbox.html`.

In addition to those found in the "official" module repositories, there are a number of modules available elsewhere. Often, information about new modules is posted to the OpenCms mailing list or to one of the OpenCms forums.

> The Synx OpenCms Forum is at: `http://opencms-forum.de/index.php`
> Information on the OpenCms mailing list can be found at:
> `http://www.opencms.org/opencms/en/development/mailinglist.html`.

In previous versions of OpenCms, the help modules and tutorials had to be installed individually after the system installation. With OpenCms 6 and later, that is no longer the case. These modules are now installed by default.

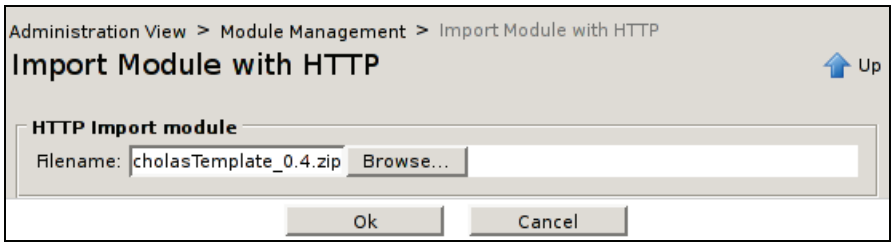
Below, we will look at the process of installing a module. As an example, I will install a very simple module from the OpenCms module sandbox. This module, the St. Nicholas Template module, just contains a set of templates. It is available in the Modules Sandbox archive on the OpenCms website.

> The St. Nicholas Template module is a simple module that contains one template. The author, Nico Michael, developed it for his own site but decided to share it as an example of a complete and functional template. This template is not as complex as the TemplateOne templates included with OpenCms, but it is a very useful example of how to use JSP tag libraries to create your own templates.

The file we will be working with is `com.st.nicholas.church.template.StNicholasTemplate_0.4.zip`, and I have downloaded it to my local workstation. I will upload it to the server using OpenCms's HTTP import. You can also install modules from the server's file system, but the modules must be located in `$CATALINA_HOME/webapps/opencms/WEB-INF/packages/modules`.

> Modules are named by strict convention. A module name should always begin with the full Java package name for the module (for example, `org.opencms.help`). The package name should be followed by an underscore and then a version number of the form `MAJOR.minor` (for example, `_1.1`). Finally, each module file must have the `.zip` extension.

Since the St. Nicholas Template module is stored on my workstation, we will use the Import Module with HTTP screen. You can load this screen by clicking on the Import Module with HTTP button in the Module actions section of the Module Management screen.

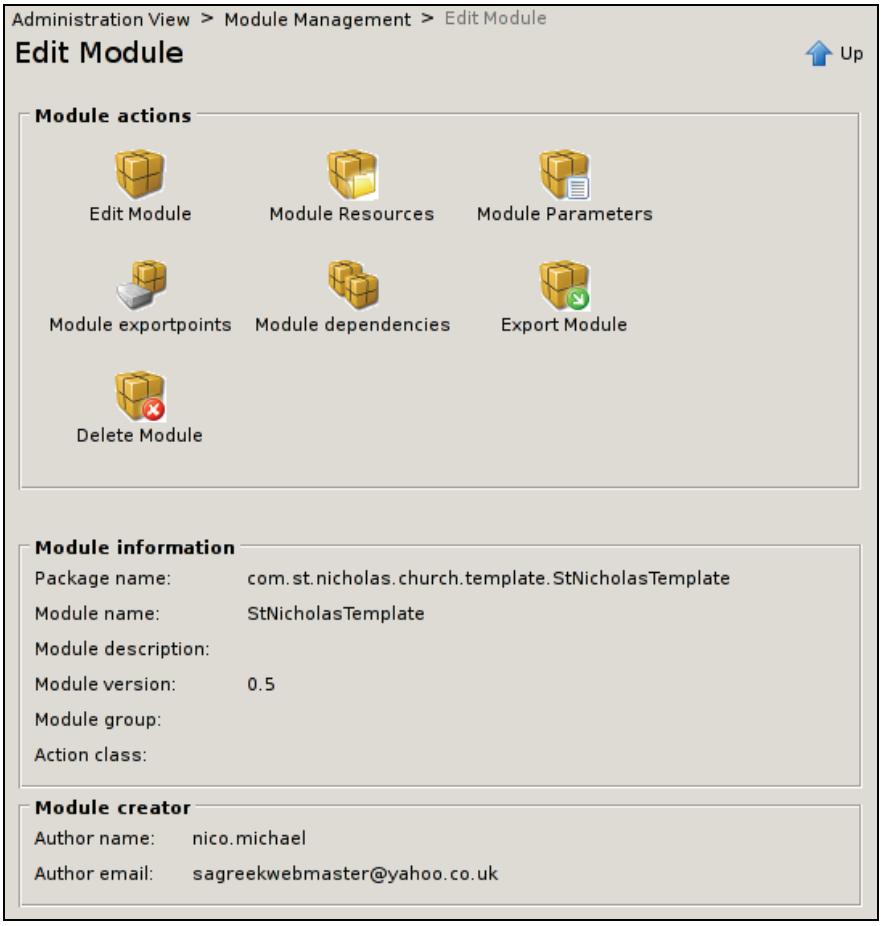


This screen has an upload field, Filename. Once we have selected the appropriate file, clicking Ok will begin the upload and installation process. As usual, progress will be displayed as the installation takes place. Once installation is complete, the Ok button at the bottom of the progress screen will be activated. Clicking it will take you back to the Module Management screen.

The new module is now listed (along with the others) in the Installed Modules table.

The First Edit Module Screen

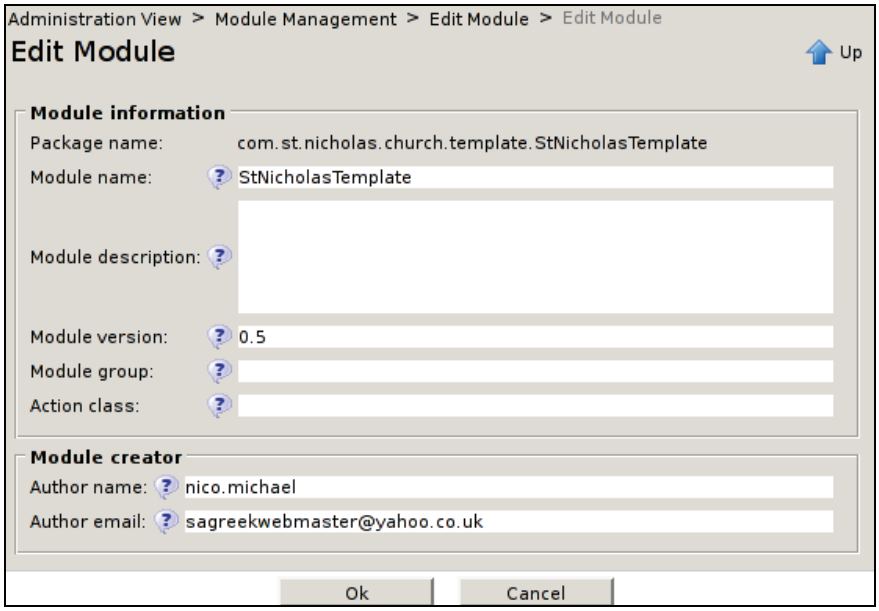
Clicking on the module's name in the Module column in the Installed Modules table on the Module Management screen will load a screen. The name of this screen is Edit Module, and it provides access to tools that manage the module.



At the top of this screen, there are seven buttons in the Module actions section. Beneath that, there is a Module information section, which contains various details about the module. At the bottom is a small section, Module creator, which provides information about who made the module and how they may be contacted.

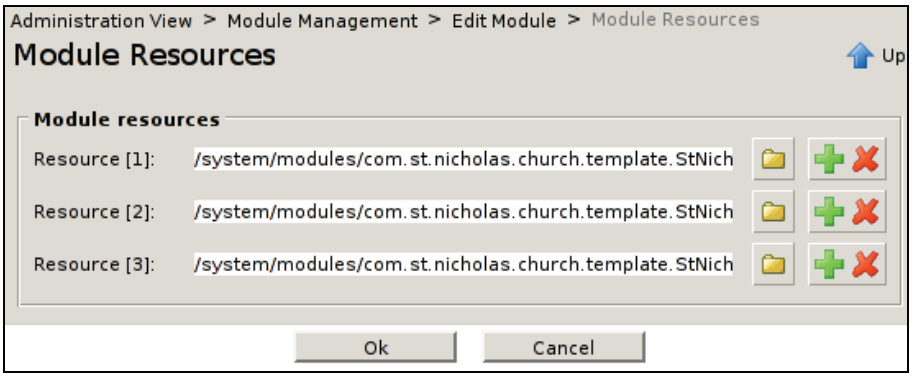
The Second Edit Module Screen

Clicking on the first button on this screen loads another screen, which is also called Edit Module. (The fact that there are two screens named Edit Module is a minor bug, which is likely to be fixed in future releases.) The second Edit Module screen is for editing the fields that show up in the Module information and Module creator sections of the first Edit Module screen.



The Module Resources Screen

Clicking the second button on the first Edit Module screen loads the Module Resources screen, which details where the module stores its files.



By default, every module stores information in its own module directory. This directory is located in the `/system/modules` folder (which is located in the site root, not in the `/sites/default` folder). This folder is named after the module's full package name (a Java convention for naming). For example, the St. Nicholas Template module's main module directory is `/system/modules/com.st.nicholas.church.template.StNicholasTemplate`.

But modules may also need to store information elsewhere in the VFS. The Module Resources screen shows the other folders the module uses to store information.

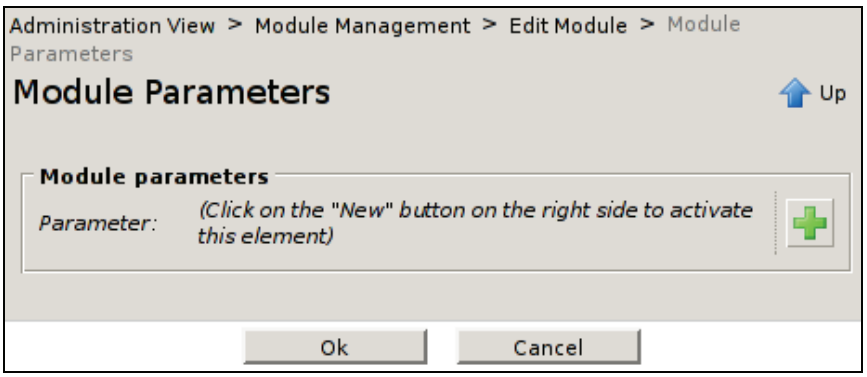
> When a module is exported, all of the files in every one of the Resources directories will be included in the module. Avoid listing a folder as a resource unless it is used exclusively for module content.

You can use the Module Resources screen to add more Resource folders as well.

All of the content that the module created is now in the VFS and is treated just like all other files. Module content can be locked, edit, published, etc.

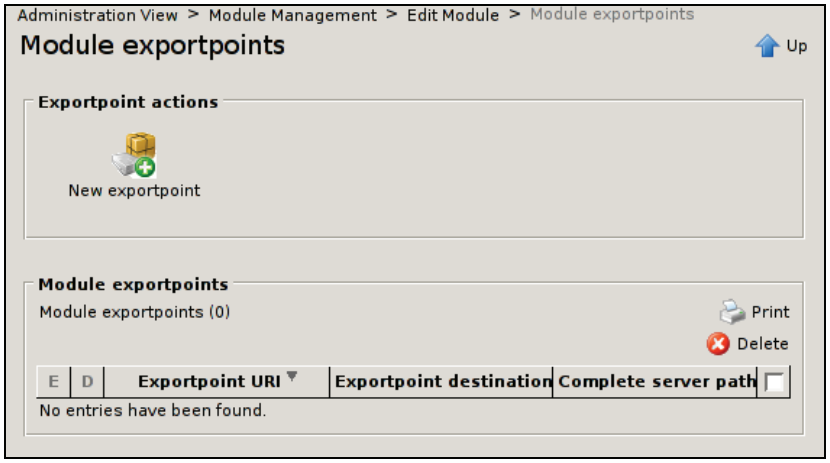
The Module Parameters Screen

Clicking the third button on the first Edit Module screen loads the Module Parameters screen. Like files, modules can have specialized parameters, in which additional information is stored. Sometimes, developers use these parameters to store configuration information, though in this case, no parameters have been specified for the St. Nicholas Template module.



The Module Exportpoints Screen

The Module exportpoints screen deals with static export information. Sometimes, some of the module data must be exported from the VFS to the real file system, and the Module exportpoints screen allows you to specify which content must be exported and where it should go.

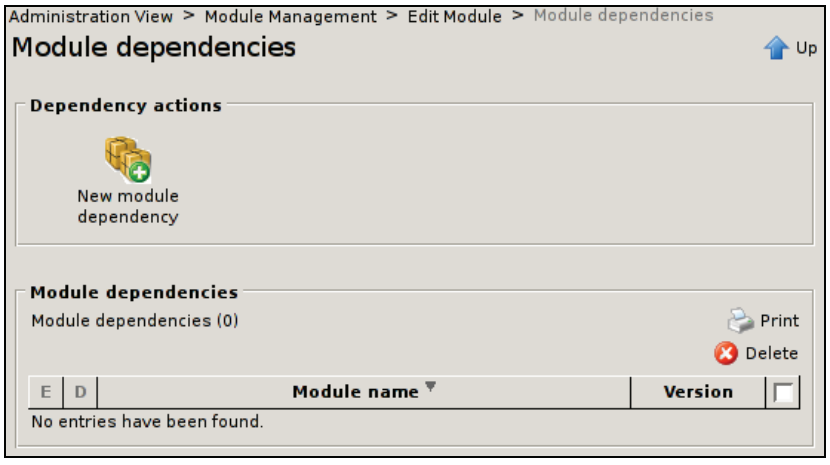


In some cases, you may need to configure custom modules to store information on the server's file system. You can use the New exportpoint screen to specify which location in the VFS should be exported and where it should reside on the server's file system. Any time that your module uses Java library files (JARS or Class files), you will need to make sure that an export point is defined for those files.

> Modules can only export content to places within the `$CATALINA_HOME/webapps/opencms` folder. Files cannot be written to other locations in the server's file system.

The Module Dependencies Screen

The Module dependencies screen provides information on which modules are required by the present module for it to function correctly.



During the course of module installation, OpenCms checks that all of the new module's dependencies are already installed. If they are not, OpenCms will not allow the new module to be installed.

The Module dependencies screen provides information on the module's dependencies. These dependencies are listed in the Module dependencies table. If you create a module that uses resources from another module, you have create a dependency. But OpenCms does not automatically detect this. You will need to use the New module dependency button on the Module dependencies screen to add information about this dependency. When the module is exported, dependency information is exported with it.

The Export Module Screen

Clicking on the Export Module button on the first Edit Module screen will promptly export the module to the file system.

Each time you export the module, the module minor reversion number will be incremented by one. That means that when we export the 0.5 version of the St. Nicholas Template module, it will become version 0.6. You can manually edit this field from the Edit Module (tool) button on the Edit Module screen.

The new module will be written to the server's file system, in the directory `$CATALINA_HOME/webapps/opencms/WEB-INF/packages/modules`.

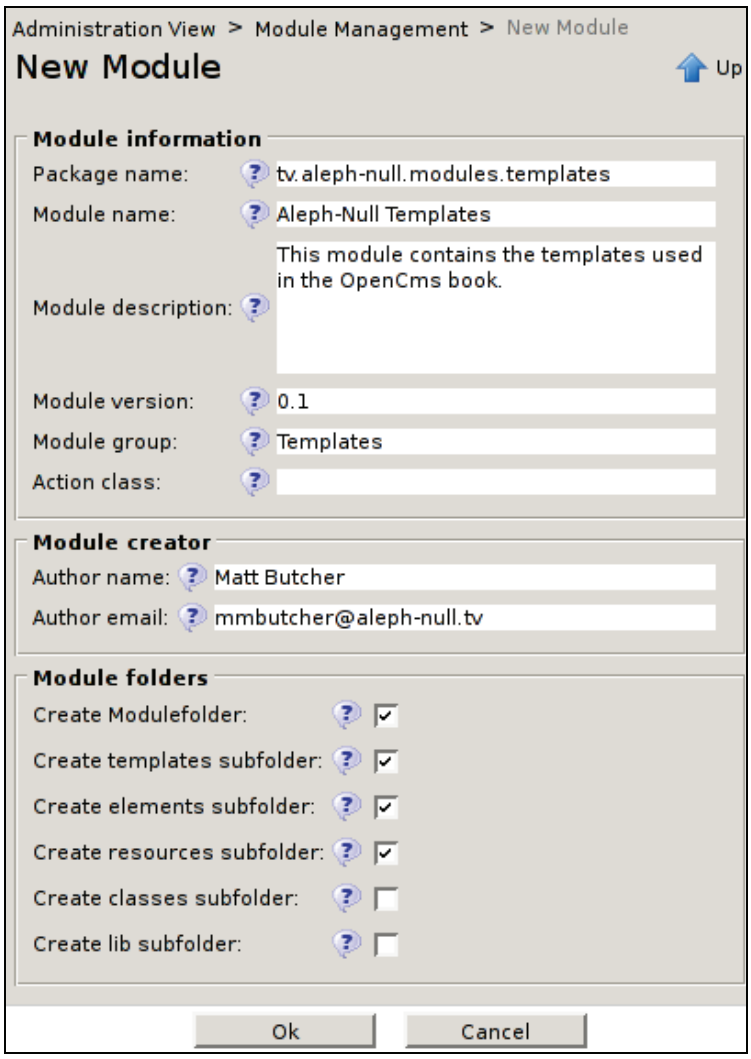
Deleting a Module

The seventh and last button on the first Edit Module screen is called the Delete Module button. Clicking on this button will immediately remove the module from OpenCms.

Creating a Module

Modules can provide a very convenient way of packaging your resources. In particular, templates, JSP files, and Java code ought to be stored in modules. As we begin working with templates, we will use a particular module for storing the new template files. We will create this module now.

Clicking the first button on the Module Management screen loads the New Module screen.



The Package name field is for the complete Java package name. The format is top-level domain name (e.g. `.com`, `.org`, `.edu`), specific domain name (e.g. `packtpub`), and then any number of additional package names. It is like a directory path that points to the particular project within your organization—it should be distinguishable from all other packages on the Internet.

The second field, Module name, should contain a human-readable title for the module, and the third field, Module description, should describe what the module provides.

The Module version field is used to specify the release level (and the level of stability) of the module. In general, the OpenCms community uses the following convention: `<Major version number>.<minor version number>.<minor revision number>`. So `1.0.1` indicates that it is the first major release of the module, and that one minor revision (which usually includes only bug fixes) has been made. If a significant feature has been added, the minor version number should be incremented. If the module has gone through any major changes (particularly in the API), then the major version number should be incremented.

Typically, until the module is stable, modules use `0` as the major version number. By default, then, OpenCms assigns our module the version number `0.1`—the first minor version of the module.

The Module group field is used to sort the modules by category. There are no established categories. Since our module is going consist primarily of templates, I have put it in the module group Templates.

Sometimes a module needs to run special code during certain OpenCms events (such as startup, shutdown, publish, or module installation). To do this, developers can create a class, specified in the Action class field, that handles particular events. Specifying the class name here will ensure that the class is loaded and executed by OpenCms.

> An action class must implement the `org.opencms.module.I_CmsModuleAction` interface.

In the Module Creator section, the Author name and Author email fields are used to provide information about you, the module creator.

Beneath the Module Creator section, there is a section called Module Folders, which has six checkboxes. These checkboxes determine which folders are automatically created in the VFS.

1. Create Modulefolder: If this is checked, a folder module will be created in `/system/modules`. The module folder is where we will put all of the module files, so this box should almost always be checked. The five other folders listed here are subfolders of this folder, so this checkbox must be checked for the others to work.
2. Create templates subfolder: If this is checked, a folder named `templates` will be created in the module folder. When we create new templates in Chapter 6, we will use this folder, so it is a good idea to create it now.
3. Create elements subfolder: If this is checked, a folder named `elements` will be created in the module folder. This folder is used for storing pieces of JSP code or text files that the module requires. Usually, you should check this box.

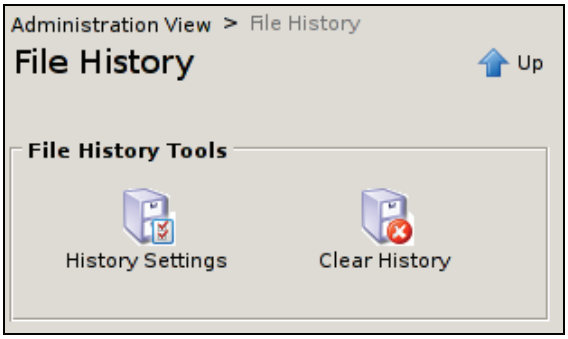
4. Create resources subfolder: If this is checked, a folder named `resources` will be created in the module folder. Resources are supporting files such as images, JavaScript, and cascading style sheets that templates use. If you create a templates subfolder, you should also create a resources subfolder.
5. Create classes subfolder: If this is checked, a `classes` folder will be created for this module, along with the requisite Java class structure subfolders. For example, in the case of our module, the path `classes/tv/alephnull/modules/templates` would be created. This option is useful to developers who intend to write custom Java classes or use custom resource bundles in OpenCms. Compiled class files stored in this folder will be loaded when OpenCms starts up. We will not be using this folder.
6. Create lib subfolder: If this is checked, a `lib` subfolder will be created. This folder is used to store Java archives (JARs) that contain compiled Java code and associated resources. Libraries in this directory will be loaded when the server starts.

Clicking Ok will create the new module.

You can now navigate to the new folder in the VFS. In the OpenCms toolbar, change Site from /sites/default/ to /, and change to the explorer view. Now navigate to the `/system/modules` folder. The new module folder (`tv.alephnull.modules.templates`) should be listed in this folder.

File History

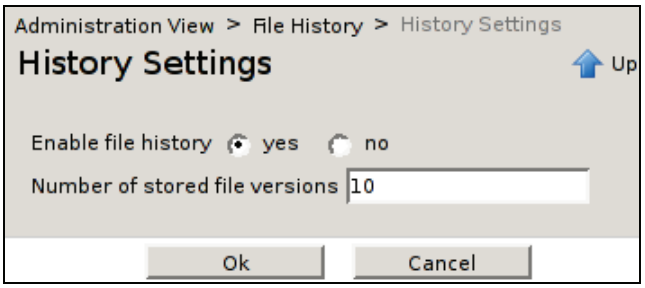
The fifth screen in the administration view is the File History screen:



OpenCms maintains history information for every file in the VFS. As we saw in Chapter 3, you can browse a file's history from the explorer view. You can even choose to revert to an earlier version of the file if you need. The File History screen can be used to access two other screens for dealing with the history features of OpenCms.

The History Settings

The first of these screens, History Settings, allows you to specify how the history facility behaves.



The first item, the Enable file history radio buttons, gives you the option of disabling history altogether. While this may conserve some space in the database, it will remove the safety net provided by the ability to revert to a previous version.

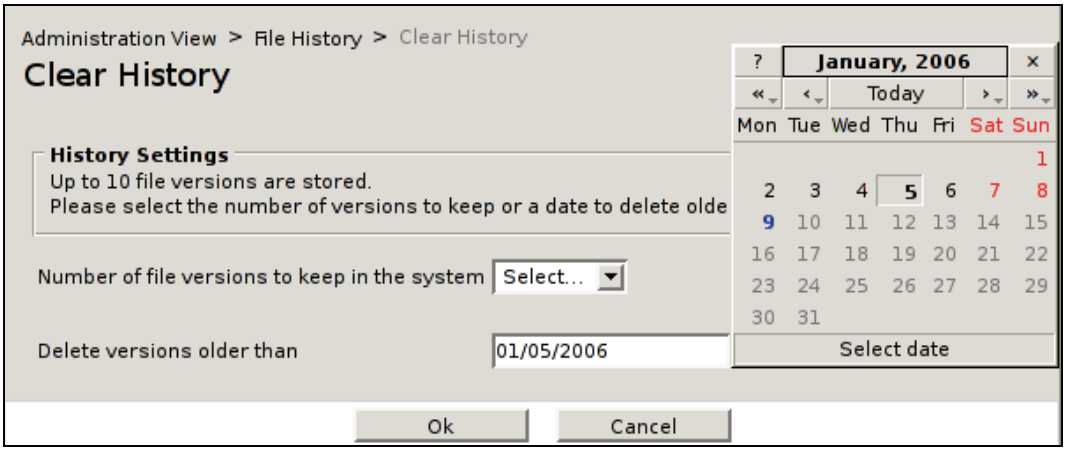
The second item, the Number of stored file versions field, allows you to specify how many older versions of the file ought to be retained. You will only be able to revert back to these versions. By default, this value is set to 10, meaning only the ten most recent versions of the document will be retained in the history database. Usually, this is sufficient.

> A history entry is created each time a resource is published, not each time a resource is edited. If the default value is set to 10, then the last ten *published versions* of the file will be saved.

Clicking Ok will save and apply the changes and return you to the File History screen.

Clearing the History

Clicking the Clear History button in the File History screen loads a screen that provides a way to selectively remove history information from OpenCms:



When clearing history, you have a number of different options. You can either specify how many of the existing history entries to keep, or you can specify that history entries prior to a specific date be deleted.

Looking at the History Settings screen, we saw how to set the number of stored file versions. On this screen, we can choose to delete existing history entries prior to a specific entry. Suppose that we set Number of stored versions in the History Settings screen to be 10 (the default). On the Clear History screen, the first drop-down list, Number of file versions to keep in the system, has the numbers 1 to 10. If we choose 3, for example, only the three most recent versions of each file will be kept in the VFS, and versions four to ten of the files will be deleted.

> This does not reset the number of stored versions to a value other than 10. The number of history items stored from this point on will still be ten.

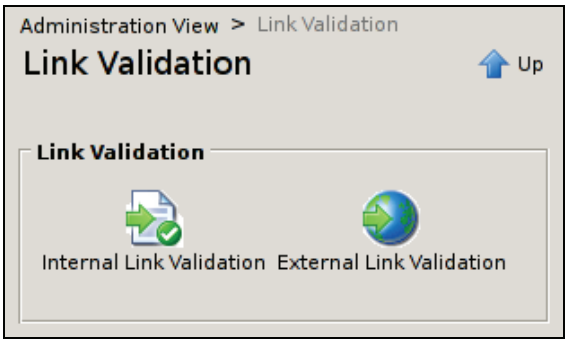
This is one way to reduce the amount of history data stored. But another way—one which is often more useful in the practical world—is to delete history information older than a specific date. You can do this with the Delete versions older than field. You can either enter a date into the text input field in the format `mm/dd/yyyy` (for example, 01/06/2006), or you can click the calendar icon and select the desired date from the pop-up calendar widget.

Once you have selected one of these two options, clicking the Ok button will begin the deleting process. OpenCms will present a screen showing the progress. Once the process is complete, you can click the Ok button at the bottom of the screen to return to the File History screen.

When you installed OpenCms, historical versions of all of the OpenCms default content were created. Clearing the history may result in clearing versions of those files. This is OK, as there is not necessarily a reason to keep this history information in the database, but it may be surprising to see that the history has been deleted for files that you have never edited.

Link Validation

The sixth screen in the administration view is called Link Validation. This screen contains tools for verifying hyperlinks contained in Page-type files in the VFS.



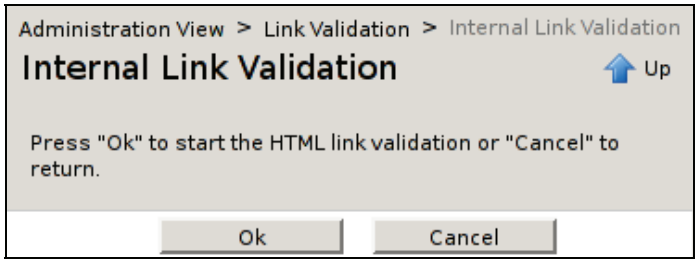
This screen has two buttons, Internal Link Validation for verifying links to resources on the same server and External Link Validation for verifying links to some other server.

The Link Validation screen is one of only two tools available to anyone in the Users group. (The other is the Gallery Overview screen.) Users who are not administrators or project managers can access the Link Validation screen and validate links on their own.

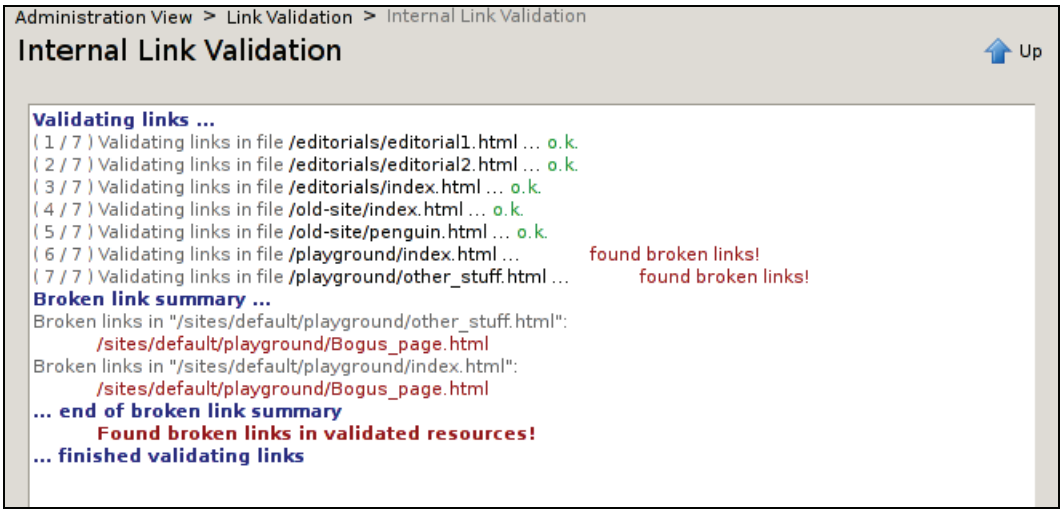
If you are working with the Online project, the links for currently published resources are checked. If you are working with the Offline project or one of its sub-projects, links in documents in the Offline project are checked.

Internal Link Validation

Clicking on the Internal Link Validation button brings up the screen for testing links whose destination is within this instance of OpenCms. This validation tool will make sure that the link points to a resource that exists in the VFS. It tests links to images and downloads as well as links to other files.



All you need to do to start the verification process is click Ok. OpenCms will then search all of the modified folders in the VFS and check all of the changed files looking for broken links. Progress is displayed on the screen. If any file contains a broken link, an error message will be displayed:



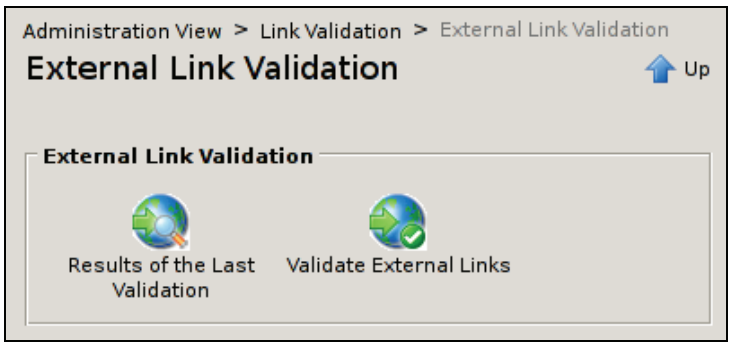
In the screenshot above, the link validator warns that the files `/playground/index.html` and `/playground/other_stuff.html` both contain links to the non-existent page `/playground/Bogus_page.html`. To fix this, I would have to switch back to the explorer view, edit the files, and remove or change the problematic link. Returning to the administration view, I would be prompted to revalidate the links.

Fortunately, OpenCms does not force you to fix the offending files at this stage, but when you attempt to publish the files, OpenCms will ask whether you want these files to be published when they have broken links.

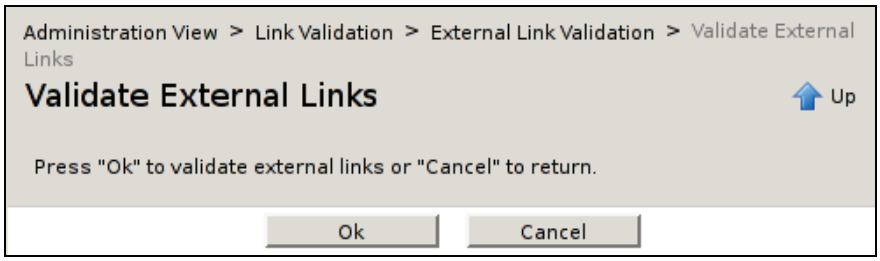
Clicking the Ok button at the bottom of the screen returns us to the Link Validation screen.

External Link Validation

The External Link Validation screen is used to search all the Page-type documents for links that point to a remote server and tests all of these links to see whether they are valid. It has two buttons:



Clicking the first button, Results of the Last Validation, will display a summary of the broken links found last time the validation was run. Clicking the second button, Validate External Links, will run the validation tool:



Clicking Ok will begin the link check. OpenCms will display progress information, warning you at the end whether any broken links have been found.

> A failed link may be the result of network connectivity issues or temporary outages on the remote site. OpenCms has no control over this and simply reports the failed request.

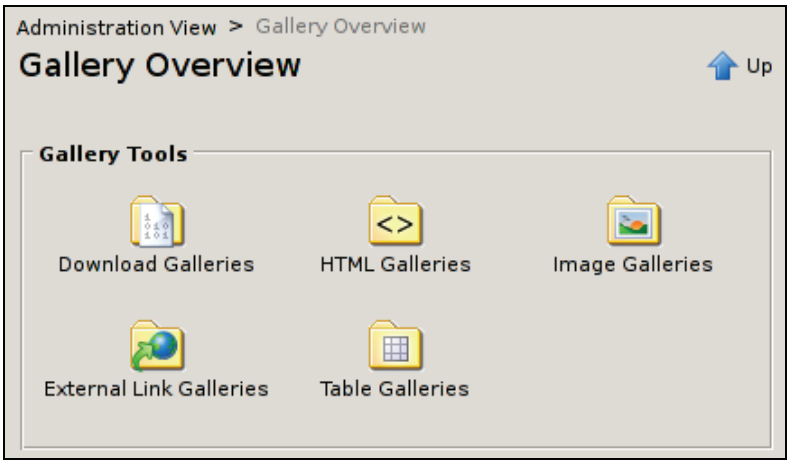
To fix a broken external link, you should switch to the explorer view and edit the file, locating the offending link and deleting or modifying it.

While OpenCms checks internal links during the publishing process, it does not check external links. Thus, it is a good idea to periodically run this validation process in order to catch any broken or obsolete links.

Clicking Ok at the bottom of the results screen will take you back to the External Link Validation screen.

Managing Galleries

The Gallery Overview screen is a convenient interface to access all the galleries in the VFS. As we saw in Chapter 3, galleries are a special type of folder (extended folder) used to store a specific type of information. For example, image galleries are used for storing files of type Images, and external link galleries are used to store External Link files. Galleries can be stored anywhere in a site. In some respects, this can be very convenient, allowing editors to locate galleries in the same part of the file system as the content which uses them. But managing a decentralized collection of galleries would be difficult without the tools accessed from the Gallery Overview screen.

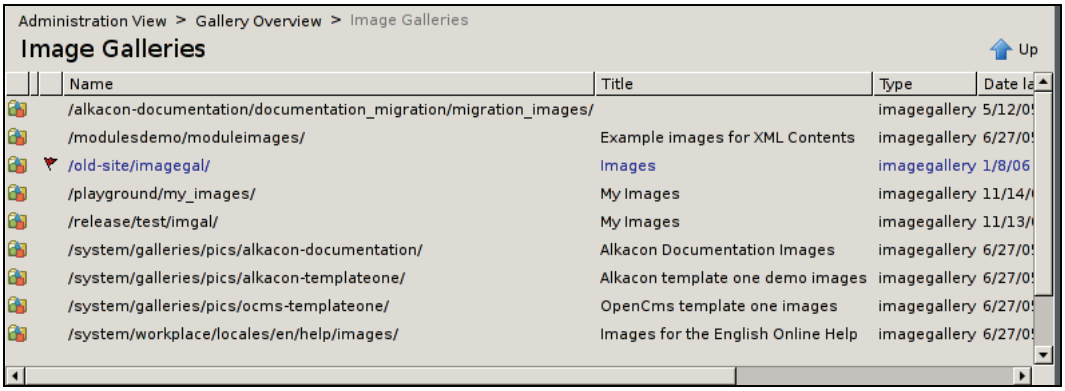


There are five buttons on this screen (one for each type of gallery), Download Galleries, HTML Galleries, Image Galleries, External Link Galleries, and Table Galleries.

> Gallery Overview is one of only two tools available to anyone in the User group. (The other screen is Link Validation.) A user need not be an administrator or project manager to use this screen to access the galleries. However, the user is still constrained by the permissions of the galleries, so that he or she cannot view a file through the Gallery Overview screen that he or she could not view in the explorer view.

All five of these tools function identically, so for the sake of brevity I will only cover one. The same procedures can be applied to the other four gallery tools.

Clicking on the Image Galleries button will bring up an explorer-like view of all of the image galleries in the site.



The main difference between this view and the standard explorer view is that this view collects and presents information on all of the image galleries in the site. This is reflected in the information displayed in the Name column. Here, the path (relative to the site `/default/site` in this case) is displayed instead of just the name of the file.

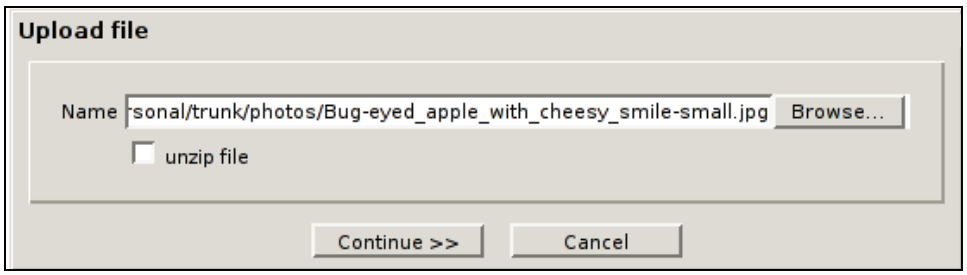
As in the explorer view, clicking on a gallery's icon will bring up a context menu for that gallery. Clicking on the Open gallery button will open a window with the gallery viewer.

The drop-down list at the top of the gallery displays the name and path of the current gallery, and you can click on it to choose a different gallery. Beneath this drop-down list are the button for uploading new images into the gallery and the search input box (for searching within the gallery).

Beneath that is a list of all of the images in the gallery. This gallery, the playground gallery, has only one image. Clicking the image in the list will display it in the image viewer.

When an image is loaded in the image viewer, a small toolbar will also be loaded, giving you the option to publish the image (button with two arrows), delete the image (button with red X) or change the name of the image (input box and green pencil button).

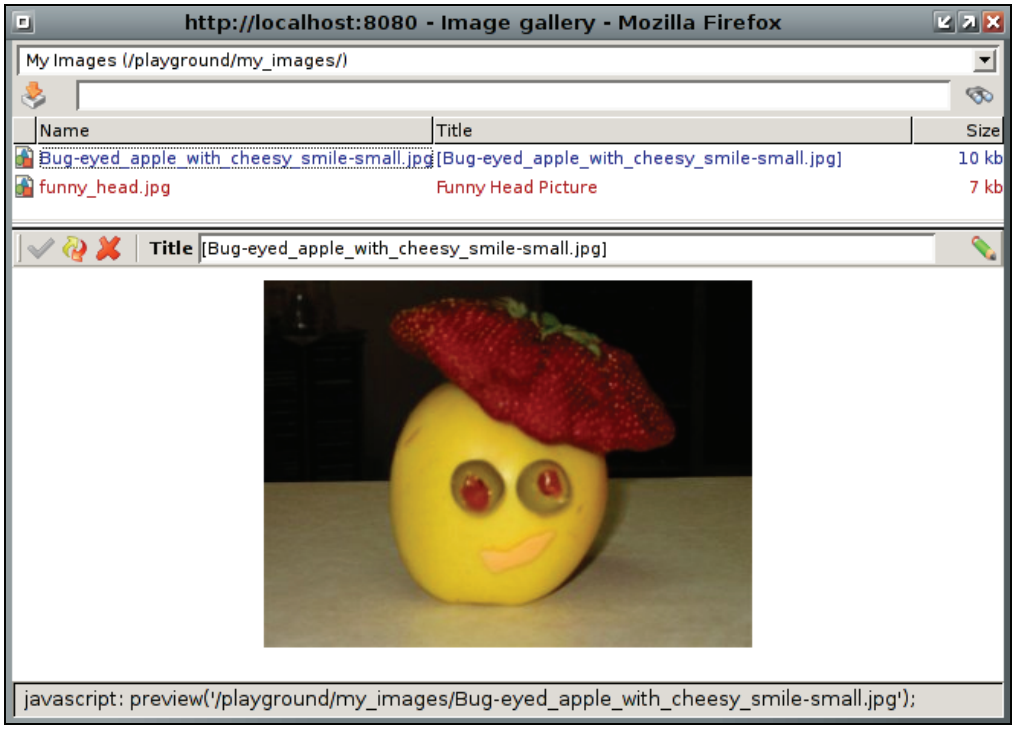
You can upload new images to a gallery from within the image gallery viewer. To do so, click on the Upload new file button (an orange arrow pointing down at a gray box) located just beneath the gallery drop-down list at the top. You will be prompted to select the image file that you want to upload to the server from your local file system.



Underneath the Name field there is a checkbox called unzip file. If you are uploading a ZIP file containing one or more images, you should check this box.

> If you have a large number of images to upload, you may find it convenient to add them all to one ZIP archive and then upload the archive file. OpenCms will unzip the file and add all the images to the current gallery. You will not be prompted to edit the properties of each file.

Click the Next button to continue. From here, OpenCms will take you through the file creation wizard that we examined in Chapter 3. Once you have edited the file's properties, you will be returned to the gallery manager, and the new image(s) will appear in the manager's file list.



While the other screens differ in minor details (namely, in what information they show in the previews), they function in almost exactly the same way as the Image Galleries screen.

What are the other galleries for? Here's a brief explanation of each:

Download galleries are used to store binary files, such as word processor documents, PDF files, or programs, that visitors can download to their local computers.

HTML galleries provide a place to store snippets of useful HTML code that editors can share and reuse.

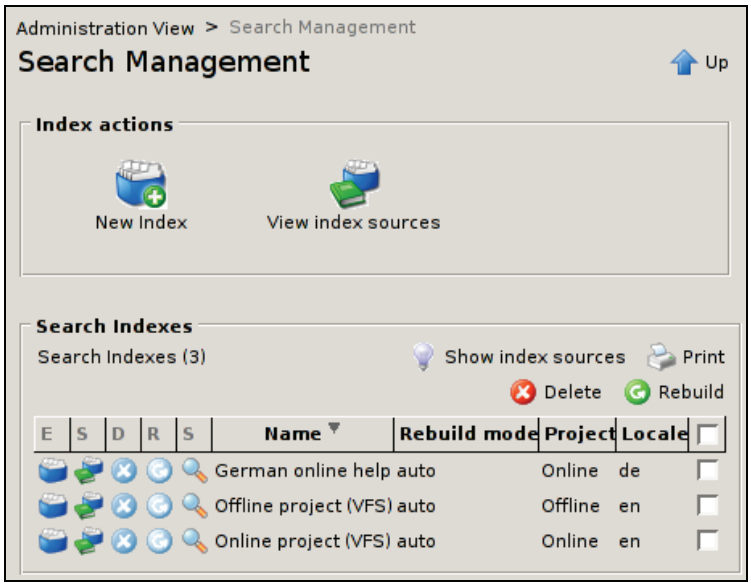
External link galleries are for storing links to external websites. Using one of these galleries saves you having to retype long URLs every time you want to create a link, with the additional advantage that OpenCms automatically verifies the connection and warns you if the link goes bad.

Table galleries are used to keep a stock selection of HTML tables to use in different documents.

We will now move on to the Search Management screen.

Searches and Indexes

OpenCms comes with a built-in search engine. The Search Management screen is used to access tools that tune this search engine.



The Search Management screen is divided into two sections. The top section, Index actions, is used to access tools for managing the search engine's index files. The bottom section, Search Indexes, lists the existing indexes.

There are two buttons in the Index actions section, New Index and View index Sources. Before explaining these tools, we will see briefly how searching works in OpenCms.

The most efficient search engines read through a collection of data (such as a website or a group of documents) and create a special catalog of all of the words in the collection and which documents these words appear in. This catalog is called an **index** because it performs a similar task to the index in the back of a book—it correlates a word to the locations where it appears.

More advanced search engines, such as the one in OpenCms, store additional information such as how many times a word appears in a document and where in the document the word appears. The search engine can use this additional information to return an ordered list of documents with the best matches first.

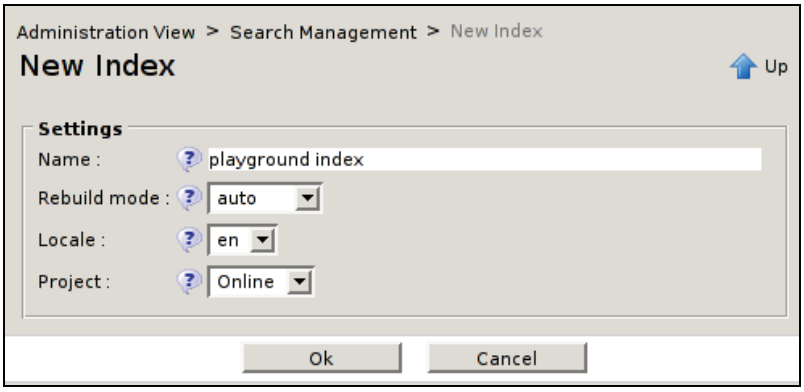
Once a search engine has an index, it can perform searches. When a request comes in for a new search, the search engine breaks down the terms in the search request, examines the indexes, and returns a list of matching documents.

> The search technology included with OpenCms relies upon the **Lucene** search engine library, an advanced Java library for efficiently searching large numbers of documents. Lucene is Open Source software maintained by the Apache Foundation. This search engine can be found at `http://lucene.apache.org`.

The tools accessed from the Search Management screen provide functions for managing your search indexes and for running test searches.

Creating an Index

Most of the time, the built-in indexes will be sufficient, but you may sometimes want to create a separate index. For example, you may want to search only a selected area of your site. You can create a new index using the New Index screen.



We will create a new index for searching in the `/playground` folder only.

The Name field should contain a descriptive name of the index's contents. The Rebuild Mode drop-down list has two options, Auto and Manual. If it is set to Auto, OpenCms will rebuild the search index as needed. If it is set to Manual, then we will need to run the index rebuild by hand. (We will discuss this below.) Usually, Auto is what you want.

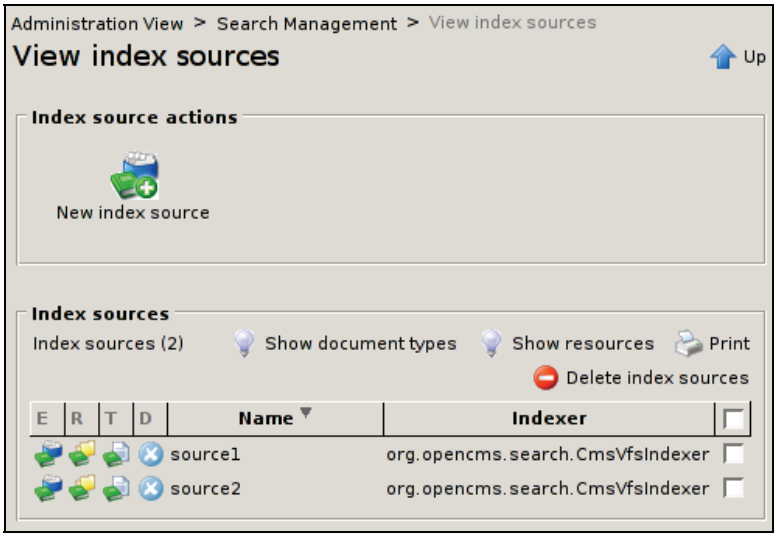
You can use the Locale drop-down list to select which language to index. You can only use one language in each index, so if you desire multiple language support, you will need to create multiple indexes.

The Project drop-down list determines which project will be indexed. If this is set to Online, the index will only search (and therefore only return hits for) published documents. If site visitors are going to use this search index, you want to use this option. If you set this to Offline or to another custom project, OpenCms will build an index against the unpublished files in the VFS. You should not allow site visitors to use this index.

Clicking Ok will create the index and return you to the Search Management screen. You will now see the new index listed in the Search Indexes section.

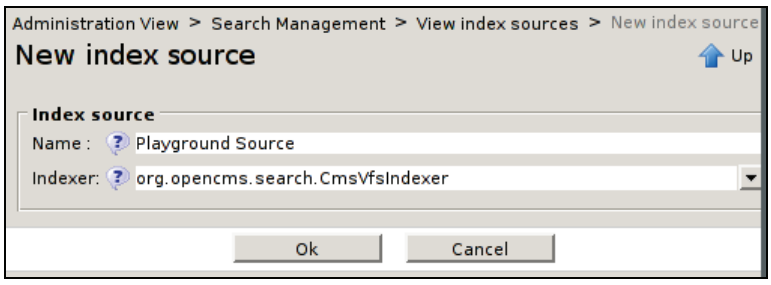
Index Sources

An index can contain information about different areas of the VFS and also about different types of files. Clicking on the View index sources button will take you to the main management interface for overseeing the sources.



At the top of the View index sources screen is the Index source actions section, which has one button, New index source. In the Index sources section there is a list of all the currently configured index sources. By default, there are two pre-configured sources, Source1, which indexes all files of all types in the `/sites/default` folder, and Source2, which indexes the German-language help information. Source1 is used by both the Online and Offline search indexes, and Source2 is used by the German online help search index.

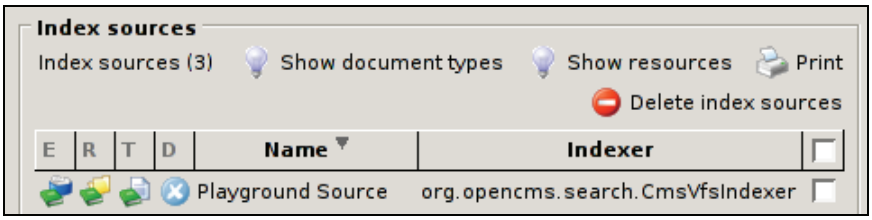
We will create a custom index to index just the `/playground` folder by clicking on the New index source button in the Index source actions section.



Put a descriptive, human-readable name in the Name field. The second field, Indexer, is a drop-down list. By default, it has only one entry, org.opencms.search.CmsVfsIndexer. This is the name of the Java object that will do the actual indexing. Developers can create their own custom indexers if they need, but most of the time the CmsVfsIndexer is sufficient.

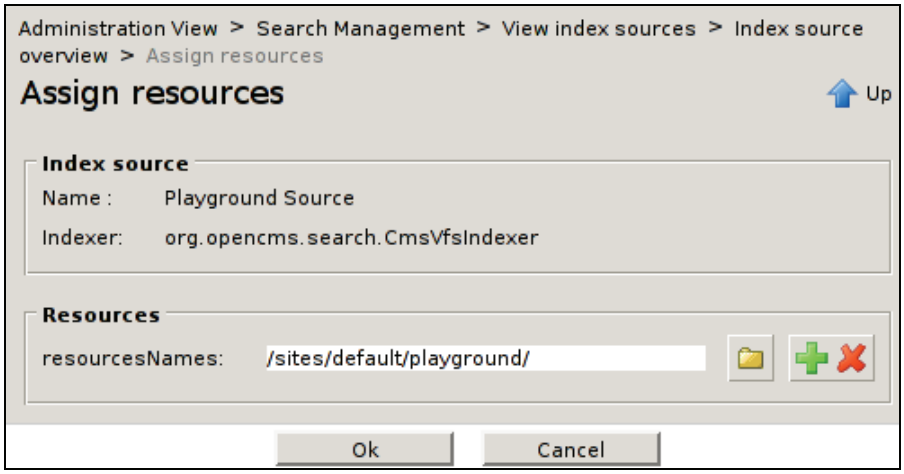
Clicking Ok will create the new index source and return you to the View index sources screen.

Now our new index shows up in the Index sources list at the bottom of the page.



At the beginning of the line for our new Playground Source entry, there are four buttons. The first button, a green book in front of a blue folder, loads the Edit index source screen, which allows you to change the indexer (which we set when we created the source).

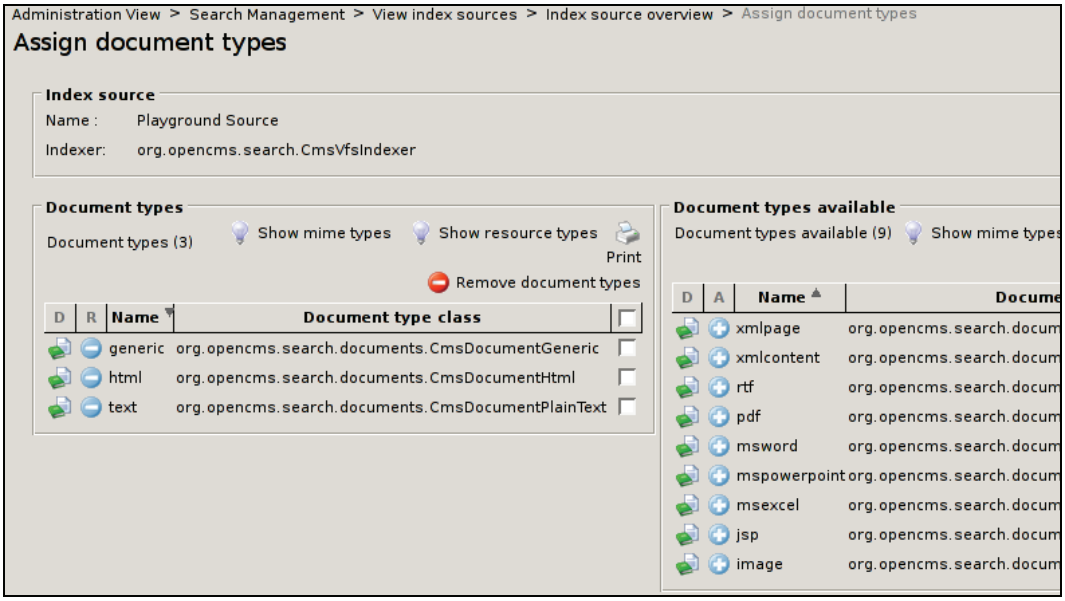
The second button, a yellow folder behind a green book, loads the Assign resources screen.



With this tool, we can specify which areas of the VFS this source will index. For our current project, we just want to add one directory, `/sites/default/playground`, but you can add as many resources as you want. (When a folder is added to a source, all of its subfolders are also automatically indexed.)

Clicking Ok adds the resources to the source and returns you to the View index source screen.

Clicking the third button at the beginning of an entry in the Index sources list, a white page behind a green book, loads the Assign document types screen.



We have seen that it is possible to specify exactly which types of documents are indexed. That is done here. Using this interface you can set which types of documents will be indexed in this source. Usually, you will want at least HTML pages and generic documents to be indexed.

To add new types, check the desired boxes in the Document types available section and then click the Add document types button. The types will then show up in the Document types section. (generic, html, and text are shown in the screenshot above.) These document types will now be indexed.

Click View index sources in the breadcrumb trail at the top of the screen to return to the View index sources screen.

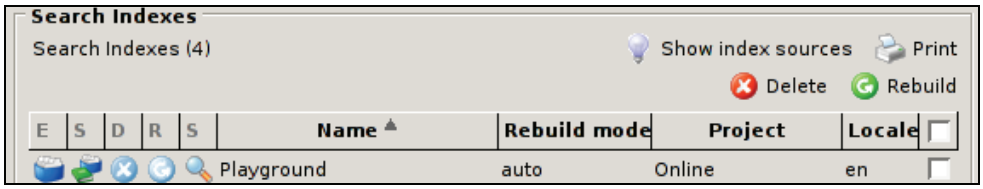
The last of the four buttons in an entry in the Index sources section of this screen is a white X on a blue circle. Clicking this will delete the source.

After the buttons is the name of the index source. If you click on this, OpenCms will display an overview of the index source settings for that source.

At this point we have looked at all of the index source tools. Now click on the Search Management item in the breadcrumb trail, and we will look at the rest of the search tools.

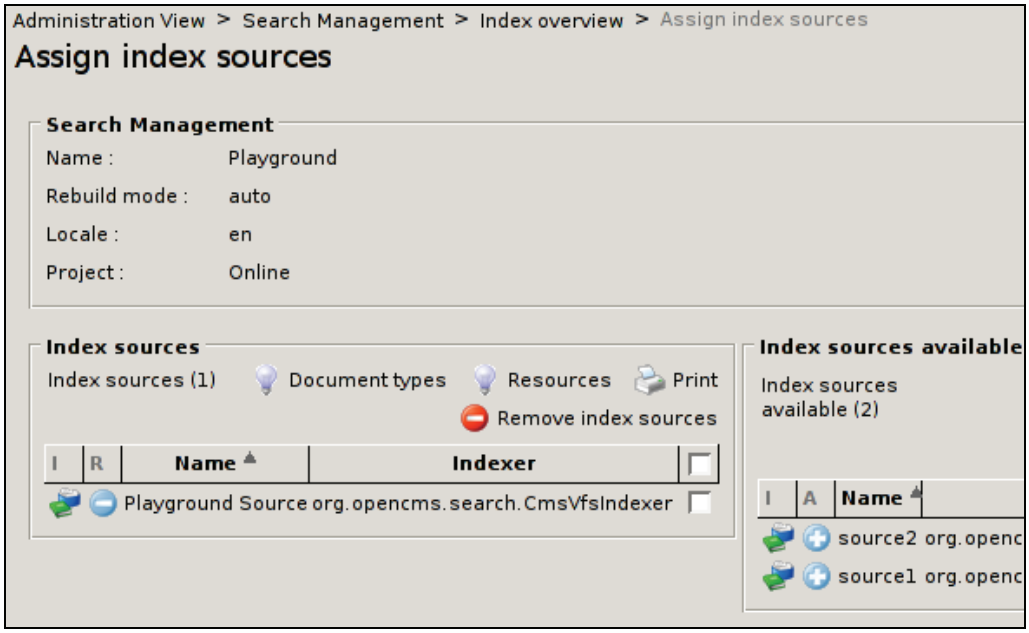
Managing Search Indexes

We will now look at the Search Indexes list at the bottom of the Search Management screen.



Previously, we created a new index called Playground. It should be displayed at the top of the Search Indexes list. At the beginning of each entry in this list, there are four buttons. The first, a blue expanding folder, loads the Edit screen for the index. There you can change the rebuild mode, the locale, and the project for that index. (We set these when we created the new index.)

The second button, a green book in front of a blue expanding folder, loads the Assign index sources screen.



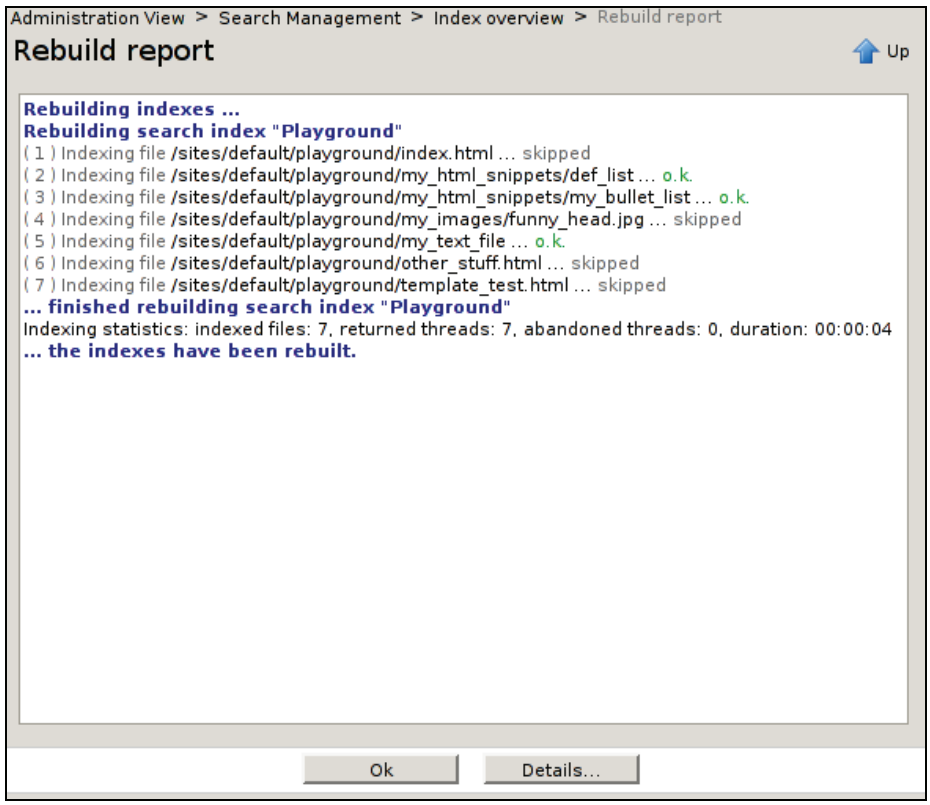
This screen allows us to specify which sources to use with this index. To add the Playground Source source that we created earlier, we can select it from the list on the right and click the Add index sources button. This will add the item to the Index sources list, which indicates that this index is now using our new source.

Click on Search Management in the breadcrumb trail to return to the Search Management screen, where we will continue looking at the Search Indexes list.

Next to the button that launched the Assign index sources screen, there is a blue circle button with a white X. Clicking this will delete the index.

The next button, an arrow wrapped in a circle on a blue circle, is the button for rebuilding an index. Since we just created this index, it is a good idea to run the rebuild right now.

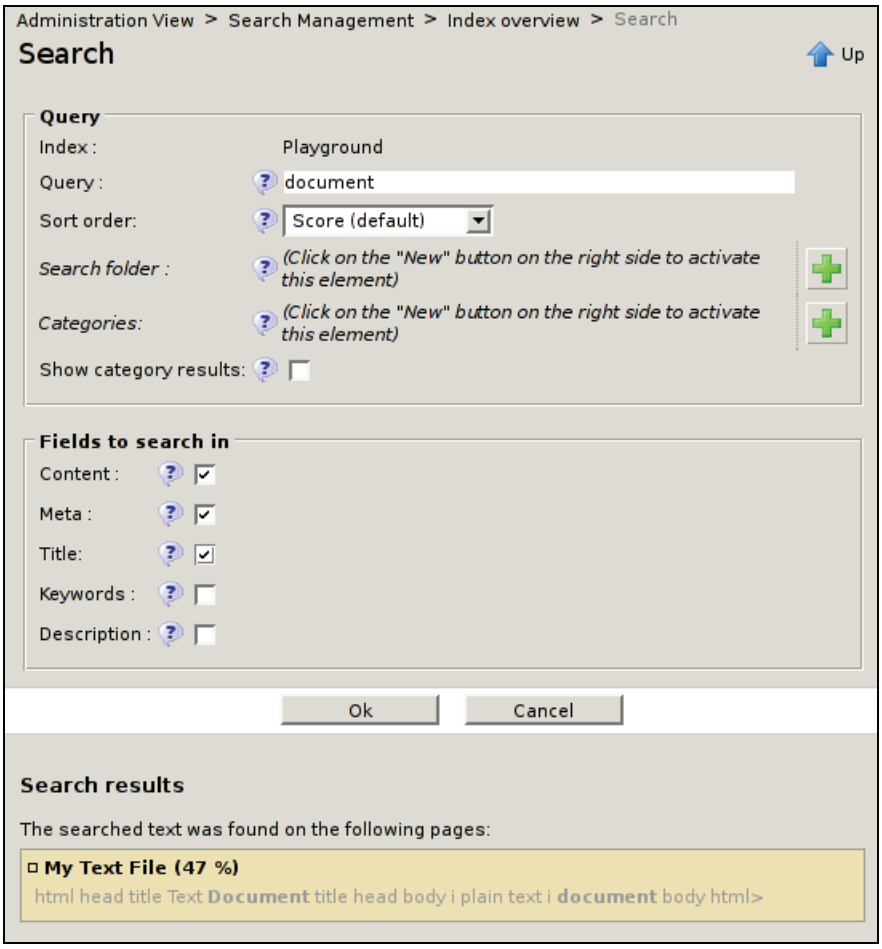
Once you have pressed the button, you will be prompted to confirm your choice. If you click Ok, the search index will be rebuilt. This can take a long time. Once it is complete, you will get a report of the files indexed.



Clicking Ok will take you to the Index overview screen for the rebuilt index. This page displays information about this index. All of the functionality available on this screen is also available on the Search Management screen. (You can get to the Index overview screen from the Search Management screen by clicking on the name of the desired index.)

Click on Search Management in the breadcrumb trail at the top of the screen.

The last item in an entry in the Search Indexes list is a magnifying glass. Clicking on this will load a search screen for searching the index.



Using this tool, you can search the index for a specific search string (one or more words). Note that some words, like 'a', 'an', and 'the', are not added to the search index. They are too common, and including them would just bloat the index.

Type the search string in the Query field. Using the Sort order drop-down list, you can specify whether the returned results are organized by Score, Date created, Date last modified, or Title.

You can also specify which folders and categories you want to search, and you can even organize results by category (by checking the checkbox at the bottom of the Query section).

In the Fields to search in section, you can specify which parts of the documents to search. Content is the body of the document. Checking Meta is the same as checking Title, Keywords, and Description.

Clicking Ok will run the search, and the results will be displayed at the bottom of the screen. In the example above, you can see that the document My Text File contained the search string document.

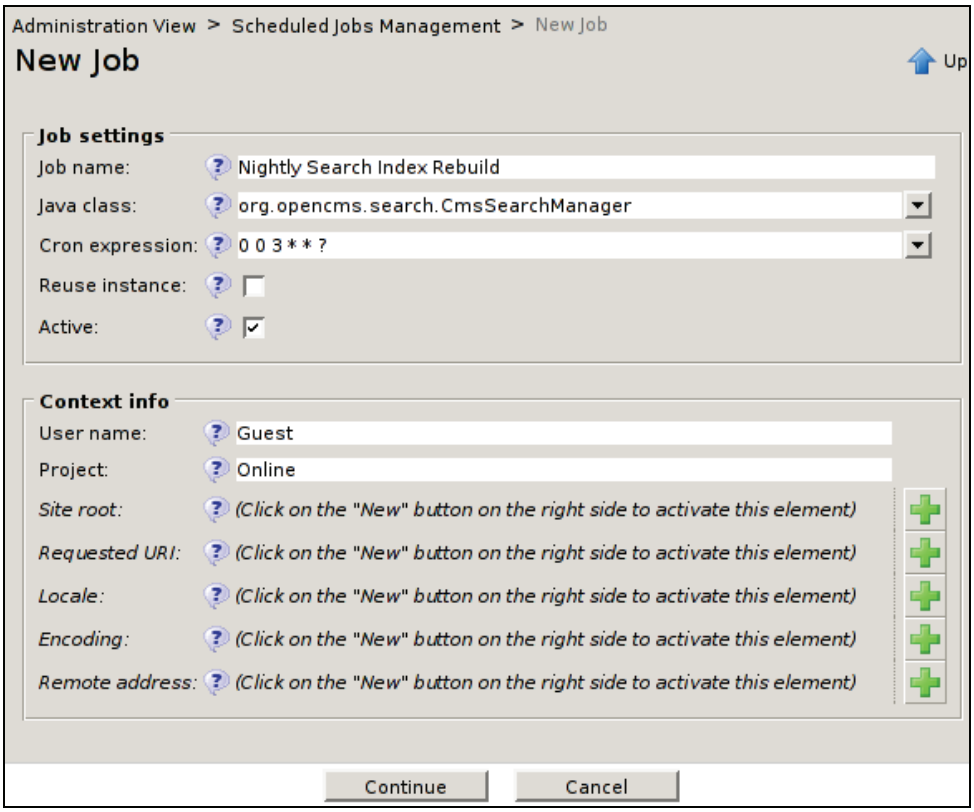
The tools we have looked at here are for administrators. How do users search? By default, a practical search tool is not included in OpenCms. Instead, the OpenCms developers include JSP hooks for creating your own search interface, and some custom code must be written to add search functionality to your OpenCms pages.

Scheduled Jobs

The Scheduled Jobs Management screen in the administration view provides an interface for scheduling specific tasks to run at specific times. (This is like Cron in UNIX/Linux or the Task Scheduler in Windows.)

> The Scheduled Jobs Management screen is designed for developers and can be difficult to use for those who do not have a working knowledge of the OpenCms code. You may need to consult the developer documentation for your modules before configuring a particular job.

Clicking on the New Job button will load the job creation tool.



This screen has two sections. On the top is the Job settings section, which specifies what task OpenCms will execute and when. Beneath this is the Context info section. Fields in this section relate to the task itself. They specify the environment in which the task will execute.

The first field in the Job settings box is Job name. This should be a human-readable name for the task. In this case, I am configuring a task that will automatically rebuild the search indexes once a day, so the name reflects this. (See the previous section of this chapter.)

The second field, a combo-box, is labeled Java class. This field takes a fully qualified Java class name. You can either type the class name in by hand or select a class from the drop-down list.

> The Java class must be in OpenCms's class path and must implement the right Java interface, `I_CmsCronJob`. Usually, custom Cron classes are stored in modules.

The drop-down list contains several useful Java classes that do things like automatically publish or export projects, automatically validate links, and monitor memory. Our job is to have the search engine automatically rebuild the index files once a day. For that, we will use the `org.opencms.search.CmsSearchManager` class.

The next field is for scheduling the time that the job will run. You can either type a cron-formatted time string into the field, or you can choose one of the pre-configured times from the list. We will choose 0 0 3 * * ?, which instructs OpenCms to execute the job every day at 3:00 AM.

> Of course, the pre-defined cron expressions that OpenCms provides may not always suit your needs. See Appendix A for detailed information on writing your own Cron expressions.

The Reuse instance checkbox determines whether the class above will be loaded once and reused or loaded and used each time it is executed. If a job runs frequently, you may want to check this box, but since the script is only running once a day, we don't need to worry about reusing the same instance—the amount of time that it takes to create a new instance is negligible.

> If you are using a custom job class, you may need to check with the class developers to find out whether the class can be safely reused. If you don't know, it is safer to leave the Reuse instance box unchecked.

If the Active box is checked, then the job will be scheduled to run. To prevent a job from running, you can uncheck this box (or deactivate the job from the Scheduled Jobs Management screen).

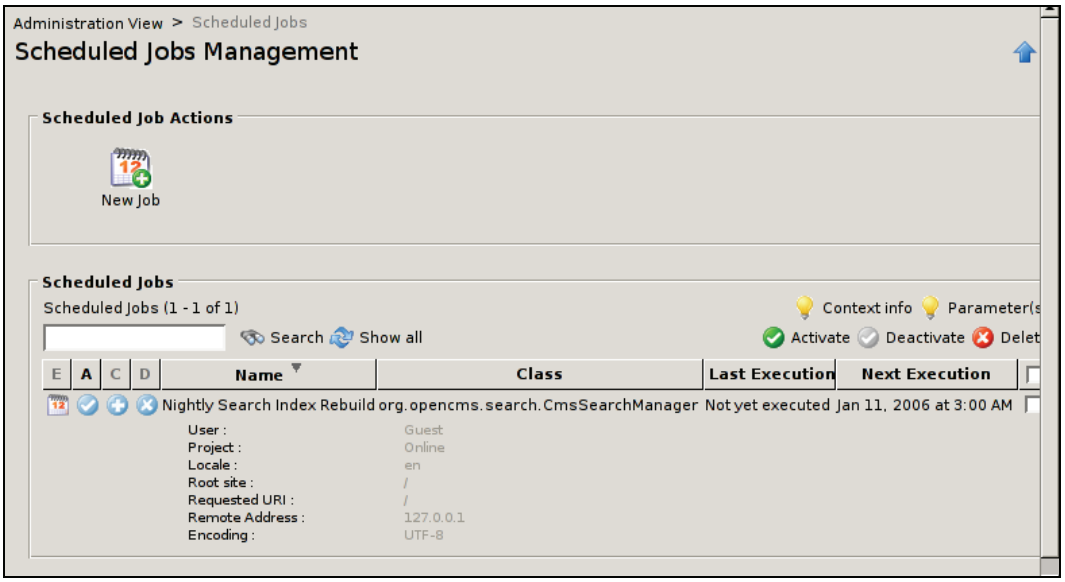
The fields in the Context info box determine the environment that the job will run in. For example, the first field, User name, allows you to specify what user will perform the task. OpenCms will log in as that user and will be restricted to the files that that user can access.

The Project field determines which project the job will run in. By default, it is the Online project, but we could set it to any existing project in OpenCms. For our task, it does not matter. (All of the index files will be rebuilt.)

The next five fields, which are optional, provide additional information about the environment in which the job should be run (such as which site should be used).

Clicking Continue at the bottom will take you to a new screen displaying the Parameters field for this job. Some jobs have optional parameters that must be set for the job to run correctly. You will have to consult the documentation for your job class to find out what

parameters are required. In our case, no parameters are required. Clicking on Ok will place the job in the queue and return you to the Scheduled Jobs Management screen. You should now see the new task in the Scheduled Jobs table at the bottom of the screen:



By clicking on the Context Info and Parameters light bulb icons, you can see additional information about all of the entries in the Scheduled Jobs table. At the beginning of each row are four icons. Clicking the calendar icon will load a page for editing the job. The checkbox will activate or deactivate the job. Only activated jobs are scheduled to run. Deactivated jobs will never run. If you click the plus icon, OpenCms will create a copy of the existing job, which you can modify to create a second, similar job. This can save time when creating a number of similar tasks (for example, publishing multiple projects). Finally, clicking the X icon deletes the task.

> Job management information is saved in the server's file system in the file `$CATALINA_HOME/webapps/opencms/WEB-INF/config/opencms-system.xml`. You can edit this file by hand, but do so carefully. Errors could prevent OpenCms from functioning correctly.

Flex Cache Administration

OpenCms uses a sophisticated caching mechanism to speed up processing. Without a cache, OpenCms would have to do a number of database lookups, compile one or more pieces of code (main JSP pages) and then execute the newly compiled code, and this can be a time consuming, resource-intensive process. To minimize the amount of repetitive work, OpenCms caches certain pieces of compiled code. Then, when the page is requested again (under similar conditions), OpenCms can simply execute the cached code instead of retrieving the uncompiled code from the database, compiling it, and executing it. The Flex Cache Administration screen in the administration view lets you tune or clear the cache.

This screen is complicated, and the following description is necessarily technical. If this explanation gets too tedious for you, you may want to skip to the next section of this chapter. If you just need to clear the entire cache, just click the Clear cache keys and variations completely button.

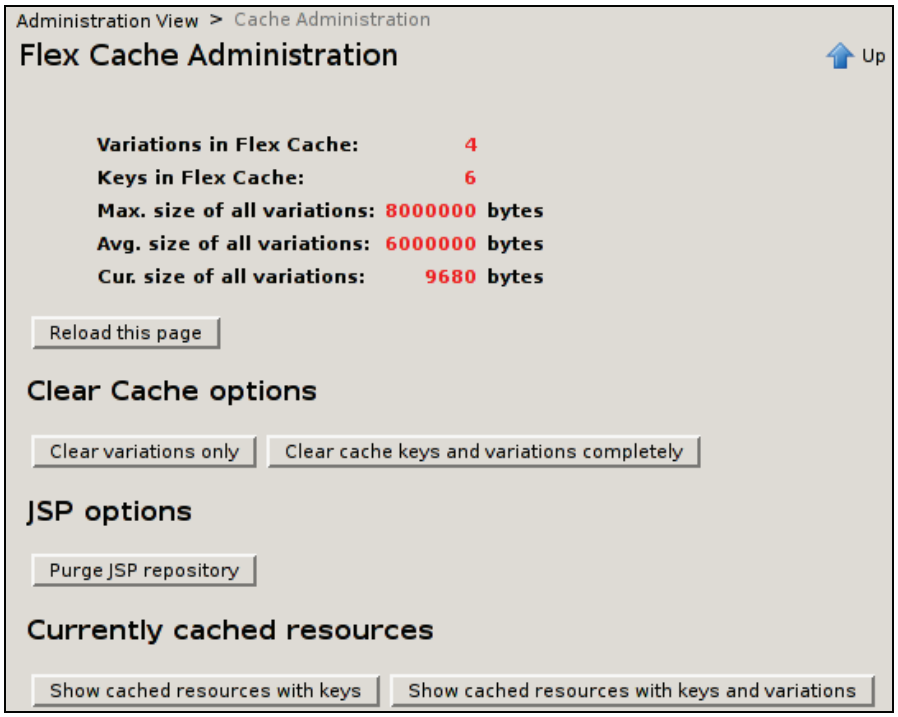
Before looking in depth at the FlexCache tools, I will explain the basics of the FlexCache. When a servlet engine (such as Tomcat) encounters a JSP file on the file system, it compiles it into a Java class file. It then executes this class file. But OpenCms does not store files in the server's file system—it stores them in the VFS.

To make JSPs work within the VFS, the OpenCms developers needed a way to efficiently store JSP pages in the file system where the JSP interpreter could find them. They developed the **Flex** package to accomplish this task. When a JSP page is requested, Flex writes the VFS file into the real file system, storing it in the `$CATALINA_HOME/webapps/opencms/WEB-INF/jsp/online` folder (or `offline` if the request is for a resource in one of the unpublished projects).

To expedite the process of rendering pages, the **FlexCache** stores not only the JSP file, but also any elements included in the JSP file, including plain text files and XML templates. Using a sophisticated algorithm, OpenCms manages the cache, keeping its size down and providing JSP developers with a variety of cache control options. The Flex Cache Administration screen provides tools to manually manage the cache. The complexity of the cache is mirrored in the number of functions available in the cache management screen.

The main prerequisite for understanding the Flex Cache Administration screen is knowing how cached items are stored. Each cached element has a *key*, which identifies what resources in the VFS are being cached. Each key may have multiple *variations*, where each variation indicates that the particular resource was loaded by another resource. This can be the case when two different JSP files both include a common third file.

Now we can look at the Flex Cache Administration screen.



The first two fields, Variations in Flex Cache and Keys in Flex Cache, provide information about how many items are currently in the cache. The more content, and the more active the site, the higher these numbers will be.

The Max. size of all variations and Avg. size of all variations fields display the configuration parameters for the size of the cache. OpenCms will try to keep the cache around the average size but will never allow the cache to grow larger than the maximum size.

> To change the average and maximum sizes, you will need to change the appropriate settings in the `flexcache` section of the `$CATALINA_HOME/webapps/opencms/WEB-INF/config/opencms-system.xml` file in the server's file system. You will need to restart the server for the new settings to take effect.

The Cur. size of all variations field shows how much space is currently being used. (In the screenshot above you may notice that the current size is well below the average size. That is because not many requests have come into this server.)

Here is a summary of the function of each of the buttons.

1. Clear variations only clears the data associated with each key, but leaves the keys in the cache. The key tells the location of the file in the VFS. The next time a resource is requested, OpenCms will reuse the key, assuming that the cache properties for that file have not changed. This gives you a slight performance boost as the cache is rebuilt. Most of the time, however, it is better to use the next option instead.
2. Clear cache keys and variations completely clears everything from the cache. No information about resources in the VFS is saved.
3. Purge JSP repository removes all cached JSP elements from the `$CATALINA_HOME/webapps/opencms/WEB-INF/jsp` folder (both the `online` and `offline` subfolders). The next time a JSP is requested, it will be copied from the VFS into the real file system again. This option is useful during development, when JSP files are frequently changing and cached copies quickly become outdated.
4. Show cached resources with keys and Show cached resources with keys and variations both display the items in the cache. If you click the latter button, it will list not only the keys but also the variations of each cached object.

> By default, resources in the Offline project are not cached by the FlexCache. During development, you may want to enable caching for the Offline project. To do so, edit the `opencms-system.xml` file in the `$CATALINA_HOME/webapps/opencms/WEB-INF/config` folder by setting the value of `cache-offline` to `true`. You will need to restart Tomcat after changing this file.
>
> Once you have made this change, the Flex Cache Administration screen can be used to specify *which* cache (Online or Offline) you want to clear.

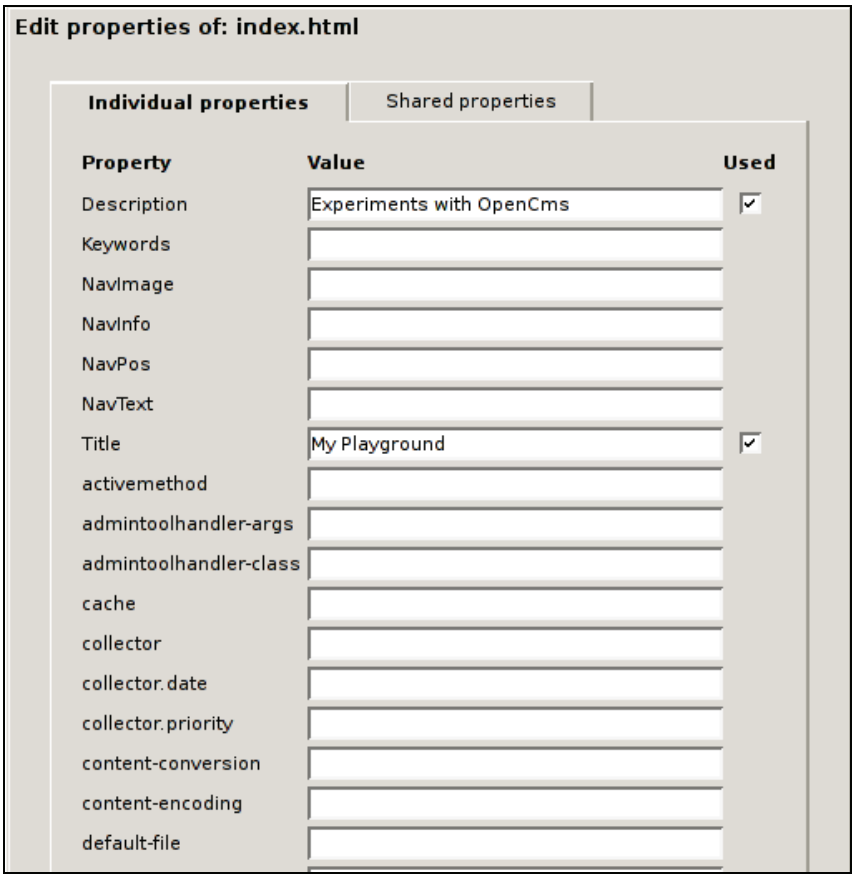
The caching system in OpenCms is complex, but its complexity pays off in the form of tremendous performance improvements. OpenCms includes documentation on the intricacies of caching. The cache system is usually self-sufficient, but if it does not meet your needs, you may wish to consult the built-in documentation.

Content Tools

The Content Tools screen in the administration view is used to access two different kinds of tools, property tools and element tools.

The property tools deal with the data that is attached to files—the metadata or properties. For example, the title, description, and keywords for a document are not stored in the text of the document. Instead, they are stored in properties that are linked to the document.

To take a look at the properties of a file, you can switch to the explorer view and find the desired file. (In this case, I am looking at the `index.html` file in the `playground` folder.) Click on the file's icon and choose Properties from the context menu. This will load the Properties screen. While this displays the most commonly used properties for the file, it does not display all of them. To see all the properties, click on the Advanced button at the bottom of this screen. You will get a screen that looks like this:

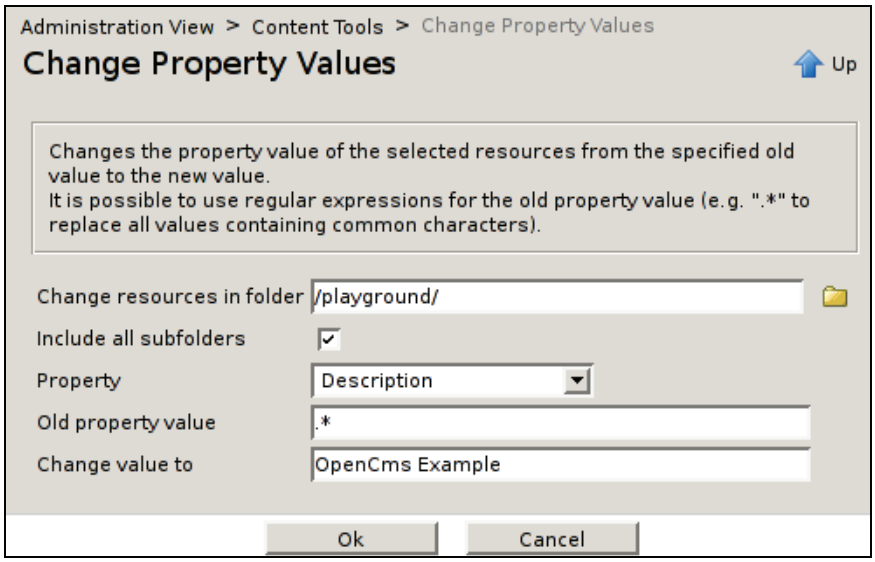


This screen displays all the properties—over fifty of them, almost all of which are currently unused. Some of the properties determine how OpenCms will treat the file; others provide additional information about the document. You can create your own custom properties with the Define button at the bottom of the screen.

We will now look at the Change Property Values screen, which allows you to change the properties of multiple files at once.

Changing Property Values

We will now see how to change the value of one of these properties for all the files in the `/playground`. This is done using the Change Property Values screen, which looks like this:



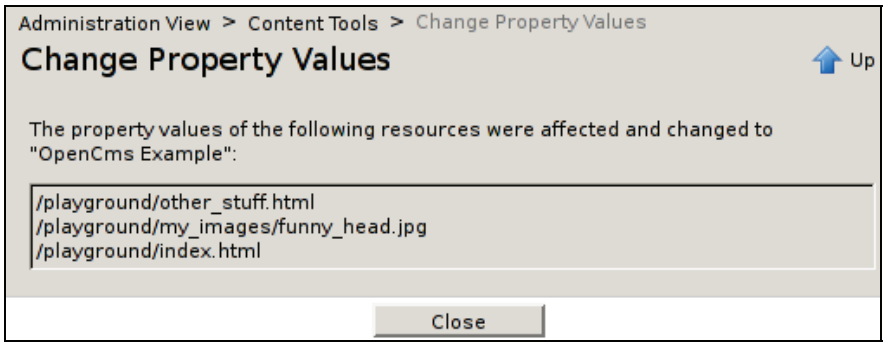
The Change resources in folder field indicates which folder we are targeting. Of course, if we wanted to change information for all of our content, we could specify the root folder. But we just want to change files in the `/playground` folder.

Checking the Include all subfolders checkbox would result in all of the files in all of its subfolders being changed too. Unchecking it restricts the change to only files in the directory itself.

The Property drop-down list specifies the property that will be changed. In this case, we are going to change the Keywords property.

The Old property value is used to restrict the change to only property entries that match this field. For example, if we only wanted to change the keywords for files whose Keywords property contained *only* the word lemons, we could set this field to lemons. As the help text at the top of the screen indicates, the Old property value field supports regular expressions, so we could also use more advanced pattern matching to determine whether the property ought to be replaced. We want to change the Keywords property for all the files, even where this property has not been set, so we will use the simple pattern .* (a period and an asterisk), which is the regular expression for "match anything".

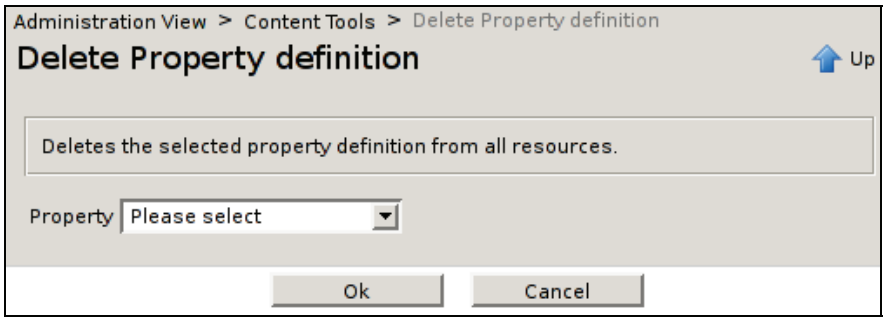
Finally, the Change value to field is for the new property value, which will replace the old values of the Keyword property. Clicking Ok will change all the applicable properties and return a summary screen.



You can see from the screenshot above that three files were modified. Clicking the Close button will return you to the Content Tools screen.

Deleting Property Definitions

The second property tool is the Delete Property definition screen:

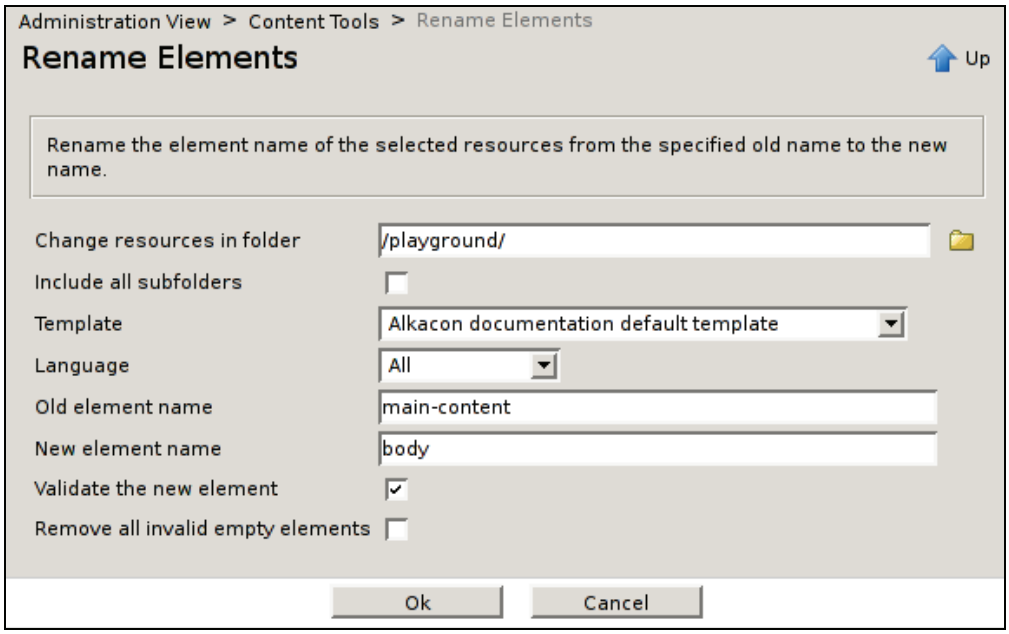


This screen can be used to remove a property. Simply choose the property from the Property drop-down list and click Ok; the property will be deleted (and from all files in the VFS). Use this tool with discretion, and do not delete properties just because you do not know what they are for.

Renaming Elements

The second section of the Content Tools screen is called Element Tools. These tools operate on the elements of Page-type documents (or any other type of document that uses elements).

As you may recall, OpenCms templates use elements to determine where content is displayed. The Rename Elements screen is used to change the name of an element in a Page-type file so that it matches the appropriate template. This can be very useful if you have created the page using one template (which uses one particular element name), and you are switching to another template (which uses a different element name). Since some templates have multiple elements, you may use this tool to manage how content appears on those pages as well.



The Change resources in folder field determines which folder the tool will scan. If the Include all subfolders box is checked, subfolders will be searched as well. Otherwise, only pages in the specified folder will be searched.

The Rename Elements screen can be used to validate that the element replacement is good for the desired template and make sure that the template has the appropriate element slot. If you know which template you are going to use, it is a good idea to use the built-in validation feature.

If you are going to validate the new element, choose the correct template from the Template drop-down list.

If you only want to change the element for a particular language, you can specify this language with the Language drop-down list.

Type in the name of the element you want to replace in the Old element name input field. The New element name input field is for the new name for the element.

Checking the Validate the new element checkbox will cause OpenCms to scan the template specified in the Templates drop-down list to make sure that it has an element that matches the value of the New element name field. This check is done before the replacement begins, and no replacements will be made if the check fails.

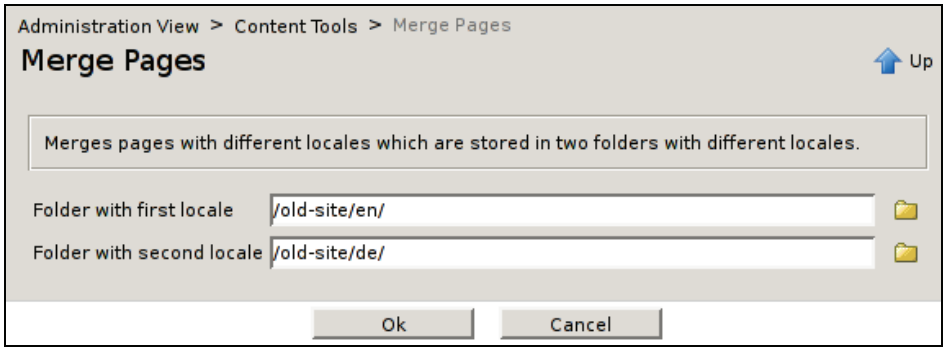
Finally, if the Remove all invalid empty elements is checked, then any invalid elements will be removed from the scanned files. It is safe to leave this box unchecked unless you need to clean some of the documents.

Clicking Ok will cause OpenCms to do the required replacement, displaying its progress as it goes. Once the replacement is done, the Ok button at the bottom of the status screen will be activated. Clicking this button will take you back to the Content Tools screen.

Merging Pages

The Merge Pages screen is designed to ease the transition from the old way of handling locales in previous versions of OpenCms to the new way of storing multiple locales in the same file in OpenCms 6.0. It is useful if you are moving from an old version of OpenCms.

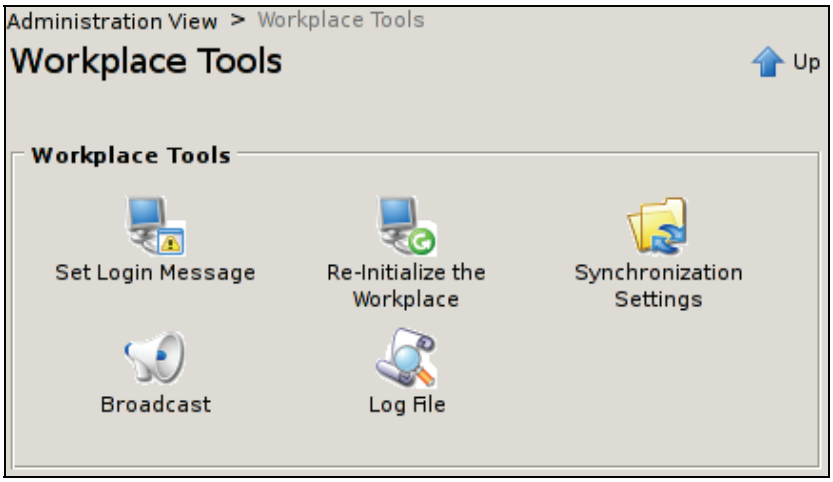
In previous versions of OpenCms, the *de facto* way of dealing with multiple locales was to create a folder for each locale and store translations of each page in each of these folders. You can use the Merge Pages screen to merge matching files in different locales and create one file with both locales.



You should specify the first locale in the field, Folder with first locale, and the second locale in the field, Folder with second locale. Clicking Ok will then merge the files. Files with matching names will be merged into one file, which will be placed in the first folder. The file in the second folder will be replaced by a sibling that points to the first. From the user's point of view, both files are exactly the same. (For example, both have English and German versions of the same content.)

Workplace Tools

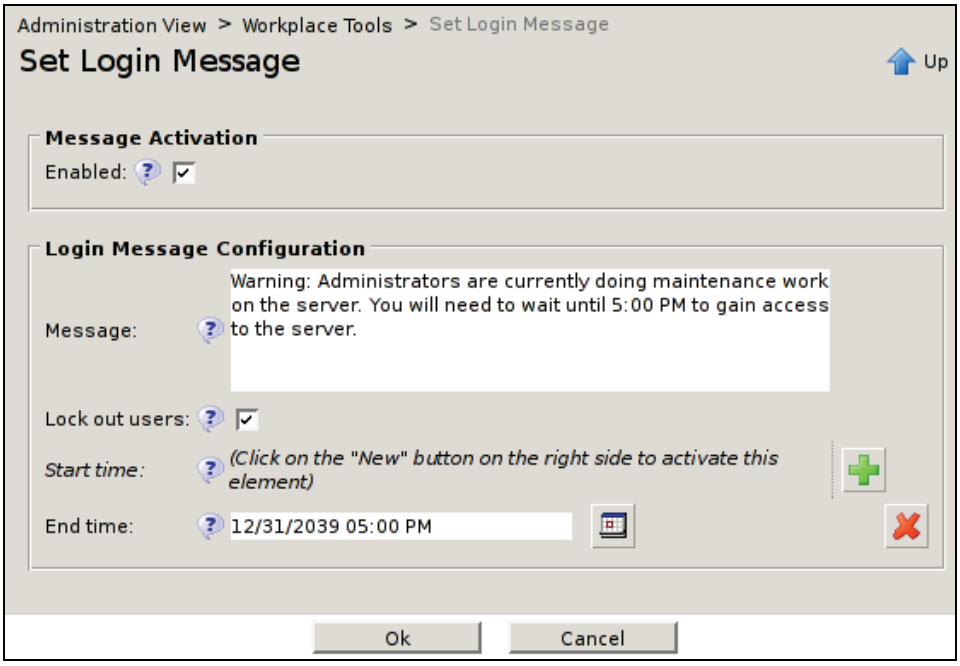
This last group of tools in the administration view deal directly with the OpenCms Workplace.



There are five buttons here, Set Login Message, Re-Initialize the Workplace, Synchronization Settings, Broadcast, and Log File.

Setting the Login Message

The Set Login Message screen lets you add a text message that will display on the user's screen after login, but before the Workplace loads. You can even use this tool to temporarily prevent non-administrators from logging into the Workplace—a useful feature when you are performing maintenance.



If the Enabled box in the Message Activation section is checked, the message entered below will be displayed to all users when they log in to the Workplace.

In the Login Message Configuration section, there are four fields for configuring the message notification.

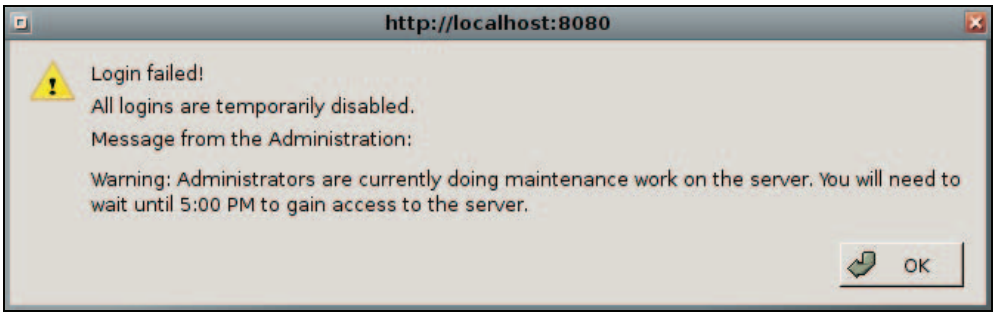
The first, the Message text box, is for the text you want displayed to your users.

If the Lock out users checkbox is checked, only members of the Administrators group will be able to log in.

The remaining two fields, Start time and End time, are optional. These automatic times are handy when a task is run automatically by the scheduled jobs tool.

Once you have clicked Ok, the message will take effect (unless you set a start time later than the present, in which case the message will not be displayed until the appropriate time).

When users who have been locked out try to log in, they will see a message like this:

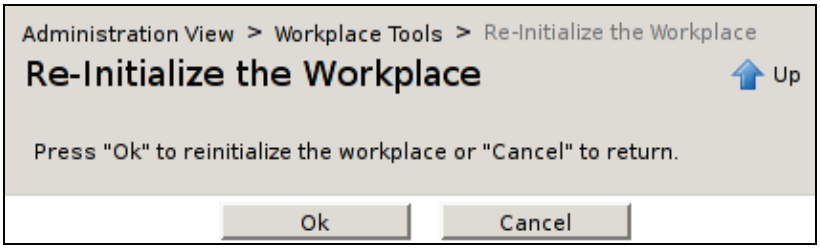


Note that even administrators will see the alert after they log in, though they will not be prevented from accessing the Workplace.

Only one login message can be configured at a time, and to edit the message, you can simply go back to the Set Login Message screen.

Re-Initializing the Workplace

Sometimes, such as after you have installed a new module that modifies the way the Workplace operates, you may need to force the Workplace to reload. This can be done with the Re-Initialize the Workplace screen.



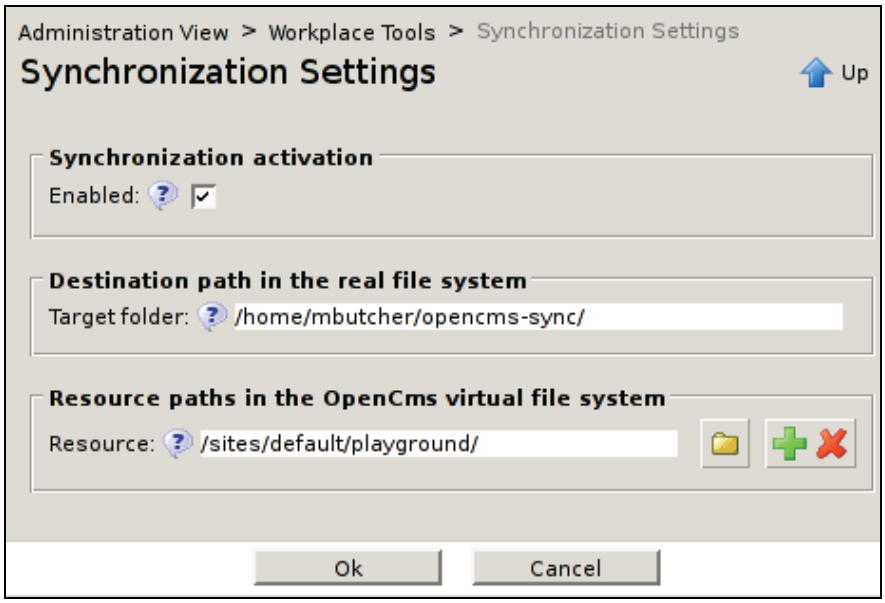
This screen just has an Ok button and a Cancel button. Clicking Ok will re-initialize the Workplace and bring you back to the Workplace Tools section.

Synchronization

Synchronization is another advanced OpenCms capability. It allows developers to work on their content outside the VFS and then synchronize the local copy with the VFS. While this may be useful for developing JSP pages or static content, it should be used with care, because it bypasses many of the safeguards in OpenCms.

> Synchronization is considered a developer's tool. It is mainly designed to help software developers edit JSP files. It can also be used as a sort of backup tool. But it is not designed to assist editors in editing content. While you can edit content in a synchronized copy, it is typically inconvenient to do so, as you will be editing raw XML.

The Synchronization Settings screen is used for setting up synchronized directories:



The Enabled box in the Synchronization activation section must be checked for synchronization to be activated. Otherwise, no synchronization will occur.

The Target folder text field in the Destination path in the real file system section should be set to the full path of the folder on the file system of the server where the synchronization files will be written and read. The servlet engine must have permissions to read and write in that folder.

The Resource field in the Resource paths in the OpenCms virtual file system section is used to specify which directory will be synchronized to the disk. If you need to synchronize more than one directory, you can add more Resource fields by clicking the green plus sign button to the right of this field.

Clicking Ok will return you to the Workplace Tools screen. Synchronization is now configured, but no synchronizing has occurred.

The next time you log in, you will see a new icon in the OpenCms toolbar—a folder with two blue arrows. It is located between the preferences button and the help button. Clicking this button will bring up a confirmation screen. Click Ok to synchronize. As usual, OpenCms will display its progress.

Now, copies of the resources identified above should be written to the server's file system. Here is what my synchronized directory looks like:

```
 mbutcher@bezer:~/opencms-sync$ ls -R
.:
sites/  #synclist.txt

./sites:
default/

./sites/default:
playground/

./sites/default/playground:
index.html  my_html_snippets/  my_images/  my_text_file
other_stuff.html

./sites/default/playground/my_html_snippets:
def_list  my_bullet_list

./sites/default/playground/my_images:
anna_teacup.jpeg                          funny_head.jpg
Bug-eyed_apple_with_cheesy_smile-small.jpg  storm-clouds2.jpg
mbutcher@bezer:~/opencms-sync$
```

As you may notice, the root folder is not `/playground`, but `/sites`. The playground folder is in `/sites/default/playground`. The structures of the synchronized folders represent the path from the root of the VFS. That way, if you add any other resources later, they will be located in the correct place in the server's file system.

> You may notice the special file named `#synclist.txt`. This contains information about the contents of the synchronized copy. Do not edit this file, or you may be unable to resynchronize the local copy back into the VFS.

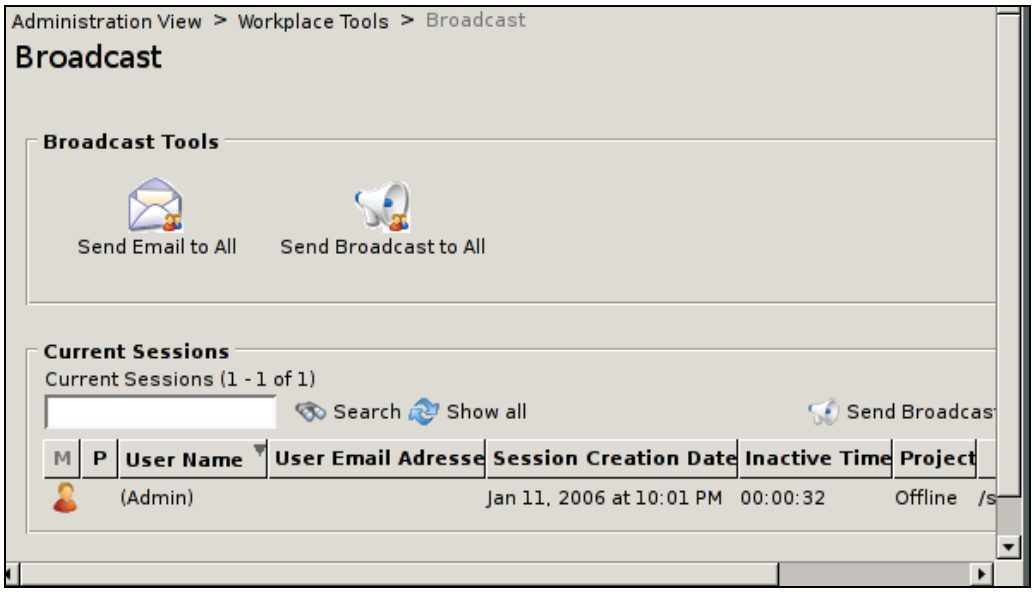
If you create a new file in the server file system, copy and then click the synchronization icon. The new file will be imported into the VFS, as will changes to the existing files, so that all of the changes made to files will be written to the server's file system. In short, OpenCms will keep the two copies synchronized.

> The files synchronized to the server's file system are exact copies of those in the VFS. Many of them will contain JSP and Java source code, XML, or even compiled Java byte code. Be very careful when you edit them.

Again, synchronization management is a development tool and is not intended to become the primary interface for working with OpenCms. When used correctly, it can expedite development time, but its abuse can lead to lost data and corruption of content.

Notification Messages

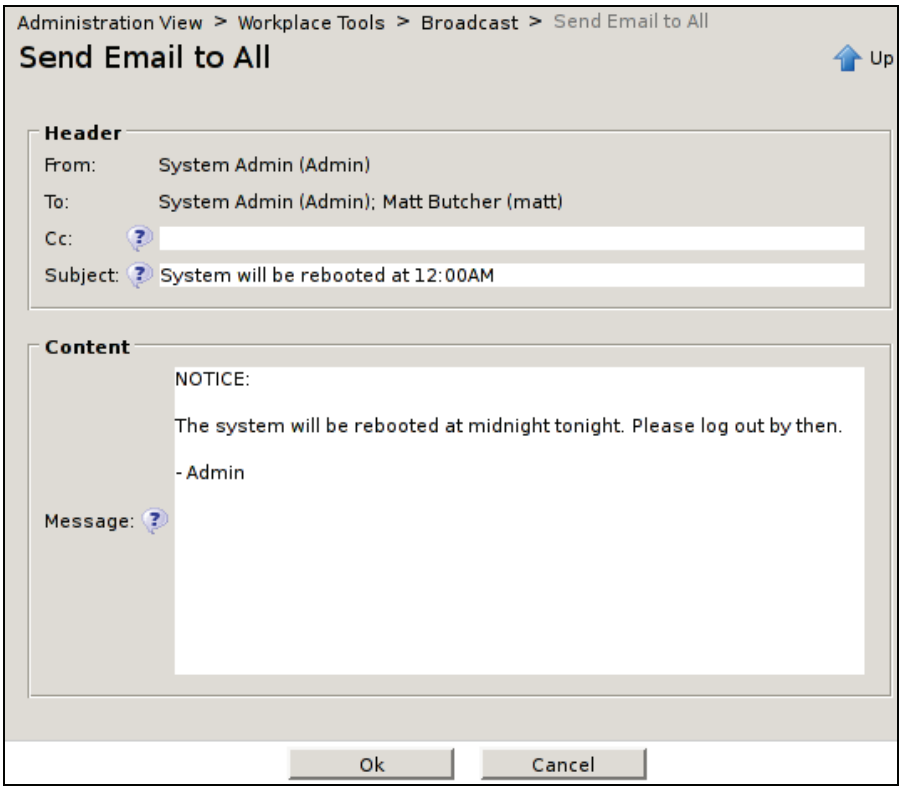
The Broadcast screen is a utility for sending notification messages to Workplace users.



In the Broadcast Tools section at the top of this screen, there are two buttons, Send Email to All and Send Broadcast to All. At the bottom of the screen, the Current Sessions table lists all the users who are currently logged in.

Sending Emails to All Users

The Send Email to All screen is used to send email messages to all users who are currently logged in:

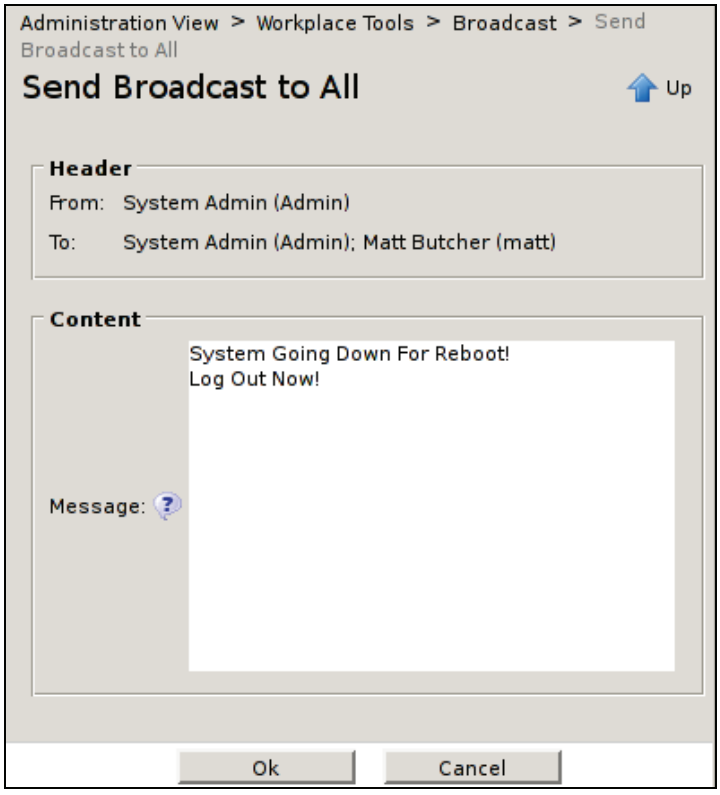


The To and From fields are generated automatically. You can use the CC field to send the message to additional email addresses. You should complete the Subject and Message fields, and click Ok to send the message.

> You will need to have the SMTP settings configured in the `$CATALINA_HOME/webapps/opencms/WEB-INF/config/opencms-system.xml` file for the email messages to be sent.

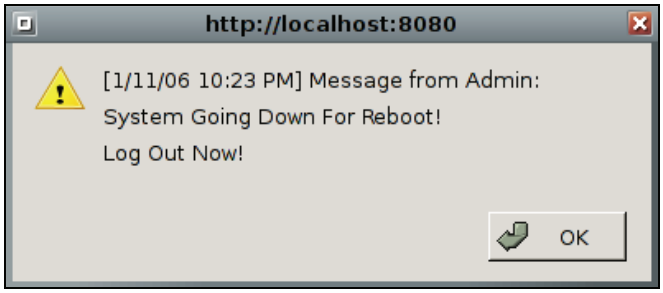
Sending Notification Messages to All Users

The Send Broadcast to All screen is for sending messages to users on the system by way of the Workplace:

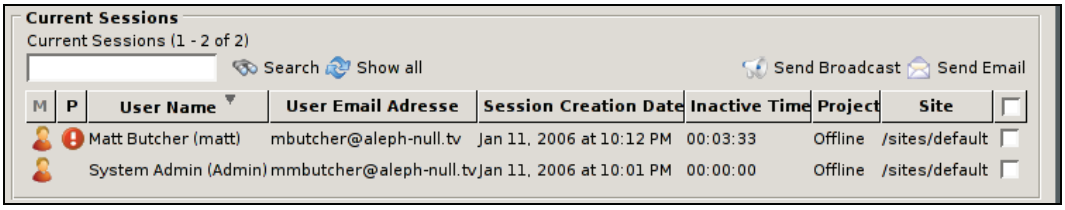


The message will automatically go to all of the users logged into the Workplace. Simply enter the message and click Ok.

Each user will be notified (as soon as they do something in the Workplace) with a pop-up message.



However, if a user is idle, the message will not be delivered. You can track the progress from the Current Sessions table in the Broadcast screen.



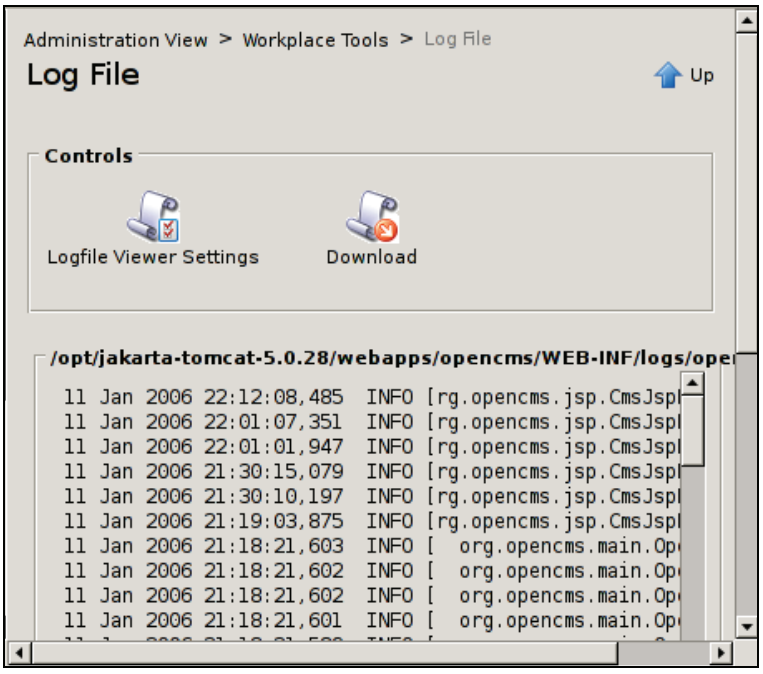
Users who have not yet received the notification will have a red exclamation point next to their user name (in the column marked P).

You can also select users from the table and send them email or broadcast messages using the Send Broadcast and Send Email buttons in the upper-right corner of the Current Sessions section.

The OpenCms Log File

The Log File screen is used for accessing the OpenCms log.

> Logs are stored in the server's file system, where you can access them with a regular text editor. They are located in the `$CATALINA_HOME/webapps/opencms/WEB-INF/logs` folder.



The log file is displayed in the lower half of the screen. The Controls section at the top of the screen has two buttons, Logfile View Settings and Download.

The Logfile View Settings screen is used to customize the way the log file is shown.

The Download screen displays some basic information about the log file (such as size, location, and time stamp) and gives you the option of downloading the file to your local file system for viewing.

Summary

In this chapter, we toured the administration view and examined the tools available to administrators.

At this point, you should have a good understanding of how to use the administration tools. Next, we will take a quick look at the workflow view, before moving on to customizing the OpenCms site.

5
Workflow

In the previous chapters, we looked at the explorer and administration views of the OpenCms Workplace. In this chapter, we will turn to the workflow view. The workflow view provides project management capabilities and task tracking for OpenCms projects. This chapter will cover:

- Understanding the concept of workflow
- The workflow view in the OpenCms Workplace
- Creating, assigning, and tracking tasks
- Managing groups of editors
- Setting and using permissions

What is Workflow?

Generally speaking, workflow is the process, from start to finish, of creating one or more pieces of content. Typically, the workflow process begins with an idea for, or requirements for, a new piece of content, and ends when the new piece of content is published. A fundamental aspect of good enterprise-level content management is the capability to handle the workflow of resources as they make their way through the editorial process.

Different content management systems handle workflow in different ways. In the most extreme cases, the workflow is so tightly integrated into the CMS that every single piece of content has associated workflow information. In others, workflow may not take such a prominent position. In OpenCms, workflow is distinct from content, meaning that there is no necessary relationship between content and workflow objects.

This separation of workflow objects from CMS content has a few distinct advantages:

- Implementing workflow is optional—if you don't want to use it, you don't have to use it.
- Since workflow is not tightly integrated with the structure of the repository, you have some flexibility in how you implement workflow processes.
- If you don't like the default OpenCms workflow, you can install third-party workflow modules or even build your own custom workflow module.

Most of this chapter covers the technical aspects of workflow—how to create and manage tasks in OpenCms, how to use projects in conjunction with workflow, and how to use various features of the workflow view. But I have also tried to provide a higher-level view of workflow to give you an idea of how it can simplify the editorial process of content creation.

Before diving into the technical details, I want to provide a brief overview of how workflow works. At the end of the chapter, I will return to some of the concepts presented here and discuss workflow strategies.

How Workflow Works

The basic assumption behind workflow is that various individuals and groups within an organization perform differing roles. If there is only one person who manages content from the idea through creation to publication, then tracking workflow is a bit excessive. Even if two or three editors work on the same system and each is wholly responsible for her or his own content, then tracking workflow adds very little value. Instead, workflow is intended for situations where different people or groups of people are responsible for different aspects of production.

Consider a simple scenario. A small organization has a team dedicated to managing its OpenCms-driven website. The manager of the team is responsible for overseeing the process and assigning new articles. A handful of writers create new content, each of them typically working alone when writing his or her piece. Once an article has been written, the editor proofreads the text, ensures that it is in accordance with organizational guidelines and, with the approval of the manager, is responsible for publishing the article on the public website.

In this simple example, there are three clearly-defined roles—manager, writer, and editor—and each of these roles has a set of tasks for which members of the role are responsible. Project managers assign and track tasks. Writers create content. Editors check and publish the content.

But even in such a seemingly simple environment, the process can get convoluted. How can a manager keep track of what each writer is working on? How does the editor keep track of what is ready to publish and what needs a little more work? If one writer is swamped with work and another has nothing to do, how can the two effectively balance their workloads?

There is a need for a set of tools to streamline the process, making it easier for all parties to keep track of their respective tasks and to interact with each other.

The first step is to create a structure in OpenCms that mirrors the organizational structure of those involved in the process of managing content. Given the example above, we need to create roles for managers, writers, and editors.

> The OpenCms workflow tools use the term *role* instead of the word *group*. In fact, workflow roles are identical with groups. (Chapter 4 covered the creation and management of groups.)

The second step is to establish a procedure. While OpenCms provides the tools for implementing procedures, these tools are intentionally flexible. It is up to the organization to formalize procedures and then to use the tools in accordance with these procedures.

The procedure that our fictional organization follows has been roughly outlined above—the manager creates a task and assigns it to a writer. The writer then creates the content, notifying the editor when the content is ready. The editor makes changes (possibly giving the content back to the writer for further revisions). When the editor thinks the piece is ready, he or she notifies the manager, who has the final say in whether or not the content is ready for publishing. If so, the editor publishes it. Of course, there may be a number of ancillary procedures, but this covers the basics.

The third step is to set access controls to allow the appropriate groups or individuals access to the appropriate areas of the VFS. Of course, this is optional, and in the simple example above it is unlikely that any such measure would be needed, but in larger organizations with teams of editors, it may be more appropriate to restrict specific groups to specific areas of the site.

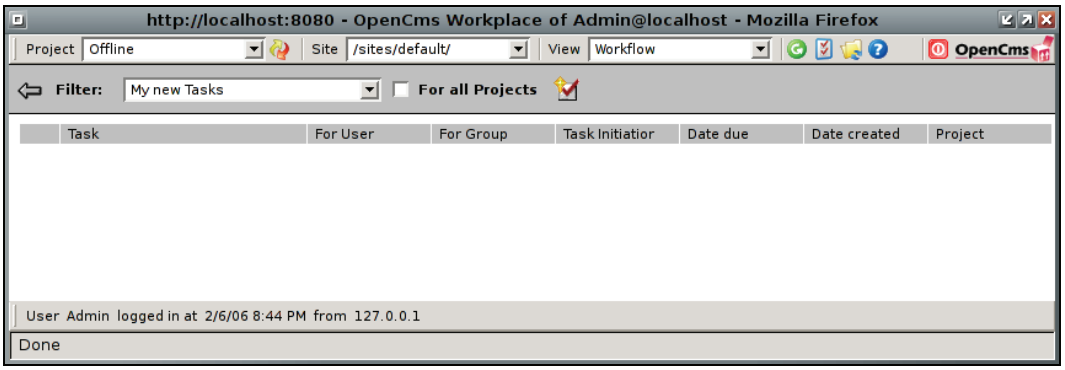
I have provided a sketch of what workflow does, what problems it solves, and how it is to be implemented. At this point, we will turn to the more technical aspects and look at the workflow view of the OpenCms Workplace. The last part of the chapter, however, is dedicated to providing higher-level strategies for using workflow in your organization.

The Workflow View

The workflow view provides a task-based interface to the workflow system. In other words, the whole process of workflow is broken down into tasks, where each task can be assigned to a user or group, and progress on the task can be monitored. We will take an in-depth look at tasks and the tools used to work with them.

In Chapters 3 and 4, we looked at creating and managing content, and in Chapter 4, we examined tools for managing users and groups. These concepts will be employed here. Through the workflow view, we will be able to assign editors tasks to complete within the project.

To enter the workflow view, select Workflow from the View drop-down box in the top toolbar in the OpenCms Workplace.



When the workflow view is loaded, you will see the task list, which you may find unimpressive, since it is empty. The interface (color scheme, icons, and layout) is different from that of the administration and explorer views (though it is similar to the legacy administration interface). If you were an OpenCms 5 user, it may look more familiar—workflow still uses the older layout.

In the upper-left corner of the workflow toolbar, there is a left-pointing arrow. This is the back button. When it is blue, clicking it will return you to the previous screen. Next to the back button is the Filter drop-down list. Filters determine what tasks are shown in the task list. By default, the My new tasks filter is applied. This will show all tasks that are explicitly assigned to the present user ID (Admin in the screenshot above) and marked as being in the state *new*.

The filters in the drop-down list are organized into four categories. The first three categories organize tasks by giving them one of three **states**—new, active, and completed.

States indicate what stage a particular task is in. When a task is initially created, it is put in the *new* state and assigned to a particular role and user. Once a user accepts the task, the task will no longer be marked as new. A task is active from when it is created until it is explicitly closed, in which case it is marked completed. (Note that a task can be both active and new at the same time—these states are not mutually exclusive.)

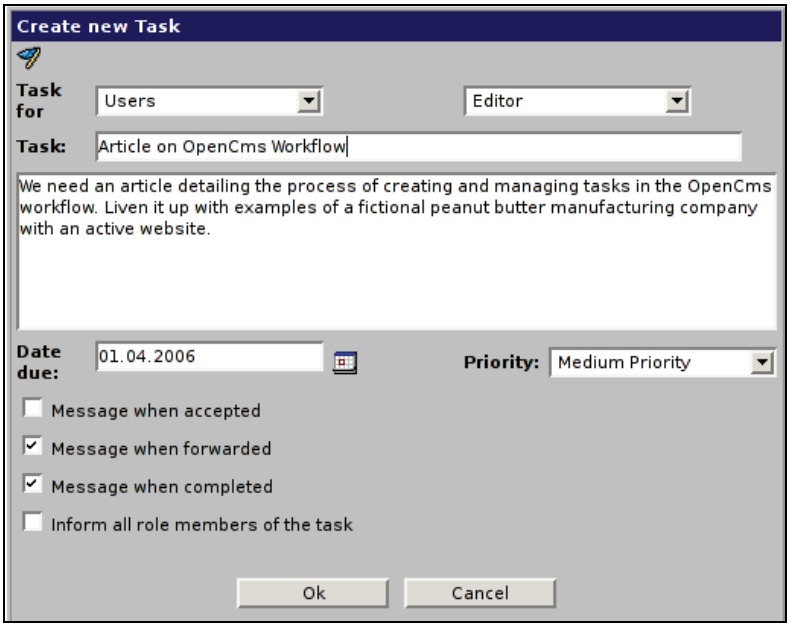
The Filter drop-down list has the following six options: My active Tasks, Active Tasks for My Roles, All active Tasks, New Tasks created by me, Active Tasks created by me, and Completed Tasks created by me.

Unfortunately, there is no filter to show you all of the tasks in the system.

Next to the Filter drop-down list, there is a checkbox called For all projects. Previously, I mentioned that tasks are project-sensitive. By default, only tasks that are in the currently selected project (in this case, the Playground project) are displayed. If this box is checked, then tasks from other projects are displayed as well.

Creating a New Task

The last item on the workflow toolbar is a button that looks like a sheet of paper with a yellow cross and a red check mark. This is the Create New Task button. Clicking it will bring up a screen for creating a new task.



The form is deceptively simple. The first thing to do is set the role and group information.

> In Chapter 4, we examined the process of creating new groups. As you may recall, when creating a new group, you have the option of checking the Group As Role box. If a group is set to act as a role, then it will be available here in the workflow view.

The first field in the form is called Task for. It is followed by two drop-down lists. Its options are the Users role and all of the roles that have the Users role as a parent. Practically speaking, that means that the Admin group is not displayed in the system, and neither are groups whose users cannot log into the Workplace.

The second drop-down list is populated with a list of all of the users that are direct members of the role selected in the first list. (The Editor user is just one of the users in the Users role in my OpenCms installation.) If no role is selected in the first list, this list is empty.

> In the Chapter 4, we saw how users could belong indirectly to roles (groups). Users who are only *indirectly* members of roles will not be displayed in the second drop-down list.

While you must select a role, you may choose not to assign the task to a specific user.

The Task text field is for the title of the task. This title will now show up on the task list mentioned above.

The text area below the Task field is for a description of the task. This is the main place to explain the purpose of the task. Unfortunately, the field is rather small, and it can be tedious to enter sufficient information. If your description is going to be long, you may want to consider restricting this field to only the basics and using a comment (explained below) to provide a more detailed explanation.

The Due date field is used to specify when the task ought to be completed. Clicking the calendar icon to the right will bring up a more friendly point-and-click date selection widget. The Due date field is required.

The Priority drop-down list provides a way of indicating how important (or unimportant) this particular task is. There are three priorities—high, medium, and low.

Beneath the Due date field, there are four checkboxes. The first three have to do with sending notifications under certain circumstances, and the last is for assigning the task to multiple users. If the Message when accepted checkbox is checked, you (the task's creator) will be sent a message when the task is marked as accepted by another user.

If the second checkbox, Message when forwarded, is checked, the task creator will be notified if the task is forwarded from one user or role to another. If the task is the writing of an article (as in the example at the beginning of this chapter), then when the writer finishes, he or she will forward the article to the editor, and if this box is checked, the manager (who created the task) will be notified that the task has been forwarded.

If the Message when completed checkbox is checked, then when the task is marked completed, the user who created it will be notified.

Finally, if the Inform all role members of the task checkbox is checked, then this task will show up in the task list of every user in the role selected in the first Task for drop-down list.

Once these fields have been completed, clicking the Ok button will create the new task. You will be returned to the task list (which will still be empty unless you have assigned the task to yourself).

Notification

Once the task has been created, an email will be sent to the assignee with some generic information about the new task. The following is an example message sent to the user Bob after I assigned him a new task:

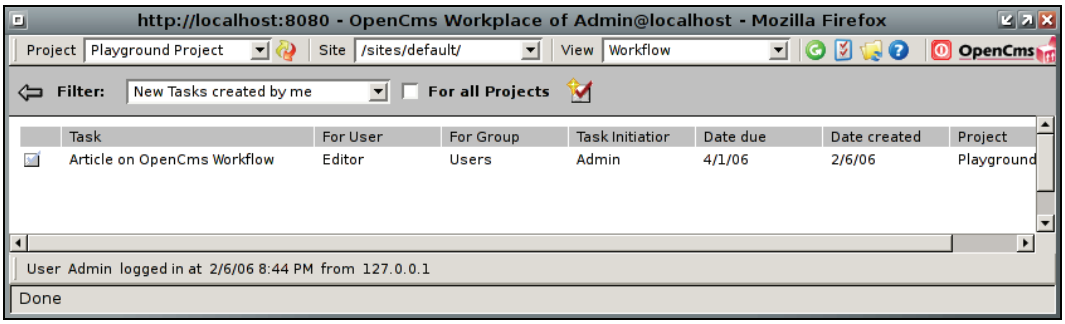
```
From: opencms@example.com
To: bob@example.com
Subject: OpenCms task management: new task for Bob Editor (Bob) /
Editors

Automatic message from OpenCms task management:
A new task was created for you or your role.
Project: Playground Project
Task: Write an article
Task Initiator:   (Admin)

http://example.com:8080/opencms/opencms/system/login/index.html?start
TaskId=34&startProjectId=13
```

The link at the bottom of the message takes the user directly to the new task (though the user must first log in if they have not already done so).

Likewise, the next time Bob views his task list, the new task while be displayed.

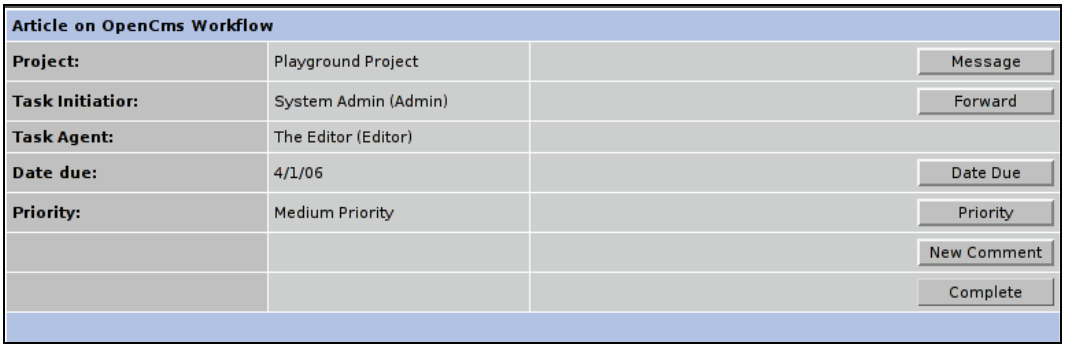
Viewing the Task

You should now be at the task list viewer, which will likely still be empty. To see the task you just created, select the New Tasks created by me item from the Filter drop-down list.



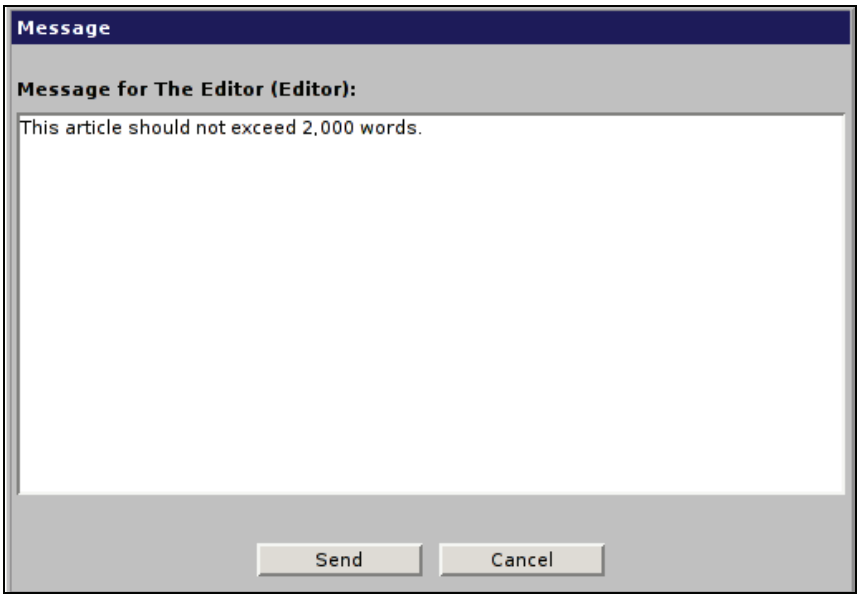
The icon in the column on the far left (in this case a blue check mark on a white rectangle) indicates the status of the task. A light blue check mark indicates that the task is new, a dark blue check mark indicates that the task is active and accepted, a red check indicates that the due date has now passed, and a black X indicates that the task is completed. In addition to this icon, items that are marked as high priority will also have an exclamation point, !, icon. Items that are marked as low priority have a blue down arrow icon. The next seven columns in the task list display basic information about the task.

To view the details of a task, click on the task name in the Task column. At the top of the task details page is a table with information about the task and a number of buttons for manipulating or modifying the task. Underneath this table, there is a (potentially long) list of documentation that is attached to the task. This list will include all the modifications and comments made on this task. First, let's look at the table at the top of the screen:

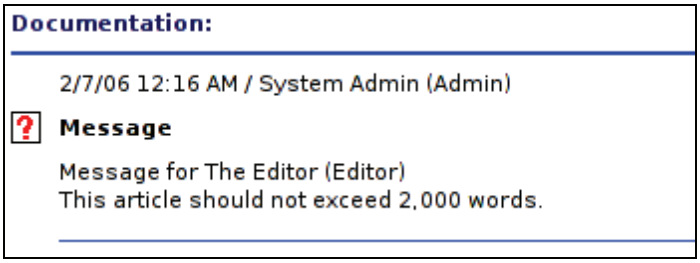


The buttons on this screen vary depending on the relation of the task to the person who views the task. For example, the task creator (the initiator) has a button called Message. But on the screen of the person to whom the task has been assigned (the agent), this button is called Query. In either case, this button is used to communicate between the various individuals working on the task.

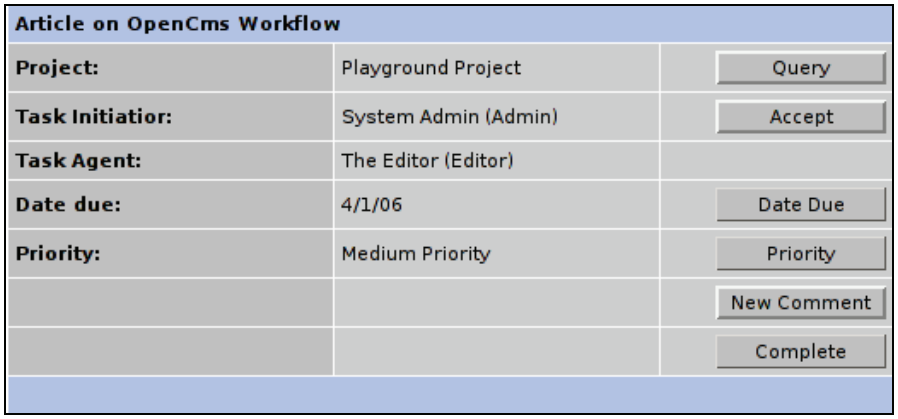
Clicking the Message (or Query) button will open a text editor screen that you can use to create a message.



Clicking Send will send the message. If you are the initiator, the message will go to the agent. If you are the agent, the message will go to the initiator. While the message recipient will be notified of the new message by email, they will have to log into the system to actually read it. The message appears in the Documentation section and look like this:



When assigned a new task, the agent must accept it before working on it. On the agent's screen, there will appear a button labeled Accept. For example, here is the task details screen for the user named Editor:



Clicking Accept will prompt the user to confirm acceptance. If the user (in this case, the user named Editor) confirms, then they will be returned to the task details screen. The background color of the table at the top will be different, indicating that the task has now been accepted, and there is a Forward button instead of the Accept button. Clicking this button will take you to a screen that allows you to assign the task to another role or user.

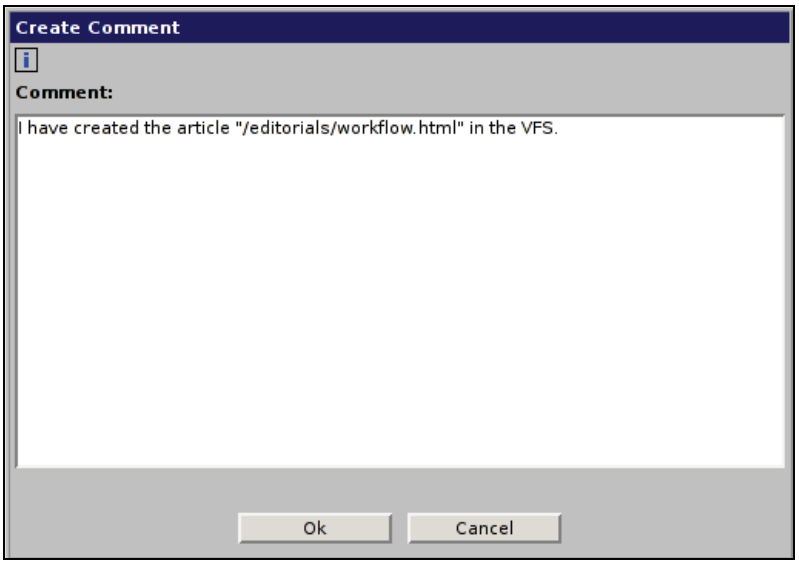
Returning to the example at the beginning of the scenario, when the writer finishes writing a draft of the article, he or she is to notify the editor. One way of accomplishing this in OpenCms is to forward the task to the editor. That way, the task ends up in the editor's queue and is easy for the editor to track. Should the editor find a problem with the article, he or she may send it back to the writer. Otherwise, the editor may do whatever is necessary (e.g. publish the article) and then mark the task as complete.

> Other users who are marked as neither the agent nor the initiator will see a Take button instead of an Accept or Forward button. By clicking that button, a user may become the agent on that ticket—in other words, they will have taken the task away from the previous agent.

A task initiator can also change the Date Due and Priority of a task. While the buttons for making the change appear on both the agent's and initiator's screens, they are deactivated on the agent's screen. The initiator can use these buttons to modify the date the task is due or the priority of the task, respectively.

Forwarding or taking a document, as well as changing due date and priority, are all events that will be noted in the Documentation section of the task view. This serves as a virtual "paper trail" for tracking the history of a task.

Clicking on the New Comment button opens a window very similar to the query/message screen.



An editor may log the progress of a task using the comments section. These comments are also logged in the Documentation section of the task view.

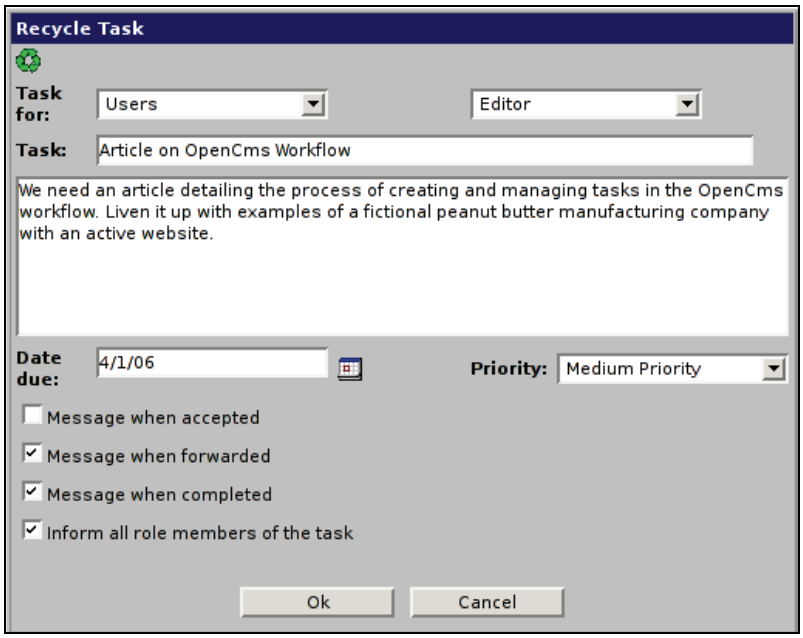
Until a task is accepted, the Complete button is deactivated. When the task is marked as accepted, OpenCms assumes that the user is working through the task, and it does no checking to see if the user is actually completing the task.

Once the task is completed, clicking the Complete button will bring up a confirmation dialog. If you click Ok to Really mark the task as complete?, the task will be marked complete and be removed from the active tasks list.

Once the task is completed, all the buttons are deactivated for all users except the task initiator. The task initiator will have one active button, appearing in the slot where the Complete button used to be. This new button is called Recycle. Recycling a task reopens it for completion again. This is useful, for instance, when you have a task that has to be completed every week or month. Rather than create a new task each time, you may simply recycle the old task.

Recycling

When you click Recycle, the Recycle Task screen will appear.



This screen has all of the same fields as the Create New Task screen had—except that these fields are all pre-populated with the previously set values.

Clicking Ok will mark the task as *new* and place it in the new tasks queue of the selected user and role. This is not really a new task, but the same task as before. Consequently, all of the old Documentation history will be preserved, and a new entry will be added to the documentation noting that the task has been recycled.

That is all there is to the OpenCms workflow view. With these simple tools, you should be able to set up some reasonably complex workflows.

Again, I want to draw attention to the fact that the workflow tools do not directly interact with the contents of the VFS. But by employing a few strategies, you can adapt workflow to your own needs. We will now look at some strategies for managing workflow in OpenCms.

Workflow Management Strategies

The whole workflow system can seem daunting to someone new to the CMS concept. However, it can be a powerful tool for managing your content. Here are a few strategies for managing tasks using projects and workflow tracking.

Use Projects to Manage Content Areas

Projects provide a way to cleanly divide the content into logical groups. It helps with access control by limiting an editor's permissions to only the resources in his or her projects. It helps with project management, as project managers have a well defined domain for which they are responsible. Over all, it is an effective way of organizing the content.

There are two strategies for defining projects. The first is to create a project for every major content area within the site. This setup takes the functional route in the division of content. A project manager may then be assigned multiple projects. The advantage to this system is that editors may be restricted to a small subset of the site. Also, tasks may be moved from one project manager to another with a minimal amount of work.

The second method is to create one project for each project manager, reflecting organizational structure (as opposed to site structure). This structure makes the job of the project manager a little easier, as he or she must only keep track of one set of content and tasks. For editors that contribute to many different areas of content all under *one* project manager, this approach is easier. Like the project manager, they only have to track one project. There are two drawbacks to this configuration. If a content area is moved from one project manager to another, reconfiguring the projects is a little more tedious. Also, it is more difficult to restrict an editor's access to only one content area, as the project includes multiple content areas. This may be resolved with careful use of groups and file permissions, but that approach requires more maintenance.

Regardless of which route you choose, using projects will help manage content as your content repository grows.

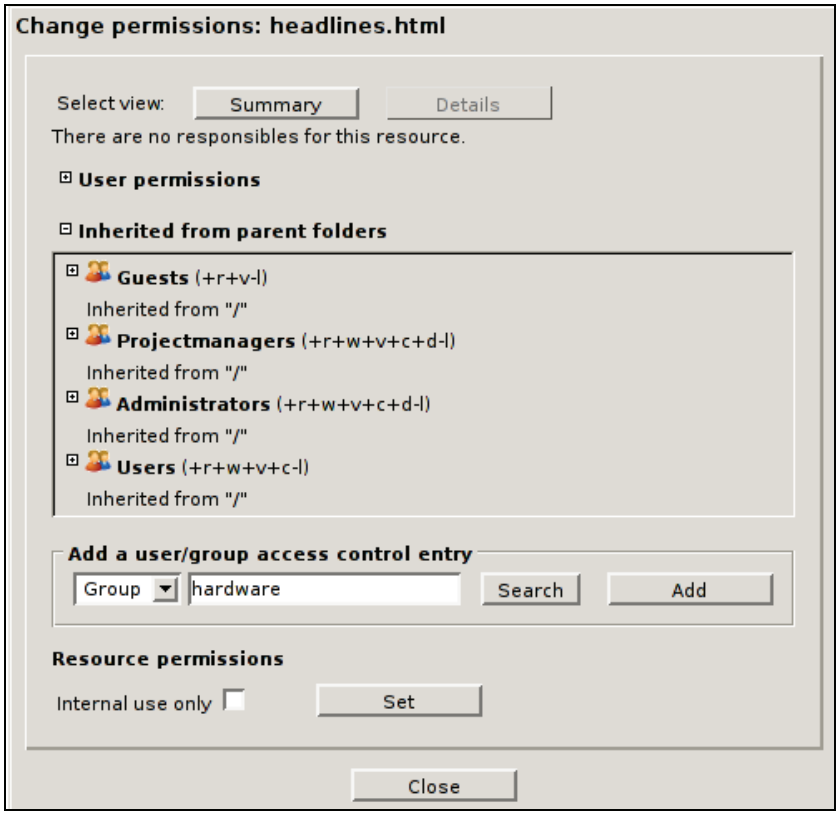
Use Group Hierarchies for Inherited Permissions

OpenCms access controls allow you to specify fine-grained permissions for any resource in the VFS. If multiple editors in different groups need to access the same resources, you will need to make a little more effort to make sure that all the right people can access shared resources.

Consider a technical news site that has two groups of editors—one that handles hardware news and one that handles software news. Most of the time, these editors are dealing with different content, but both groups have to update the `headlines.html` file. This file can only

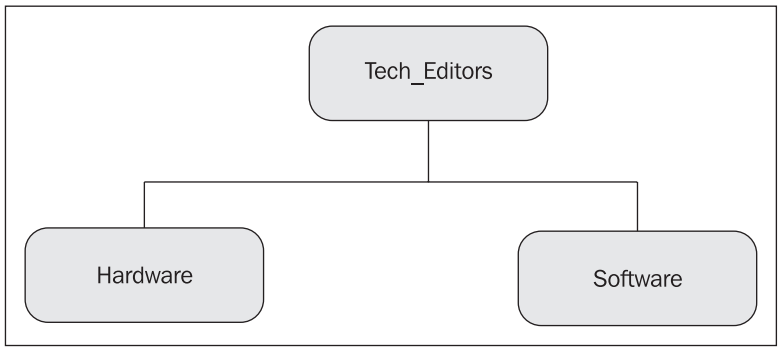
have one owner and one group, but the hardware editors are in a different group from the software editors. Since other users on the system should not be able to edit this file, simply giving write access to the file to everyone in the Users group would not be judicious.

There are two ways to solve this problem. One is to create access controls for both groups. This can be done from the explorer view by bringing up the file's context menu (clicking the icon) and choosing Permissions. You can then add new access controls by user or by group.



Alternatively, you could make the most of group inheritance and set up two groups that both inherit from a common group. Then, you can give the common group the necessary permissions. Let me explain this in a little more detail.

As we saw in Chapter 4, groups can be organized into a hierarchy so that a child group inherits the permissions of its parent. If two groups have the same parent, then giving the parent group access to the file will enable members of both child groups to access the file. For example, consider this group structure:



In this configuration, the parent group is Tech_Editors, and the Hardware and Software groups are its children. Consequently, if the Tech_Editors group has write permission for a file, then that file will be writable by members of either the Hardware or the Software group.

> To strictly enforce access and workflow rules, you will need to tighten up the access controls on the directories in your VFS. The default access controls allow all Workplace users to edit all the files in the site.

Careful planning of group layouts can make the maintenance of access controls easier in OpenCms.

Tracking Workflow with Tasks

The workflow tools are designed to provide a functional approach to managing your content. Workflow provides a clear and intuitive way to track the state of the content that is being generated, edited, and maintained. Using the workflow view can help you maximize the efficiency of the editorial process. In more complex workflows, using one task to track the entire process from creation to completion may result in tasks that are bloated and difficult to work with.

You may find it easier in such cases to assign tasks for every step of the editorial and publishing process. For example, consider a project (we'll call it 'Project X') with only one article and a corresponding JSP. Here is a list of the tasks that the project manager might create:

- Editor 1: Write the article in Project X.
- Developer 1: Create a JSP for Project X.
- Editor 2: Proofread and edit the article.

- Editor 2: Proofread the content on JSP.
- Project Manager: Approve the article.
- Project Manager: Approve the JSP.
- Project Manager: Publish the project.

We could trim down the number of tasks. For example, we could require editor 1 to forward the article to editor 2 once it is complete and then have editor 2 forward it to the project manager when it is ready for approval. This reduces the number of tasks to track, but makes it a little harder to assess progress with just a glance at the tasks list. For organizations that deal with large amounts of content, it may be more useful to use tasks that exist for a shorter period of time and which are specific to only one step in the workflow.

One way to encourage users to use (and think in terms of) tasks is to set their default view to workflow instead of explorer.

Keeping a Trail

Using assign-accept-forward-complete tracking of tasks can be sufficient. However, where there are a large number of editors, or if editors and project managers are not in the same physical location, then communication can break down and information can slip through the cracks. Emails can go unnoticed or accidentally be deleted. Conversations, whether verbal or in chat/instant-messaging are apt to be forgotten. And these methods do not keep all of the information in a central location.

To avoid these problems, you may find it more reliable to track the status of a task using the Query, Message, and Comment tools provided with the workflow tools.

If correspondence between the project manager and editor is handled through messages, and the parties involved are diligent in making comments to record progress, then each task will have a self-contained paper trail. This can promote effective communication and prevent important information or requirements from slipping through the net.

We have toured each of the three views at length, and we are almost finished with our exploration of the Workplace. All that remains is a discussion of installing and configuring OpenCms modules.

Summary

In this chapter, we toured the workflow view. We began by covering the basics of workflow; then we looked at the tools that OpenCms provides. Finally, we looked at some strategies for using workflow in an organization.

At this point, you should have a good understanding of how all three OpenCms views work. Next, we will look at customizing our OpenCms site.

6
Customizing the Site

We have already talked at length about using the system—creating and editing content, managing projects, and carrying out administrative tasks. Now, it is time to turn our attention to customizing the site. In this chapter, I will explain how to develop the look and feel of published content—adding dynamic elements and controlling resources.

We will cover:

- An brief introduction to templates
- JSP tags
- JSP scriptlets
- How to create a template
- JSP templates
- The OpenCms tag library
- Some problems that can arise when using templates, and JSP tags and scriptlets

Templates

To understand what we are doing when we create templates and add JSP tags, it will help to understand what is happening when a client requests some piece of content.

When a client requests a page, OpenCms retrieves the contents for the page. This piece of content is basically the file as it is created through the workplace. At this point, it does not have any layout or navigation. It does, however, have information about how the navigation should look, which template to use, and other such things. OpenCms uses this information to fetch the correct template and place the contents into the template. The template provides the surrounding structure, navigation, and formatting for the document. Once the content has been rendered into the template, the resulting document is shipped back to the client web browser.

> In older versions of OpenCms (5.0 and prior), templates could be authored in the XMLTemplate language. XMLTemplate is now deprecated, though you can install the legacy module if you want XMLTemplate support. From OpenCms 5.0 and on, JSP is the preferred template language.

OpenCms comes with several built-in templates. These templates, collectively called TemplateOne, provide pre-written advanced templates that you can use simply by selecting the TemplateOne template when you create a new Page file. Like regular template files TemplateOne is written in JSP and HTML, though it does use some custom Java classes to achieve some of its fancier features.

In this chapter, we will focus on creating custom templates from scratch.

Templates are composed of two types of elements. First, they use standard HTML. Parts written in HTML will be delivered to the client without any modification. Second, they can contain JSP content. JSP (Java Server Pages) is a technology, built in the Java programming language, for inserting dynamic content in HTML or XML files. JSP content can be inserted in one of two forms—tags or scriptlets.

JSP Tags

JSP **tags** look similar to HTML or XML tags. But when a client requests a JSP document, the server reads through the document in search of JSP tags. When it finds a tag, it executes the required code and inserts the results generated into the copy of the document that it will send to the client.

Tags are organized into **tag libraries** (taglibs). OpenCms provides a tag library that contains useful tags for working with OpenCms. These tags can be used to access or modify contents in OpenCms.

The following example will clarify how JSP tags and JSP tag libraries work in practice:

```
<%@ taglib prefix="cms" uri="http://www.opencms.org/taglib/cms" %>
<html>
<head>
 <title>Show User's First Name</title>
<head>
<body>
 <p>
  <b>User's First Name:
   <cms:user property="firstname"/>
  </b>
 </p>
</body>
</html>
```

Most of this example is written in HTML, and you should recognize the familiar tags. But a couple of the tags, both highlighted, are JSP tags.

The `taglib` tag at the beginning tells the server that this page will use the tags from the OpenCms library. JSP libraries are named by URI (Universal Resource Identifier). URLs are the most common sort of URI, and the URL for the OpenCms JSP tag library is `http://www.opencms.org/taglib/cms`. The first line of the code above tells the server that any tag whose name begins with `cms:` is from the OpenCms JSP tag library.

The second tag, which is the OpenCms tag library's `user` tag, tells the server to fetch user information from OpenCms and then write it into the current location in the document. More specifically, setting the `property` attribute to `"firstname"` tells the server to find the user's first name.

As the server executes the JSP tags, it creates a new document—one that contains only HTML. This new document will contain the user's first name where the `user` tag was in the original document, so when this document is sent to the client, the HTML will look like this:

```
<html>
<head>
 <title>Show User's First Name</title>
<head>
<body>
 <p>
  <b>User's First Name:
   Matt
  </b>
 </p>
</body>
</html>
```

The first line with the `taglib` tag has been removed, and the `cms:user` tag has been replaced with the user's first name (in this case "Matt").

JSP Scriptlets

JSP tags provide one way of providing dynamic content in a JSP page, but the other way is to use JSP **scriptlets**. Scriptlets are pieces of Java code that can be inserted within a document. The idea of inserting code in a document is similar to JavaScript, except that in this case, it is the server, not the client, that executes the code. The code is fully fledged Java in all its object-oriented glory. If you are going to write scriptlets, you will need to know Java.

If you do not know Java, you may want to skip to this section. We will be using JSP tags in this book, and you can comfortably write OpenCms templates without writing any scriptlets.

To give an idea of how scriptlets work, here is what the previous example would look like written as a scriptlet:

```
<html>
<head>
 <title>Show User's First Name</title>
<head>
<body>
 <p>
  <b>User's First Name:
   <%
org.opencms.jsp.CmsJspActionElement cms = new
org.opencms.jsp.CmsJspActionElement(pageContext, request, response);

out.println(cms.user("firstname"));
%>

  </b>
 </p>
</body>
</html>
```

The scriptlet is contained between the <% and %> delimiters. There are only two Java statements. The first, which spans two lines, creates a new Java object of the type `org.opencms.jsp.CmsJspActionElement` named `cms`. This object has access to all of the functions of OpenCms JSP tags. The three parameters, `pageContext`, `request`, and `response`, are all automatically provided to every JSP page.

The second statement gets the user's first name using the `user` method of the `CmsJspActionElement` class and inserts this information into the document that will be sent to the client.

Much of the OpenCms API (Application Programming Interface) can be accessed by scriptlets. Because of this, you can create powerful scriptlets that perform complex manipulations of OpenCms data. The OpenCms documentation is a good place to get started on writing more advanced scriptlets.

Working with Templates

As we have seen, JSP tags, tag libraries and scriptlets provide a method of mixing dynamic content into HTML (or XML) documents. Their main use in OpenCms is for OpenCms templates. Using JSP tags, we will be able to create templates that generate navigation information, fetch content, and do other things to make our OpenCms website attractive and compelling for visitors.

We will start by creating a place where we can store our new templates.

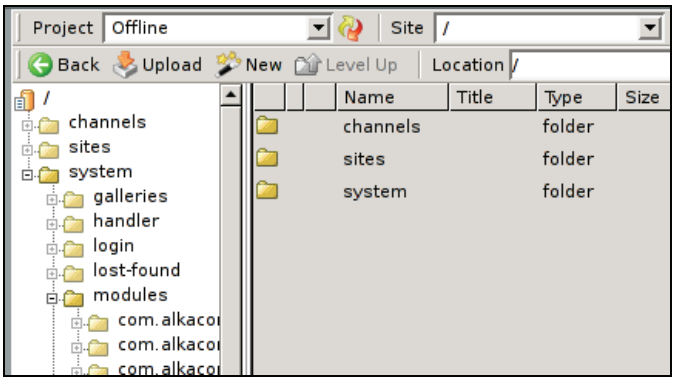
Template Module

In Chapter 4, we looked at module management. Modules are a mechanism for extending the functionality of OpenCms. Sophisticated modules can provide integration with other products, modify OpenCms itself, or extend the capabilities of OpenCms. But they can also be used for more humble tasks.

As we begin working with templates, we will need a place to store the template files. It is desirable to keep templates out of the content portion of OpenCms. Content editors don't need to see templates, nor do we want functional components cluttering up the space we use for managing content. Also, it is often useful to be able to move the templates (along with images, stylesheets and supporting files) from one OpenCms server to another without having to move all of the content. For these reasons, it can be very useful to store templates in their own module.

In Chapter 4, we created a new module, and we will now use this module as a place to organize and store our new templates.

To get to this module in the explorer view of the OpenCms Workplace, you will first need to change sites. From the Site drop-down list, choose / instead of /sites/default/. You should now be able to navigate to your module in the `/system/modules` folder in the navigation pane on the left.



The module we created in Chapter 4 was named tv.alephnull.modules.templates, and there is a folder for this module called `/system/modules/tv.alephnull.modules.templates`. Within this folder, there should be at least three subfolders—`templates`, `elements`, and `resources`. Each of these three folders will store a particular kind of data:

- `templates`: This folder will contain the JSP template files.
- `elements`: This folder will contain JSP files (or text files) that the template may use.
- `resources`: This folder will contain stylesheets, images, and other files that the client will need.

If these folders do not already exist (which may happen if, for example, the appropriate checkboxes where not checked during module creation), then you can just create them.

Modules may have other folders at this level, including the `classes` and `libs` folders for housing compiled Java classes, and the `default_bodies` folder for storing information on the body layout for a given piece of content. But we do not need these folders here.

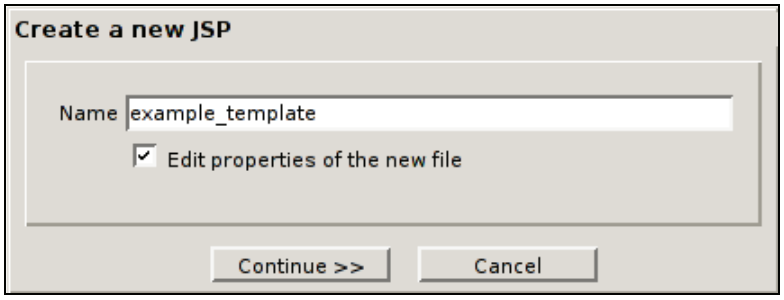
> In OpenCms 5.0 there were two folders for templates—`jsptemplates` for JSP templates and `templates` for XML templates. In OpenCms 6, JSP templates now go into the `templates` folder. XML templates are no longer used (unless you install legacy support for them). However, you may sometimes still see the folder `jsptemplates` used in modules—often for housing so-called "hidden" templates that editors cannot use directly.

Some folders, such as `templates`, are explicitly referenced by OpenCms code. This is not the case for the `resources` and `elements` folders. Their naming and use, though, is a convention encouraged by the OpenCms development community, so while there is no requirement to create these folders, or even to structure your module the way we are doing it here, you are encouraged to follow the established convention.

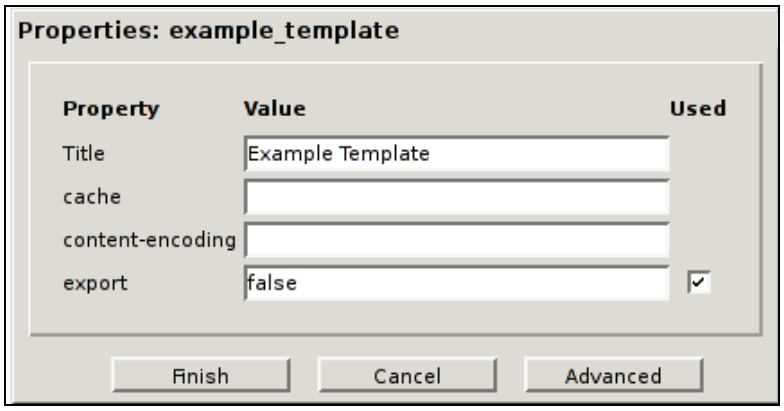
Now that we have the necessary folders, we can create our first template.

Creating a New Template

The first thing we will do is create a new JSP page. To do this, navigate to the `template` folder of the module, and click on the New button (with the magic wand icon) to create a new file. Choose JSP from the list and continue. The first thing you will need to do is give the new JSP a name.



Next, you will be prompted to edit the properties of the file:



You will definitely want to set the Title field to something distinct. OpenCms displays the title of a template in the template drop-down list that content creators will see when creating a new file.

The cache field allows you to set specific caching properties, which determine how the template is stored in the OpenCms FlexCache. Usually, it is best to leave this blank.

The content-encoding field allows you to specify a particular type of content encoding (ISO-8859 or UTF-8 for example). You only need to set this when the encoding of the template is different from the default encoding for OpenCms and the content that will be rendered in this template (that is the Page files) will also use this content encoding. In short, you will probably not need to give this field a value.

The last field is called export. This determines whether the template will be exported the first time it is accessed. By default, its value is false. If this value is true, then the template will only be executed once, and the resulting HTML will be stored on disk. When it is next requested, the static HTML will be served, and the JSP will not be executed again.

This is useful where the server is under high load and the contents of the template are not particularly time sensitive. Usually, you will want the template file to be executed every time the file is accessed so that you will want to leave the export field set to its default value, false.

Clicking Finish will create the template file and return you to the explorer view, where you should now see your new template file. Click on the file's icon and choose Edit sourcecode to edit the new template.

The JSP Template

The template will provide the layout for the contents of any associated pages. A template can include JSP tags and scriptlets, and it can also import other JSP files. In this section, we will create a simple JSP template. In subsequent sections, we will expand this template.

Here is an example of a very simple JSP template. It creates a basic HTML document and puts the contents of the requested file into the main body of the page.

```
<%@ page session="false" %>
<%@ taglib prefix="cms"
  uri="http://www.opencms.org/taglib/cms" %>
<!DOCTYPE html public "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title><cms:property name="Title" escapeHtml="true"/></title>
</head>
<body>
  <h1><cms:property name="Title" escapeHtml="true"/></h1>
  <cms:include element="text1"/>
</body>
</html>
```

The first two lines contain basic JSP tags that you are very likely to need in all of your templates. The `page` tag instructs the JSP engine not to create a session object. While some applications will require session objects, this one will not, and we can speed things up by not tracking sessions.

As we saw earlier in this chapter, the `taglib` tag tells the server that we are using the OpenCms tag library (whose URI is `http://www.opencms.org/taglib/cms`) and that any elements with the `cms` prefix will be from this tag library.

Most of the template is simply HTML. Only the tags prefaced with `cms` are interpreted by the JSP engine. The rest are sent "as is" to the client. In this template, there are only two elements from the OpenCms tag library. The `property` tag occurs twice—once to include the `Title` of the page as the contents of the HTML `title` element and once to include the `Title` of the page inside the `h1` element. Let's take a closer look at this tag:

```
<cms:property name="Title" escapeHtml="true"/>
```

The prefix `cms` instructs the server to use the element `property` from the OpenCms tag library (because the `taglib` tag tells the server that any elements with the `cms` prefix will be from this tag library). The `property` element in the OpenCms tag library provides access to the properties of the requested document. (See the discussion of properties in Chapter 3 and Chapter 4.)

The `property` tag has three attributes—`name`, `file`, and `escapeHtml`. The `name` attribute indicates that the property we want to access is called `Title`. The values of attributes are case sensitive—you cannot, for example, have `title` instead of `Title`.

The attribute `escapeHtml`, which is set to `true`, instructs the JSP interpreter to escape any HTML tags that exist in that property. For example, if the title of the page was "Using the <pre/> tag", it would be converted to "Using the >pre/< tag". This prevents the contents of a property accidentally being interpreting as HTML.

The `file` attribute allows you to specify the name of the file whose properties you are trying to access. If no file is specified, as happens in this example, then the properties of the current file are used.

> You can view all the properties for a file by locating the file in the explorer view, clicking on the file's icon, and selecting Properties from the popup menu.

The second OpenCms JSP tag used in the example above is the `include` tag:

```
<cms:include element="text1"/>
```

This tag gets the contents of the requested file and puts them inside the template for delivery to the client. Here, things get a little more complicated.

Each Page file in the VFS is stored inside a special XML file that describes the document. This XML file may include content in different languages, and it may have several different subsections. For example, some Page documents can have several "sub-documents" nested inside of them. These sub-documents are called **elements**. By default, these elements are named according to a standard convention, where the first is `text1`, and each subsequent element just increments the digit at the end—`text2`, `text3`, and so on.

By default, when we create new documents, we are only creating the first element, `text1`, so for this basic template, we just want to display the first (default) section of the requested document. The `include` tag is used to specify the element that we want to display by setting the value of the `element` attribute to `"text1"`.

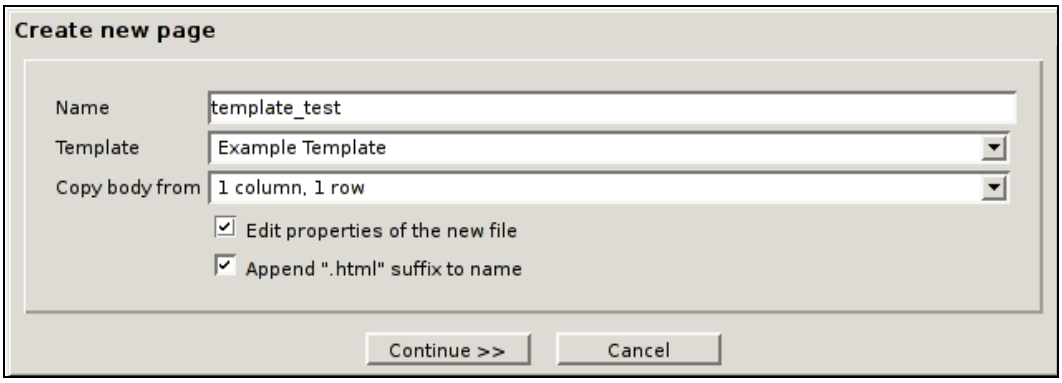
> The `element` attribute is required. When using the OpenCms `include` tag, you must specify a particular element—otherwise, OpenCms will generate a `NullPointerException` when it tries to load the template.

Once this new template is saved, we can test it out.

Testing the New Template

Choose /sites/default from the Site drop-down list in the explorer view, and navigate back to the `playground` folder created in Chapter 3. We will create a new file in this folder that uses the new template.

Click on the New button in the explorer toolbar and select Page from the list of file types. The next screen will prompt us to name the file and set a few crucial properties, including which template the page will use.

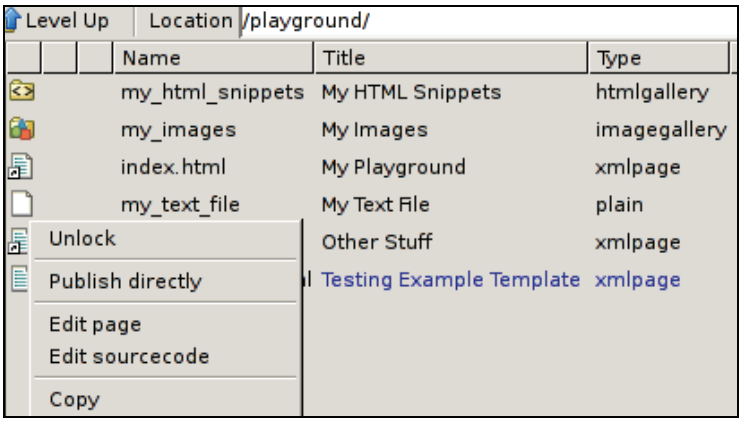


The new template should be available in the Template drop-down list.

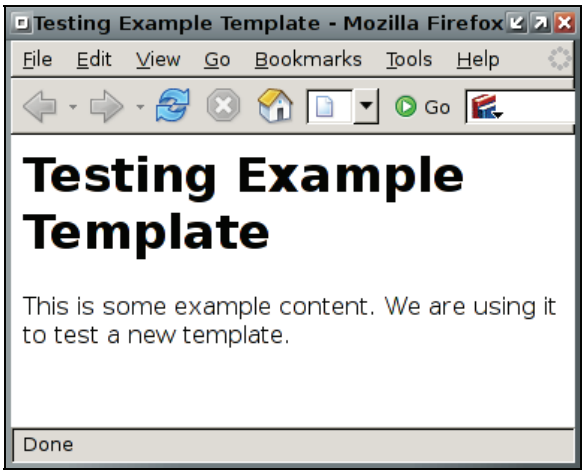
How does this template get added to the Template drop-down list? OpenCms searches all the template folders in the modules and adds any file it finds to the template list. If the file has a Title property, that property is displayed here.

Once the name and template are set, continue on to the properties screen, set any desired properties, and click Finish. (If you don't remember this process, you can look back at Chapter 3.)

We are now back in the explorer view looking at the new file. Now click the file's icon and choose Edit page to add some content to the file.



Once we have some text in the file, we can save the file and test the template. From the explorer view, we can click on the file's name to see a preview of the file. The preview will open in a new window and should look something like this:



Here is what the output looks like in HTML:

```
<!DOCTYPE html public "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Testing Example Template</title>
</head>
<body>
  <h1>Testing Example Template</h1>
  This is some example content. We are using it to test a new
template.
</body>
</html>
```

The two `property` tags have been replaced by the title of the document, and the `include` tag has been replaced with the contents of our page.

We will now make a few improvements to the template. First, we will create a stylesheet for our template.

Using Resources in a Template

Earlier, we created a `resources` folder for our module. This is where stylesheets, JavaScript files, and images are stored. Right now, let's create a stylesheet in the `resources` folder.

A stylesheet is composed of plain text, so when we create a new file, it needs to be of type Text. We will name the new file `main.css`. Once the file is created, we can edit it:

```
/*
 * Cascading Style Sheet for Example OpenCms Templates
 */
h1 {
  color: navy;
}

body {
  font-family: serif;
}
```

This is a very basic stylesheet that will change the color of the contents of the `h1` element to navy blue and will set the default font of anything in the `body` element to the browser's default serif font.

Now we need to insert this into the template:

```
<%@ page session="false" %>
<%@ taglib prefix="cms"
  uri="http://www.opencms.org/taglib/cms" %>
<!DOCTYPE html public "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title><cms:property name="Title" escapeHtml="true"/></title>
  <link type="text/css" rel="stylesheet"
href="<cms:link>/system/modules/tv.alephnull.modules.templates/resour
ces/main.css</cms:link>"/>
</head>
<body>
  <h1><cms:property name="Title" escapeHtml="true"/></h1>
  <cms:include element="text1"/>
</body>
</html>
```

The only addition is the `link` tag (which is nested within the `href` parameter). This tag takes the VFS path of a file, and converts it into a URI that the web browser can use to request the resource. Any time you are referencing a VFS file in an OpenCms template, you should use the OpenCms `link` tag (`cms:link`). We will look at this tag later.

Making a File Editable

In OpenCms 6.2, you can configure the template to allow logged-in and authorized users the ability to edit a file on the spot. This is called "direct edit". With direct editing, when a content editor is logged in and viewing the site, she or he will see buttons that, when clicked, allow them to edit the resource right there, without having to go through the OpenCms Workplace.

This feature is enabled in the template file. To add it, you will need to add one line and modify one line:

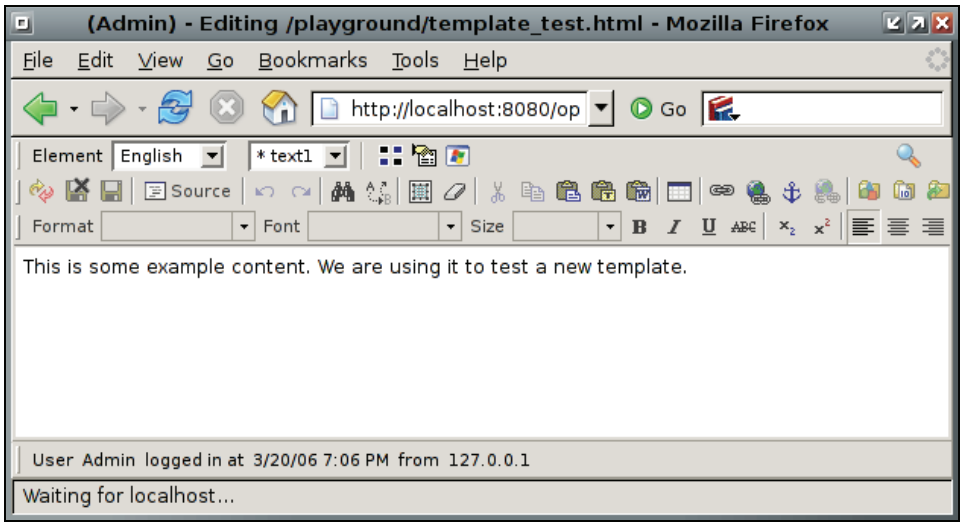
```
<%@ page session="false" %>
<%@ taglib prefix="cms"
  uri="http://www.opencms.org/taglib/cms" %>

  <!DOCTYPE html public "-//W3C//DTD HTML 4.01 Transitional//EN">
  <html>
  <head>
    <title><cms:property name="Title" escapeHtml="true"/></title>
    <link type="text/css" rel="stylesheet"
href="<cms:link>/system/modules/tv.alephnull.modules.templates/resour
ces/main.css</cms:link>"/>
    <cms:editable/>
  </head>
  <body>
    <h1><cms:property name="Title" escapeHtml="true"/></h1>

  <cms:include element="text1" editable="true"/>

  </body>
  </html>
```

The first highlighted line, located in the head, contains the `editable` tag from the OpenCms tag library. This tag instructs OpenCms to load the editing CSS and JavaScript files.

The second highlighted item is the `include` tag from the OpenCms tag library. The only change to this is the addition of the `editable` attribute, which has the value `true`. This tells OpenCms to allow this particular piece of content to be edited with the direct edit feature.

Now, when we view the test file (in the `/playground` folder), it looks like this:

Clicking on the target button on the right side will load the content editor:



To disable the direct edit feature, simply remove the `editable` attribute from the `include` tag. For good measure, you should probably remove the `editable` tag as well, but that is not necessary.

Next, we will look at loading data stored in other JSP files into the template.

External Elements

Sometimes it is useful to store parts of the template in separate files. You might want to do this to reduce clutter in the template, or you might want to do this so that one particular piece of template code can be used by multiple templates.

Code that generates navigation is frequently needed and often reused. Unfortunately, there is no JSP tag to do this, so we will have to resort to a scriptlet. However, by putting this scriptlet into its own file, we can re-use it in multiple templates.

The best place to store these files is in the module's `elements` folder. We'll create a JSP file in that folder called `navigation.jsp`. This file will be a short scriptlet.

A Short Scriptlet

The following scriptlet fetches the navigation information from OpenCms and then prints it out. This cannot be done with JSP tags. Otherwise, we would create our navigation that way.

If you are not a Java programmer, you can feel free to skip this section, or you may want to copy the code and use it in your own applications. After the code, there will be a brief explanation.

The contents of the file look like this:

```
<%@ page session="false"
  import="java.util.Iterator,
          java.util.List,
          org.opencms.jsp.CmsJspNavBuilder,
          org.opencms.jsp.CmsJspNavElement,
          org.opencms.jsp.CmsJspActionElement"
%>
<%
/*
 * Provides basic site navigation.
 */
// Create the class from which we will get navigation.
CmsJspActionElement cms =
  new CmsJspActionElement( pageContext, request, response );

// Get navigation info
CmsJspNavBuilder navigation = cms.getNavigation();
List navItems = navigation.getNavigationForFolder();
Iterator i = navItems.iterator();

// Loop through all of the items in the ArrayList and print the
// menu.
while( i.hasNext() ) {
  CmsJspNavElement navElement = ( CmsJspNavElement )i.next();
  String link = cms.link( navElement.getResourceName() );
  String title = navElement.getTitle();
  out.println("&raquo;<a href=\"" + link + "\">" + title +
              "</a><br/>");
}
%>
```

This scriptlet looks fairly daunting. In fact, it uses many important features of the OpenCms scriptlet API, but it is actually quite simple.

Let's break it down into smaller chunks and examine how it works.

The `page` tag imports all the Java classes that will be needed. Those whose names begin with `java.` are built into the Java programming language. Those whose names begin with `org.opencms.` are OpenCms Java classes.

```
<%@ page session="false"
  import="java.util.Iterator,
          java.util.List,
          org.opencms.jsp.CmsJspNavBuilder,
          org.opencms.jsp.CmsJspNavElement,
          org.opencms.jsp.CmsJspActionElement"
%>
```

The next thing we do is create a `CmsJspActionElement` object named `cms`. This object provides access to the OpenCms navigation information.

```
CmsJspActionElement cms =
  new CmsJspActionElement( pageContext, request, response );
```

Once this object has been created, we need to get the navigation information from it. This is done in the next three lines:

```
CmsJspNavBuilder navigation = cms.getNavigation();
List navItems = navigation.getNavigationForFolder();
Iterator i = navItems.iterator();
```

The first line gets a `CmsJspNavBuilder` object, which has all of the necessary information about navigation. We use the `getNavigationForFolder` method to get a list of all of the items in the current folder and store it in a variable called `navItems`. Once we have the list, we need an `Iterator` object that will allow us to walk through the list in order.

Now we are ready to go through the list, one item at a time, and print out the navigation links:

```
while( i.hasNext() ) {
  CmsJspNavElement navElement = ( CmsJspNavElement )i.next();
  String link = cms.link( navElement.getResourceName() );
  String title = navElement.getTitle();
  out.println("&raquo;<a href=\"" + link + "\">" + title +
              "</a><br/>");
}
```

While there are more items, this code will:

1. Get the navigation element, which stores information about a particular file (line 2).
2. Create a link for that file (line 3).
3. Get the title of that file (line 4).
4. Print out a hyperlink that points to that file (line 5). Don't get too distracted by all of the quotes and backslashes in the `println` method. The backslashes help Java understand which quotes need to get written to the HTML output.

That's all there is to it. All we need to do now is add it to the template.

> The OpenCms documentation that comes bundled with OpenCms includes more documentation on working with the navigation objects. If you are interested in improving on this simple navigation script, you can start by looking there.

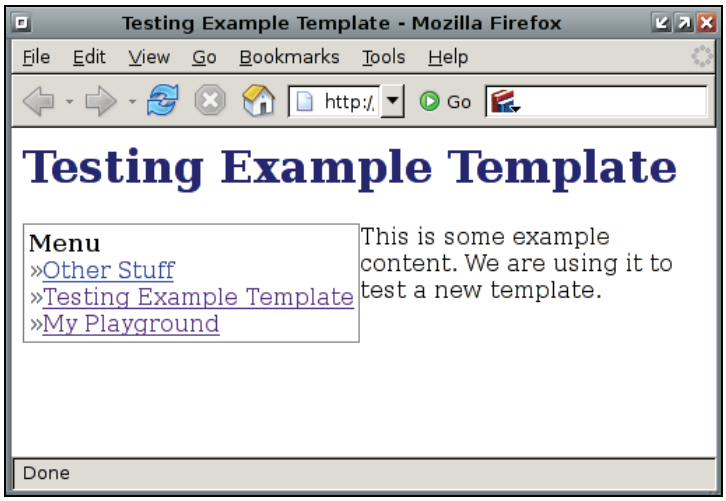
Including the Scriptlet in the Template

Now that we have a navigation scriptlet stored in the `elements` directory, how can use it in the template? A simple addition to our template will add navigation. Here is the template file, with the new addition highlighted:

```
<%@ page session="false" %>
<%@ taglib prefix="cms"
  uri="http://www.opencms.org/taglib/cms" %>

  <!DOCTYPE html public "-//W3C//DTD HTML 4.01 Transitional//EN">
  <html>
  <head>
    <title><cms:property name="Title" escapeHtml="true"/></title>
    <link type="text/css" rel="stylesheet"
href="<cms:link>/system/modules/tv.alephnull.modules.templates/resour
ces/main.css</cms:link>"/>
    <cms:editable/>
  </head>
  <body>
    <h1><cms:property name="Title" escapeHtml="true"/></h1>
    <div name="menu"
        style="float:left;border:1px solid gray;padding: 3px;">
      <b>Menu</b><br />
      <cms:include file="../elements/navigation.jsp"/>
    </div>

  <cms:include element="text1" editable="true"/>

  </body>
  </html>
```

The `div` element is just used to set the navigation off from the rest of the text. In fact, the only functional change above is the addition of the OpenCms `include` tag that points to the `elements/navigation.jsp` file.

If we view our test file again, it should have a navigation menu that looks something like this:



Our simple template has now grown a bit.

Next, we will explore the OpenCms tag library in more detail. Along the way, we will revisit some of the tags that we have used already, and we will discuss some new tags.

More on JSP Tags

We have looked at the basics of JSP and Scriptlets and even created a basic template using JSP tags. We will now look at JSP tags in more detail and at how they can be used within HTML documents.

JSP Directives

The JSP tag libraries use HTML-like (or XML-like) element syntax for including dynamic functionality in a document. Core JSP tags, or **directives**, are characterized by the percent sign, `%`.

For example, the JSP `page` directive from the JSP template earlier in this chapter looked like this:

```
<%@ page session="false" %>
```

Core tags are used to provide information to the server about how the page is to be processed. So far, we've looked at the `page` directive, which provides the server with information on how to treat the page, and the `taglib` directive, which tells the server what JSP tag libraries will be used in the page.

A JSP **expression** is a Java statement that is converted to a string and automatically printed. JSP expressions also have a percent sign. The following JSP expression, for example, prints out the date and time:

```
<p>Right now, it is <%= new java.util.Date() %></p>
```

New XML Syntax for Core JSP Tags

Recent developments in JSP technology, especially in version 1.2, encourage moving from the original JSP percent-style tag format to an XML-compliant syntax. Core tags have been redefined to use XML namespaces, using the `jsp` prefix instead of the percent-style notation. The new XML syntax is not completely supported by OpenCms. While you can sometimes get away with using it, you might want to consider a potential drawback.

In the Chapter 4, I pointed out that the FlexCache mechanism stores JSP pages in the real file system, as well as the VFS, so that those pages can be executed by the servlet engine. When these files are moved from the VFS to the local file system, OpenCms must replace VFS path information with absolute file-system paths. To accomplish this, OpenCms scans the JSP files, trying to replace paths in tags that need replacement. However, this scanning mechanism does not understand all of the JSP tags—in particular, it does not

understand any of the XML-style JSP tags. It ignores any tag that it cannot interpret. Consequently, XML-style tags that need path translation will be ignored. I will try to point out particular instances of this as we go through examples, and I will show examples of XML syntax when appropriate. But, if you choose to use the XML-style JSP syntax, be wary of tags that reference file-system paths.

The remainder of this book will use only the old syntax.

The OpenCms Tag Library

The OpenCms tag library is a built-in component of OpenCms. It provides access to the data managed by OpenCms and provides basic functions for presenting OpenCms data in a template. To begin this examination of the OpenCms tag library, we will return to the JSP template that we created earlier in this chapter.

Breaking a JSP Template into Sections

The first OpenCms JSP tag that we will examine is the `template` tag.

Our original template simply printed the title, generated a rudimentary navigation menu, and included the body of the requested page. This worked fine for our examples, but we can make the template much more versatile with a few extensions. Doing this can make it useful for other JSP pages, as well as for advanced pages.

We want to break the template into pieces that can be identified and selectively used. In our first example, there was no way to address a particular part of the template. JSP pages that use a template need to be able to identify where the top (head) of the template should go and where the bottom (foot) should go. Here is our previous example partitioned into a header, body, and footer.

```
<%@ page session="false" %>
<%@ taglib prefix="cms"
  uri="http://www.opencms.org/taglib/cms" %>

<cms:template element="head">
  <!DOCTYPE html public "-//W3C//DTD HTML 4.01 Transitional//EN">
  <html>
  <head>
    <title><cms:property name="Title" escapeHtml="true"/></title>
    <link type="text/css" rel="stylesheet"

href="<cms:link>/system/modules/tv.alephnull.modules.templates/resour
ces/main.css</cms:link>"/>
    <cms:editable/>
  </head>
  <body>
    <h1><cms:property name="Title" escapeHtml="true"/></h1>
    <div name="menu"
        style="float:left;border:1px solid gray;padding: 3px;">
      <b>Menu</b><br />
      <cms:include file="../elements/navigation.jsp"/>
```

```
    </div>
</cms:template>

<cms:template element="body">
  <cms:include element="text1" editable="true"/>
</cms:template>

<cms:template element="foot">
  </body>
  </html>
</cms:template>
```

As you can see, there are three `template` sections called `head`, `body`, and `foot`. Each of these sections can be called by name from an OpenCms `include` tag.

When processing page files, OpenCms reads through the template file in order, applying each template as it is encountered. (This means that, if you defined the foot before defining head, the foot would be printed before the head.)

None of the sections is added by default to a document of any other type besides Page. In a moment, we will look at the process of creating a JSP document that uses a template. In the process, we will see how this breaking up of the template can come in useful.

Finally, there is no hard and fast rule about `template` sections. While it is a good idea to call them `head`, `body`, and `foot` and some other OpenCms applications might assume their existence, you can add others if you wish.

For instance, I could define a template element called `hello` that contained a short message.

```
<cms:template element="hello"><p><b>Hello!</b></p></cms:template>
```

While it would show up by default in any pages, it would be accessible to other JSP files with a separate include. This mechanism allows you to conditionally apply template elements in JSP pages that are automatically applied to page documents.

Using Templates from a JSP Document

Just as we can create page documents in the OpenCms file system, we can also create JSP documents in the file system. JSP documents can be useful for handling forms or displaying dynamic information. But using templates with JSP pages is different from using templates with page files.

Now that our JSP template is complete, we can create a JSP document that uses it. We will create a JSP document called `/playground/cms-info.jsp`. Click the New button to create a new file, and choose JSP from the list of types. There is only one default parameter to set for a JSP file, Name, and we will set this to cms-info.jsp.

When we created the JSP template, we looked at the properties for a JSP page. While the `Title` property is useful (and will be used by the template), everything else can remain as the default specifies. But we are not ready to click Finish. We will need to use the Advanced button to edit one additional property for the JSP file. We will need to set the template that the JSP will use.

Near the bottom of the Individual Properties list in the Advanced Properties page, there is a field labeled template. We need to set this to point to our new template: `/system/modules/tv.alephnull.modules.templates/templates/example_template`.

> You can also access the Advanced Properties page for a file by clicking on the file's icon in the explorer view, choosing Properties, and clicking the Advanced button in the lower-right corner.

Click Finish to save the new file, and then open the file for editing.

We want to create a very simple JSP page that uses our template. We will start by creating the basic framework for the page:

```
<%@ page session="false" %>
<%@ taglib prefix="cms"
  uri="http://www.opencms.org/taglib/cms" %>
<cms:include property="template" element="head"/>

<p>This is a JSP page.</p>

<cms:include property="template" element="foot"/>
```

The page begins with the typical JSP declarations in which we import the `cms` taglib.

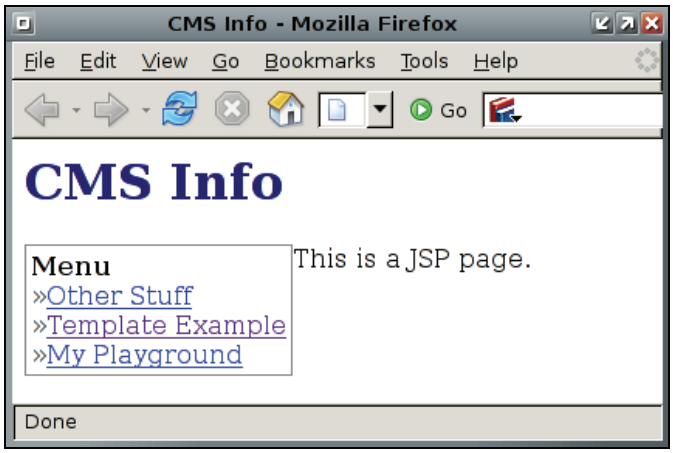
We have already assigned a template to this document, and we have broken the template into three sections. Now, we can selectively access the elements in that template.

The OpenCms `include` tags import the `head` and `foot` elements from the template. When this page is processed, these tags will be replaced by the contents of the `head` and `foot` elements.

Since we are not accessing the body of a particular Page file, there is no need to import the body element from the template. Instead, I just added a paragraph of text:

```
<p>This is a JSP page.</p>
```

When the above JSP is requested by the client, it will produce a page that looks like this:



We have seen how to create a JSP page and assign a JSP template to the page. Now, we will continue on and look at some other features of the OpenCms JSP tag library.

The property Tag

The JSP tag library provides some handy tools for providing information about resources within OpenCms. For instance, it is often useful to be able to get information about a particular file.

The OpenCms tag library's `property` tag gives us access to any of the properties associated with the file. For example, we can access the `template` property that we just set. To do this, we can simply add the following line above the final `include` in the JSP we created above:

```
<p>This file uses the template <cms:property name="template"/></p>
```

The `property` element will be replaced with the value of the `template` property—in this case, the absolute path in the VFS to the template: `/system/modules/tv.alephnull.modules.templates/templates/example_template`.

In fact, we are not strictly limited to the file at hand. Here's how we could grab the `Title` property from the `index.html` file in the same folder:

```
<p>The index.html file has the title
<b><cms:property file="index.html" name="Title"/></b>
</p>
```

In short, you can use the `property` tag to get the values of properties for files in the VFS.

The link Tag

Since we are referring to the `index.html` file anyway, we might as well provide a link to it. Linking within OpenCms could be very confusing. To write a link, you would have to know where the resource is in the VFS, whether the resource is served statically or dynamically, how Tomcat is configured, etc. Fortunately, you won't have to worry about these things. The OpenCms tag library provides a tag that handles this for you—the `link` tag.

The `link` tag is simple. It has no attributes and contains only the path to the file it will expand. For example:

```
<cms:link>index.html</cms:link>
```

This line of code generates the full path of the `index.html` file.

The above example works because the `index.html` file is in the same directory as our JSP. But if the file is in a different location in the VFS, then you will need to use the full VFS path, like `/sites/default/playground/index.html`. For an example of this, take another look at the JSP template we created at the beginning of this chapter.

To create a link to another file in OpenCms, you will need to create a hyperlink that looks something like this:

```
<p>
Go to <a href="<cms:link>index.html</cms:link>">
<cms:property file="index.html" name="Title"/></a>
</p>
```

When the server processes these directives, it will send the client a piece of HTML that looks like this:

```
<p>
Go to <a href="/opencms/opencms/playground/index.html">
My Playground</a>
</p>
```

The URL in the `href` attribute now has an absolute path to the `index.html` file, and the OpenCms `property` tag has been replaced with the full title of the file.

The user Tag

Another useful tag in the OpenCms tag library is the `user` tag. We looked at this tag briefly before, but now we will examine it in a little more detail.

The `user` tag provides access to information about the current user.

```
<h2>User Info</h2>
<p>The current user is <cms:user property="name"/>.</p>
<p>The current group is <cms:user property="group"/>.</p>
```

The `name` property will be replaced by the full name of the user, while the `group` property will be replaced by the user's current group.

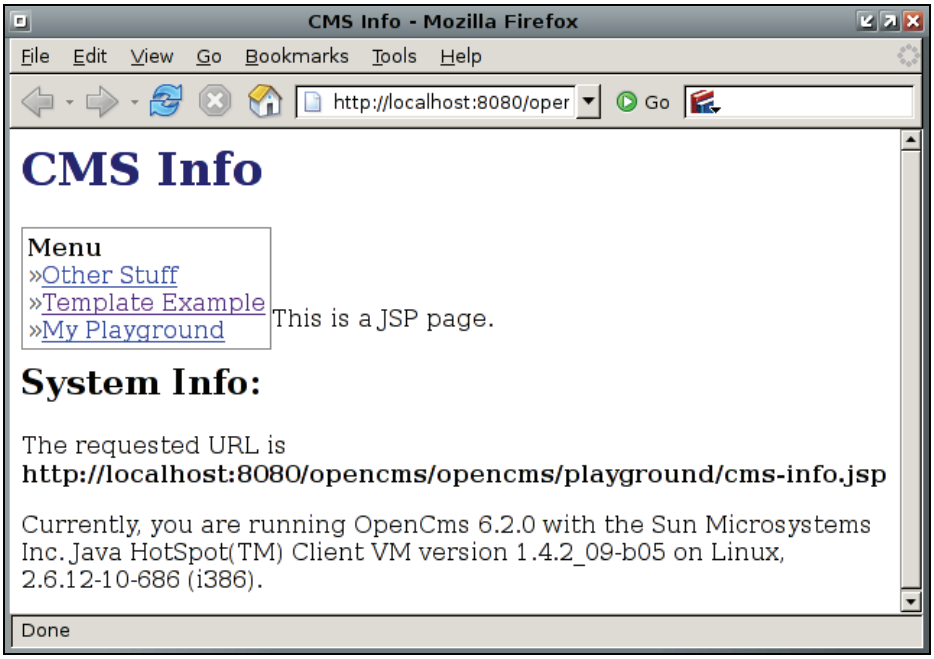
The `property` attribute determines which user information will be returned instead of the `user` tag. The available properties are `name`, `group` (or `currentgroup`), `defaultgroup`, `firstname`, `lastname`, `email`, `street`, `zip`, `city`, `description`, and `otherstuff`. The `otherstuff` attribute is not particularly useful. It is actually an unformatted dump of the contents of the `java.util.Hashtable` that stores extra user information.

The info Tag

The next tag in the OpenCms tag library that we will cover is the `info` tag, which returns information about OpenCms, the Java VM, and the underlying operating system. Syntactically, it is almost identical to the `user` tag. Here is an example that prints out the vital information about the platform:

```
<h2>System Info:</h2>
<p>The requested URL is <b><cms:info property="opencms.url"/></b></p>
<p>Currently, you are running OpenCms
  <cms:info property="opencms.version"/> with the
  <cms:info property="java.vm.vendor"/>
  <cms:info property="java.vm.name"/> version
  <cms:info property="java.vm.version"/> on
  <cms:info property="os.name"/>,
  <cms:info property="os.version"/>
  (<cms:info property="os.arch"/>).</p>
```

When this JSP is run, the client will see something like this:



In addition to information about the platform, the `info` tag provides a wealth of information about paths and URIs and can be very useful for constructing links when the OpenCms `link` tag will not do.

The img Tag

The `img` tag is new in OpenCms 6.2. It provides the functionality of the HTML `img` tag, plus automatic path adjustment for the VFS (like the OpenCms `link` tag) and server-side image scaling. The image scaling feature can be very useful. You can reference full-sized images in the VFS and use the `img` tag to generate thumbnails. The code for the OpenCms `img` tag looks the same as the HTML `img` tag:

```
<cms:img
  src="/playground/my_images/funny_head.jpg"
  alt="Funny Head"
  height="130"
  width="120"
  />
```

The value of the `src` attribute is the image's location in the VFS. OpenCms automatically re-writes this into the appropriate URL.

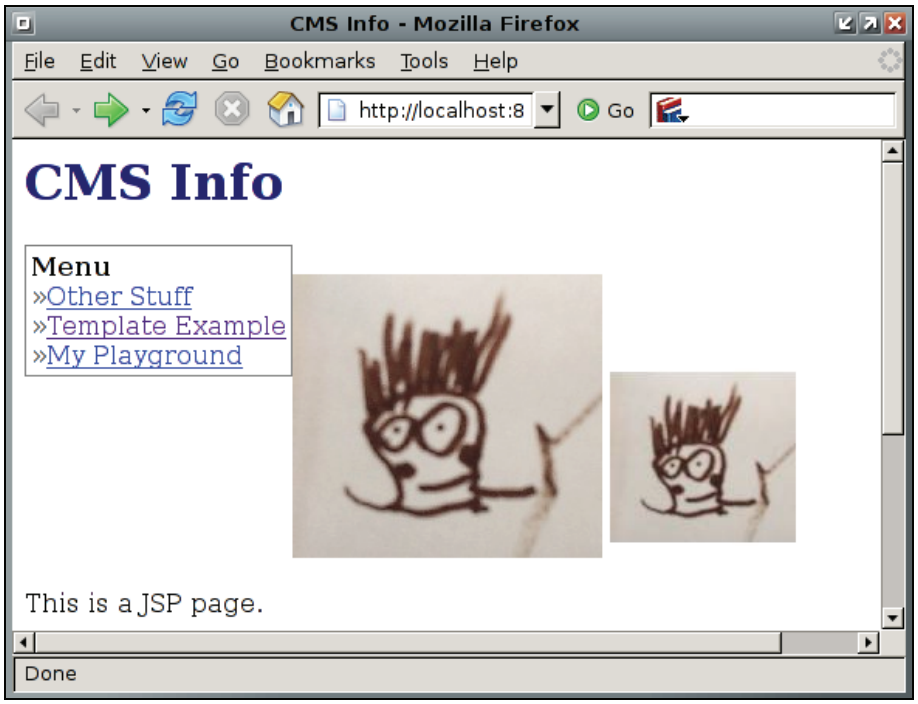
The `height` and `width` attributes specify the dimensions of the image. OpenCms uses this information to scale the image. In the example above, the original image is 200 × 180. OpenCms will scale the image down to 130 × 120 before sending it to the client's web browser.

If the dimensions do not maintain the same aspect ratio as the original image, then the background of the image will be padded—the image will not be stretched. That means you don't have to worry about distorting the image when scaling it.

The attributes of the regular HTML `img` tag, like `alt`, `style`, and `border`, are also supported by the OpenCms `img` tag. Here's a piece of code that contains both the OpenCms `img` tag and the regular HTML `img` tag:

```
<img
  src="<cms:link>/playground/my_images/funny_head.jpg</cms:link>"
  alt="Funny Head"
/>
<cms:img
   src="/playground/my_images/funny_head.jpg"
  alt="Funny Head"
  height="130"
  width="120"
  scaleType="stretch"
  />
```

When this code is run, the output looks like this:



The image on the left is the one using the HTML `img` tag. OpenCms has scaled the one on the right.

> Here is the URL that OpenCms generates: `http://localhost:8080/opencms/opencms/playground/my_images/funny_head.jpg?__scale=w:120,h:130`. You may notice that the scaling information is appended to the end. For images stored in the VFS, you can directly instruct OpenCms to scale images by providing this information at the end of a URL.

The decorate Tag

Another tag introduced in OpenCms 6.2 is the `decorate` tag. This tag can be used to take an existing piece of HTML and automatically add new formatting according to a pre-defined pattern.

For example, we could set up a `decorate` tag that would find all instances of "VFS" and surround them with the HTML abbreviation tag, `abbr`. This would take the text:

```
In OpenCms, all files are stored in the VFS.
```

and turn it into:

```
In OpenCms, all files are stored in the <abbr title="Virtual File
System">VFS</abbr>.
```

Since the `decorate` tag can be inserted into templates, it can be used to automatically decorate the contents of page files authored in the WYSIWYG editor.

Using the `decorate` tag is a little more complicated than other tags because the it uses a configuration file and one or more text files to figure out what to decorate and how to decorate it, so the first thing we will need to do is create the configuration files.

Decorator Configuration Files

There are two types of configuration files. First, there are CSV (Comma Separated Values) files that contain lists of the terms to be decorated. Also, there is an XML configuration file that contains information on how to decorate the terms in the CSV file.

The configuration information is stored in a shared folder: `/system/shared/decoration`. (Note that you will have to be able to access the `/system` folder.) We will create our CSV file in this folder.

To do this, create a text file in `/system/shared/decoration` called `decorationAbbr.csv`. This file will contain the fields that tell the decorator what to look for and what information needs to be added.

```
VFS|Virtual File System
CSV|Comma Separated Values
CMS|Content Management System
WYSIWYG|What You See Is What You Get
```

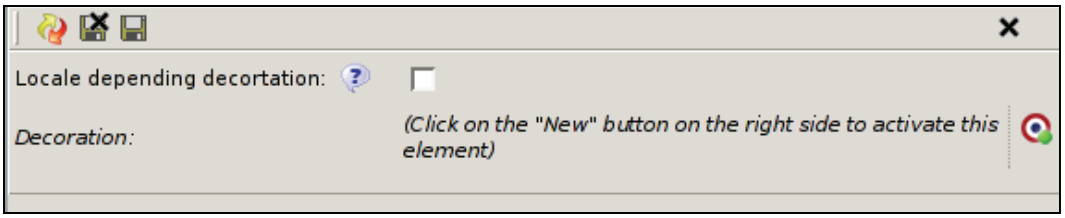
Note that the field delimiter above is not a comma (as the name CSV would suggest) but a vertical pipe ('|').

The first entry is the pattern that the decorator will look for. The second entry is the information that the decorator will need to decorate the text—in this case, the term that will be put in the `title` attribute of the `abbr` tag.

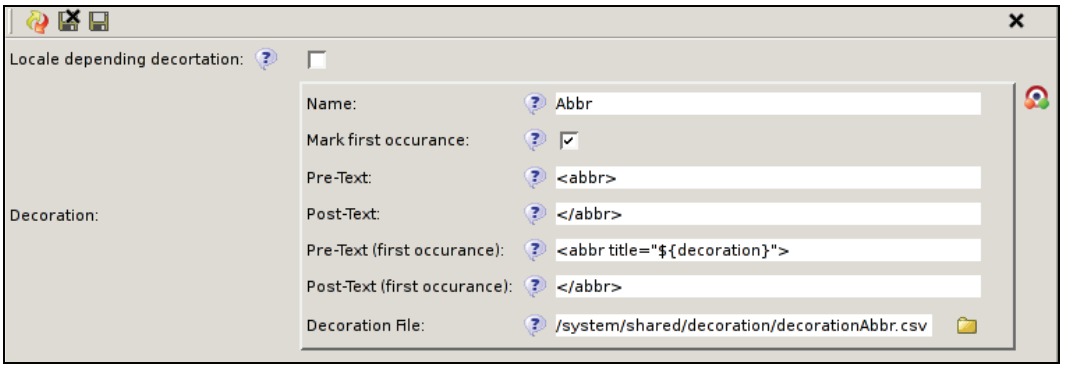
> In addition to the default decoration file, you can create decoration files for specific locales. For example, we could create a file named `decorationAbbr_de.csv` that contains decoration data for German-language locales. OpenCms allows multiple locales to be set using different files.

Once the `decorationAbbr.csv` file has been saved, we can work on the XML configuration file. OpenCms comes with a generic decoration XML configuration file, which is stored in `/system/shared/decoration/configuration.xml`. Rather than modify this file, we will make a copy of the file and edit that copy.

To copy the file, click on the `configuration.xml` file icon, choose Copy, and name the copy `my_decorations.xml`. Click on the file icon for `my_decorations.xml` and choose Edit from the menu. This will open the XML editor.



If you are going to set up multiple locales, check the Locale depending decoration checkbox. If this is checked, OpencCms will automatically search for locale-specific CSV files for this decoration. Next, click the target button on the right. This will bring up a context menu with a button shaped like a green plus sign. Click this button to add a new decoration definition.



Now, seven new fields are available. The first field, Name, is for giving this decorator a human-readable name.

The Mark first occurance checkbox, if checked, instructs OpenCms to treat the first occurrence of a match differently from subsequent versions of a match. In our case, we are adding `abbr` elements. For the first occurrence, we want the value of the `title` attribute to be "`Virtual File System`", but for subsequent occurrences, we don't want to assign the value to the `title` attribute at all, so in this case, the box should be checked.

The next two fields, Pre-Text and Post-Text, specify what should come before and after each match. For our example, Pre-Text is <abbr> and Post-Text is </abbr>. However, we want the first occurrence of each term to be marked with the title. The next two fields, Pre-Text (first occurance) and Post-Text (first occurance), are for specifying pre- and post-text for the first occurrence only.

For the first occurrence of any abbreviation, we want the `title` attribute to be filled using the second element in the CSV file created above. For `VFS`, the decorator should search the CSV file and find `VFS|Virtual File System`. We want that second field to be placed in the title attribute. So the field Pre-Text (first occurance) should be given the value, <abbr title="{$decoration}">. {$decoration} will be replaced by the second field in the CSV file.

> There is a second variable, aside from `{$decoration}`, that is available—the `{$locale}` variable, which will contain the name of the locale currently used.

The final field, Decoration File, should contain the full path reference of the CSV file that holds the definitions.

Once everything is complete, click the "save and close" button (the yellow floppy disk with a black X) to save the definition and exit the XML editor. We are now ready to test the decorator.

Using the decorate Tag

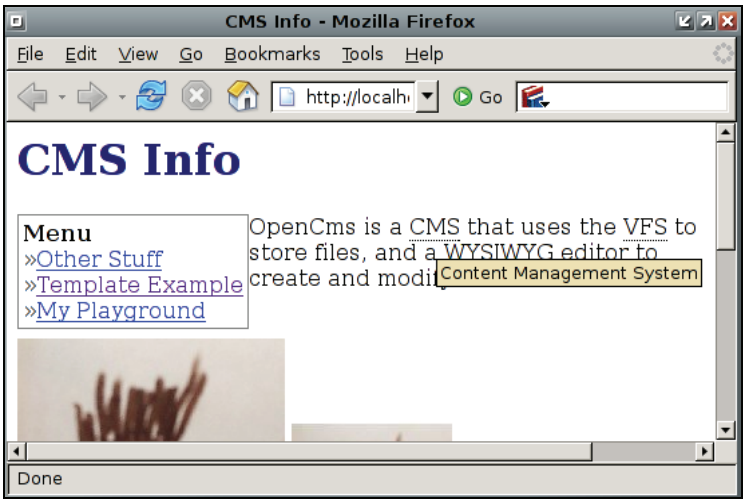
Going back to our JSP page in the `/playground` folder, we can now add a decorator section:

```
<cms:decorate file="/system/shared/decoration/my_decorations.xml">
<p>
OpenCms is a CMS that uses the VFS to store files, and
a WYSIWYG editor to create and modify content.
</p>
</cms:decorate>
```

The decorator will search through all of the text between the `decorate` tags and surround any matching terms with `abbr` tags.

The `decorate` tag has two attributes, `file` and `locale`. The `file` attribute tells the decorator where the XML configuration file is. The optional `locale` attribute lets you explicitly set which locale should be used.

The result of our JSP in action looks like this:



Note that the terms we specified in the CSV file, "CMS", "VFS" and "WYSIWYG", have a dotted underline. Hovering the mouse pointer over one of these terms (as I did above over the term CMS) displays a context box that shows what the abbreviation stands for ("Content Management System").

As I mentioned before, you can use the OpenCms `decorate` tag in a template file. That way, the contents of a page file are run through the decorator. But there is one caveat when doing this—you will need to turn off caching for the `import` tag. For example, applying a decorator to the body of a template would look like this:

```
<cms:template element="body">
  <cms:decorate file="/system/shared/decoration/my_decorations.xml">
    <cms:include element="text1" editable="true" cacheable="false"/>
  </cms:decorate>
</cms:template>
```

This is from the `body` section of the template. The `decorate` tag is the same as in the previous example, but to make sure the decorator works, we need to set the `include` tag's `cacheable` attribute to `"false"`.

That is all there is to using the `decorate` tag. Of course, the examples I've given above are fairly straightforward. The decorator is an advanced tool, and with a little creativity, you can use it to accomplish more complex processing.

At this point, we have walked through the basics of the OpenCms taglib. OpenCms includes additional documentation on tag libraries in the OpenCms JSP Taglib Documentation Alkacon documentation module.

Documentation and TemplateOne

Now that we have walked through the process of creating a simple module, you may want to move on to more sophisticated creations. OpenCms includes plenty of documentation to help you on your way. You may want to look at the OpenCms TemplateOne templates, which provide a sophisticated pre-built template environment (built with tag libraries, scriptlets, and Java objects). OpenCms includes built-in documentation on the TemplateOne templates.

At this point, we have covered the basics of template creation. The remainder of this chapter will explain how to avoid some of the common pitfalls of OpenCms template development.

A Few Things to Watch Out For

Whether using a tag library or writing scriptlets, there are a few things that can cause confusion and surprise to developers new to the platform.

The File System

The VFS, in spite of its name, does not behave exactly like a real file system. Furthermore, neither the servlet engine nor the JSP interpreter is aware of the existence of the VFS. When using JSP tags, be wary of the JSP `include` and `directive` tags, and the `import` tag from the Java Standard Tag Libraries (JSTL). These tags do not handle the VFS/real file system split very well and can have some unpredictable results. According to OpenCms developers, judicious use of the `import` tag will work for some files, but the caveats are many, so I do not recommend it. The best way to handle inclusions is to use the OpenCms tag library's `include` tag (as we have done already), or its API equivalent, the `CmsJspActionElement` class's `include` method.

The built-in Java IO classes are also not aware of the VFS, and it is not possible, for example, to open a `FileReader` object on a location inside of OpenCms. There should be no need for this, as the `CmsObject` class provides methods for VFS file access.

Redirecting and Forwarding

OpenCms provides a custom subclass of `HttpServlet`, and some of the behaviors of this subclass do not match what the JSP interpreter assumes. This is the case with forwarding (using the JSP `forward` tag), which simply does not work in OpenCms. In the OpenCms documentation, the recommendation is that developers attempt to use the tag library provided for handling the problem. However, redirecting is straightforward and can be done in a scriptlet by executing the `response` object's `sendRedirect` method.

Dynamic Content and Publishing

Development is usually done by a user who has administration privileges, and tests are often performed (as we've done so far) on the Offline project. However, security settings for a visitor to the site are markedly different from those for an internal administrator, and code that works during development may fail for security reasons when a visitor tries to execute it. Likewise, publishing files from the Offline project to the Online project can change expected behavior if some or all of the content is exported. When JSP pages are exported statically (not the default behavior), they will be rendered once into static HTML, and will lose all of their dynamic functionality. Likewise, hard-coded links that do not use the OpenCms tag library's `link` tag can break or point to non-existent documents.

Obviously, the solution to this problem is to be aware of the problem and test a lot. If you are doing static exports of content, make sure you test out the template code. When content is rendered as static HTML, the JSP template is also rendered once. In my early exposure to OpenCms, I discovered this when I used the `java.util` package's `Date` class in my main template. Two days after publishing my files as static resources, I realized with horror (or acute embarrassment) that the date hadn't changed since I published the project. Setting the `export` property to `false` for JSP files should keep this from happening.

Structuring Code and Content

OpenCms is designed to be flexible, allowing organizations to adapt it to their own needs. Because of this, JSP files can get scattered around the VFS, and valuable (and reusable) functionality can be lost in a labyrinth of content folders. A well designed development process, though, should not allow this to happen.

The OpenCms module structure provides an ideal mechanism for structuring code in a way that promotes reuse and organization. That is why we created a module before writing the first JSP. By structuring things wisely from the beginning, we avoid a JSP diaspora.

As a general rule, move as much JSP code as possible into a module, particularly code that performs useful functions. If you find that a large number of scripts have a functional similarity (e.g. a set of blogging scripts), you may decide to create a new module specifically for that set of tasks.

As an added advantage, modules are designed to be versioned and moved, and they can vastly improve the development-to-production process. The site can be designed and developed in a module on a development server and then exported. Then it can be imported into the testing and production environments. Since content exists separately (outside of the module), there is no risk of losing valuable content or migrating development content into production. The built-in versioning makes it very easy to quickly ascertain the state of each server.

Summary

In this chapter, we have explored the basics of customizing OpenCms, we have created a module for our own code, and we have created custom templates, resources and elements within that module. Using the OpenCms tag library and the scriptlet API, we have added useful functional elements to templates and documents.

At this point, you should have all of the tools you need to start customizing your own OpenCms site.

A
Cron Expressions

This appendix covers the string format for setting job schedules using the Scheduled Jobs Management screen in the administration view, and how to work with cron expressions using the revised format for OpenCms 6.

What are Cron Expressions?

OpenCms can run specific tasks at specific times, eliminating the need for a human operator to manually run tasks on a regular basis. As we saw in Chapter 4, the Scheduled Jobs Management screen in the administration view provides an interface for creating, modifying, and removing scheduled jobs.

One of the most confusing aspects using of this tool is setting the Cron expression field in the New Job screen to execute at the desired time.

A **cron expression** is a string of characters, organized into six or seven fields separated by spaces, that provides detailed information about when a task should be run. While cron expressions may initially look daunting, they are fairly simple to write and can be much quicker to use than alternative systems.

There are a number of pre-defined cron expressions from which you can select, but these times may not fit your needs. Here is a brief explanation of how these cron-like time settings work.

Changes from OpenCms 5

The OpenCms 5 series used a version of the cron expression that is rooted in the traditional UNIX `cron` server. In this format, there are only five fields (minute, hour, day, month, and day-of-week).

But OpenCms 6 has an extended cron expression format that is much more suitable for an enterprise environment. It has seven possible fields (with seconds and years added to the original five) and some convenient synonyms (such as month and day names that can be substituted for numbers in the appropriate fields).

Unfortunately, the changes are such that cron expressions that worked in OpenCms 5 will definitely not work in OpenCms 6.

Cron Expressions in OpenCms 6

A cron expression has six mandatory fields (or slots) and one optional field. Adjacent fields are separated by a space. The asterisk `*` is a wild card meaning "any legal values". In theory, we could construct an empty expression that would look like this: `* * * * * * *`. (This expression will not work in OpenCms for reasons explained below.)

> The source of information for this text is the JavaDoc (documentation contained in the source code) for the `org.opencms.scheduler` package's `CmsScheduledJobInfo` class. For more information, you can consult the source code.

Each field in a cron expression stands for a particular time period. Here is a list of the fields with all of their legal values:

1. Seconds: `0` to `59`, `*`
2. Minutes: `0` to `59`, `*`
3. Hours: `0` to `23`, `*`
4. Day-of-the-month: `1` to `31`, `L`, `*`, `?` (modifier: `W`)
5. Month: `1` to `12`, `Jan` to `Dec`, `*`
6. Day-of-the-week: `1` to `7`, `Sun` to `Sat`, `L`, `*`, `?` (modifiers: `L`, `#`)
7. Year (optional): `1970` to `2099`, `*`

Note that the first three fields use `0` as the first number, while the next three begin with `1`.

The day-of-the-month and day-of-the-week fields are mutually exclusive, so one of these two fields will always need to be empty. This is done using a question mark, `?`. For example, if I want to run a job at noon on the first day of every month, the cron expression will look like this:

```
0 0 12 1 * ?
```

Alternatively, if I want to run the job every Sunday, the string will look like this:

```
0 0 12 ? * 1
```

The month and day-of-the-week fields accept abbreviated names in lieu of numbers (e.g. month can have `Jan` to `Dec` instead of `1` to `12`). These are not case sensitive, so `MON`, `Mon`, and `mon` will all evaluate to `2`.

`L` in the day-of-the-month field indicates that the job should be run on the last day in the month. When it is used in the day-of-the-month field, the scheduler will run the job on the last day of the month.

`L` as a value in the day-of-the-week field means the last day in the week, so it has the same effect as `7`.

`L` can be also used as a modifier in the day-of-the-week field by appending it to a number, so that the job will be run on the last of that day in the month. For example, here is a cron expression to run a job at noon on the last Tuesday of the month:

```
0 0 12 ? * 3L
```

The `W` modifier in the day-of-month field indicates that the task is to be run on weekdays only. If the date falls on a weekend, then the task is delayed until the next weekday. For example, consider this cron expression, which runs a job at 1:00 AM on the first of every month:

```
0 0 1 1W * ?
```

`LW` indicates that a job is to be run on the last weekday in the month. For example:

```
0 0 1 LW * ?
```

If the first of the month fell on a Saturday, then the job would not be executed until Monday.

The hash sign, `#`, is used in the day-of-the-week field to indicate that the job should be run on the nth such day in the month. For example, to run a job at noon on the third Friday of the month, you would use a string like this:

```
0 0 12 ? * 6#3
```

As mentioned above, an asterisk, `*`, indicates *any value*.

A dash, `-`, indicates a range. For example, if I want to run a job once an hour for three hours in a row, I can use this cron expression:

```
0 0 1-3 * * ?
```

To set several different values in the same field, you can use commas to separate them. For example, to run four times in an hour, you could use the following:

```
0 0,15,30,45 * * * ?
```

it will then run every fifteen minutes. Do not put spaces before or after the commas.

Another way to do this would be to tell the job scheduler to run the job at various intervals. This can be accomplished with a forward slash, `/`. So instead of telling the scheduler to run at the specific minutes `0`, `15`, `30`, and `45`, we could tell it to run every fifteen minutes:

```
0 0/15 * * * ?
```

The `0` indicates the offset from the beginning of the hour. (`0` means there is no offset.) The `15` indicates that it should run every fifteen minutes. If we want the job to run at five minutes after the hour and then every fifteen minutes, we would have `5/15` instead of `0/15`.

The optional seventh field, year, can be specified to restrict the job to only run during the specified year or years. For example, to run a job at 7:15:30 AM every Tuesday during the first quarter of the years 2006 and 2007, I could use the string:

```
30 15 7 ? Jan-Mar Tue 2006-2007
```

The year must be between `1970` and `2099`, inclusive. I would not recommend using OpenCms 6 in the years following 2099.

Summary

In this appendix, we have examined the cron expression syntax, looking at examples along the way. Expressions in this format can be used to schedule tasks using the New Job screen in the administration view.

B
Upgrading OpenCms

In this appendix, we will cover upgrading from OpenCms 6.0.4 to OpenCms 6.2 or later.

Getting the Upgrade Package

The first thing you will need to do is get the upgrade package from the official OpenCms website, `http://www.opencms.org`. The OpenCms upgrade package is different from the OpenCms installation package since it contains a special wizard to guide you through the process of upgrading.

At the time of this writing, the most recent version of the updater is called `opencms_upgrade_to_6.2.1.zip`.

Once you have the package, you are ready to begin the installation process.

Preparing for the Upgrade

While the upgrade process has been tested and is reliable, there is always the possibility that something could go wrong. Therefore, before proceeding, you should make a backup copy of your OpenCms installation. You may want to export your VFS and your custom modules. (See Chapter 4.) You may also want to dump a copy of your database. (Consult your database documentation.)

> During the upgrade, OpenCms will re-install all of the core OpenCms modules, including TemplateOne and the documentation. If you have modified any of these modules, your changes will be lost during the upgrade. You may want to export the changed resources (rather than the whole modules) so that you can re-import them after the update.

It can also be a good idea to make a spare copy of the OpenCms web application directory. To do that, stop Tomcat (or whatever servlet container you are running), and copy the `$CATALINA_HOME/webapps/opencms` folder to another location.

The next thing to do is extract the contents of the `opencms_upgrade_to_6.2.1.zip` file. It is best to unzip the archive into a temporary directory.

Once you have unzipped the file, you should have three items in your working directory—the `WEB-INF` folder, the `update` folder, and the `readme.txt` file.

The `readme.txt` file will contain a brief set of instructions on the upgrade process. The `WEB-INF` folder contains the new and updated OpenCms files. The `update` folder contains the upgrade wizard.

Moving Files

At this point, we will start moving files. Make sure your servlet container (for example, Tomcat) is stopped. If it is running during the upgrade, you may corrupt the OpenCms installation.

Copy the `update` folder into the root directory of the existing OpenCms installation. With Tomcat, this is in `$CATALINA_OPTS/webapps/opencms`. (Chapter 2 tells you where these files are stored on the file system.)

Copy the contents of the `WEB-INF` folder from the upgrade package to the `WEB-INF` folder in your existing OpenCms folder. This will overwrite a number of files, which is exactly what you want—new versions of OpenCms libraries will replace the older versions.

There is one last thing to do before restarting the servlet container—we need to turn the wizard back on. After doing the original installation of OpenCms, the installer automatically disables the wizard interface. To do an upgrade, we need the wizard interface re-enabled.

With the text editor of your choice, open the file `$CATALINA_HOME/webapps/opencms/WEB-INF/config/opencms.properties`. At the very bottom of this file is a line that reads:

```
wizard.enabled=false
```

Change `false` to `true`, save the file, and close the editor.

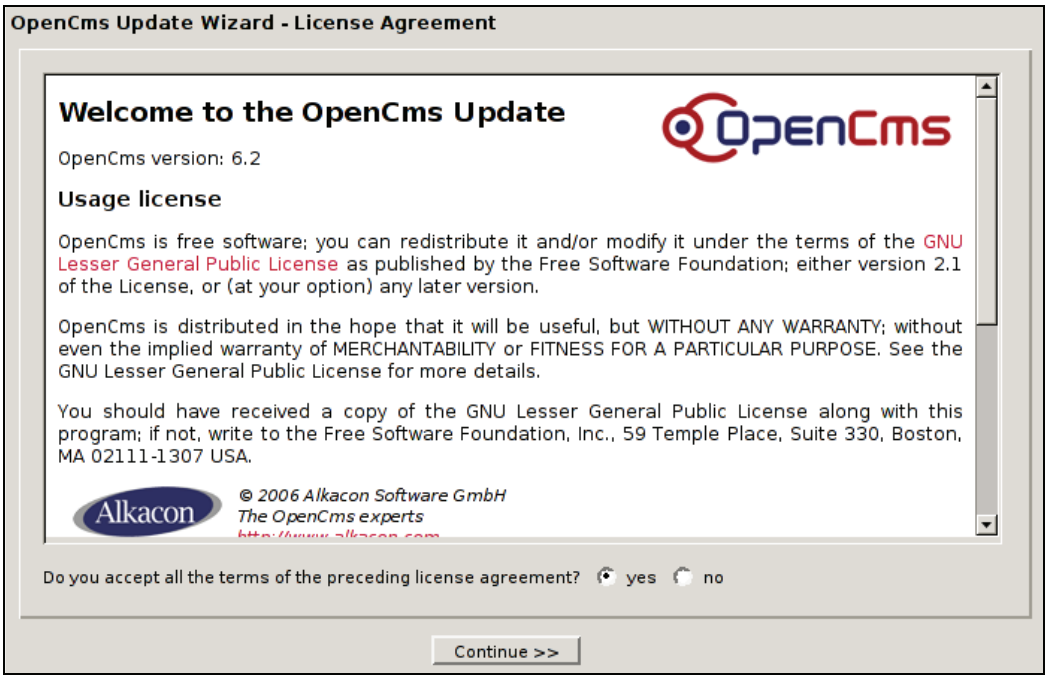
At this point, you can restart Tomcat.

Running the Upgrade Wizard

We are now ready to run the web-based upgrade wizard. Open a web browser and point it to the upgrade wizard. The URL should be something like `http://myserver:8080/opencms/update`.

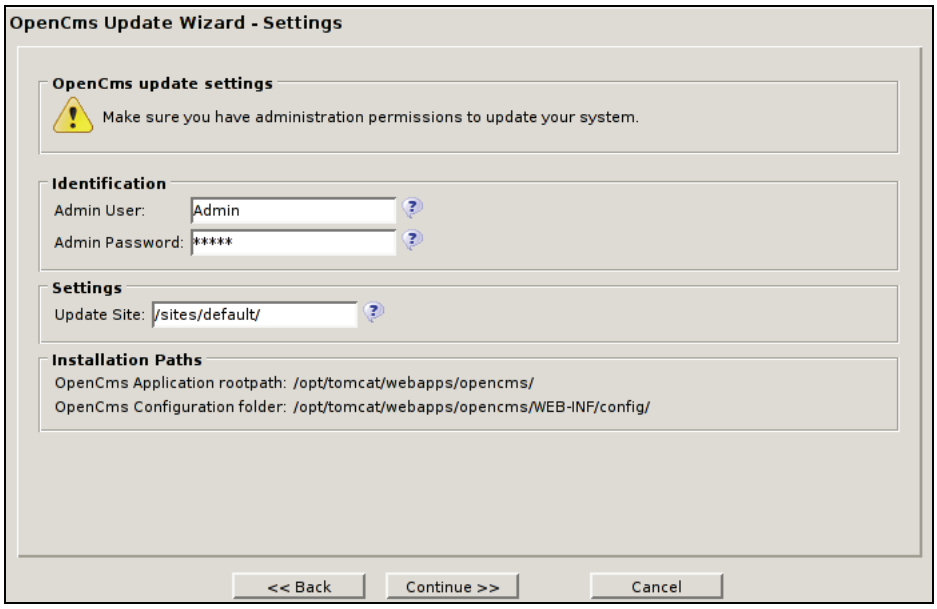
> As long as the wizard is enabled, you will not be able to access either the public OpenCms site or the Workplace.

As with the regular OpenCms installation, the first thing you will have to do is accept the OpenCms license agreement, which states that OpenCms is released under the GNU Lesser General Public License, an open-source license.

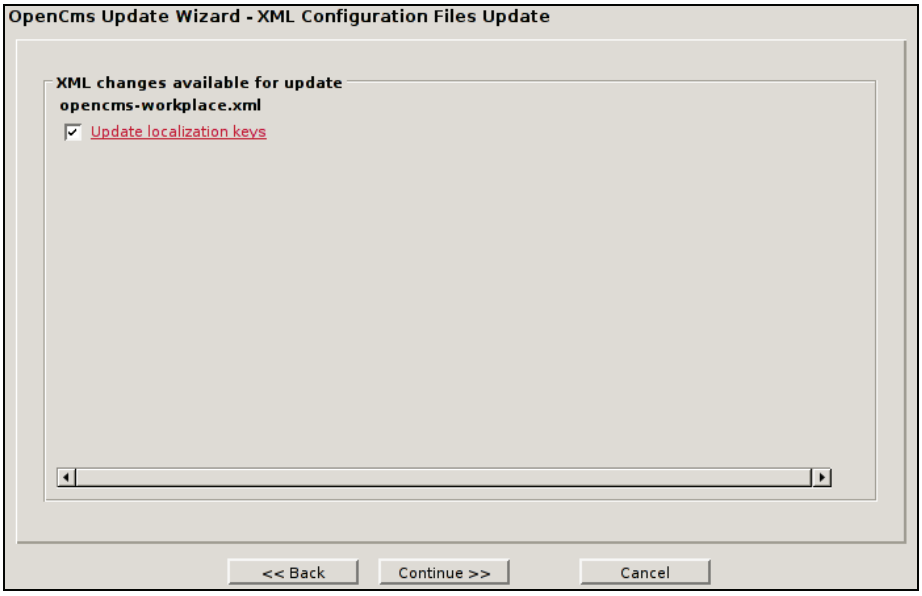


Once you accept the license, the Continue button will be activated.

The next screen will prompt you to enter some information that will be necessary for the upgrade. You will need to provide the administrator user name and password, since the update utility requires access to the entire VFS.

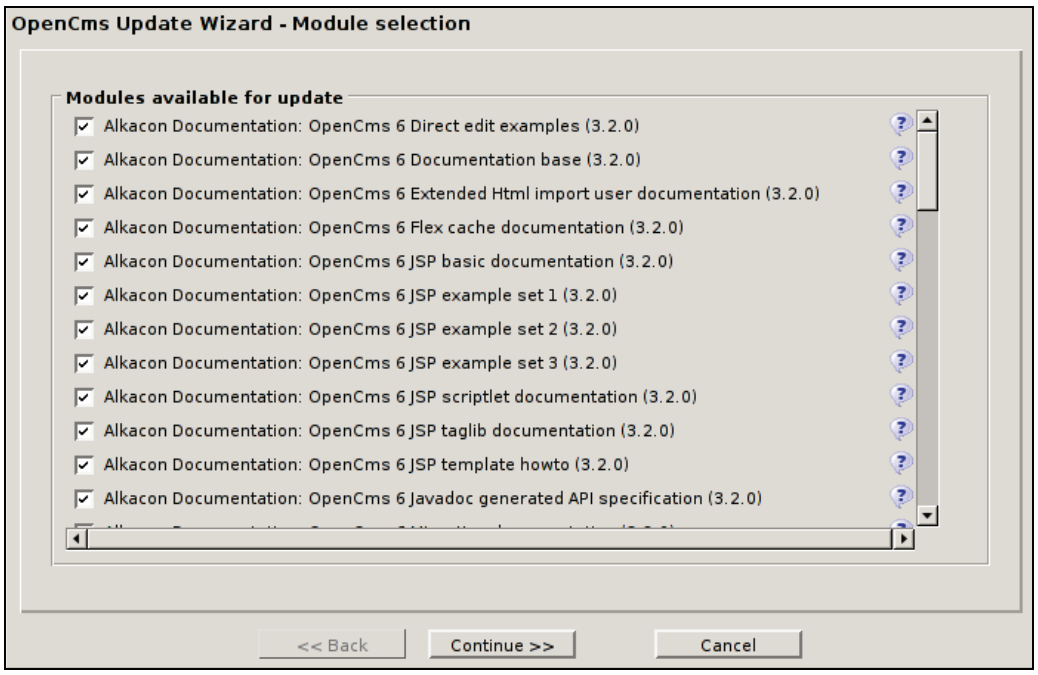


Once you have entered the correct information, clicking Continue will load test the settings and load the next screen. If something you have entered is incorrect, the wizard will prompt you to start over again. Otherwise, it will take you to the XML Configuration Files Update page.



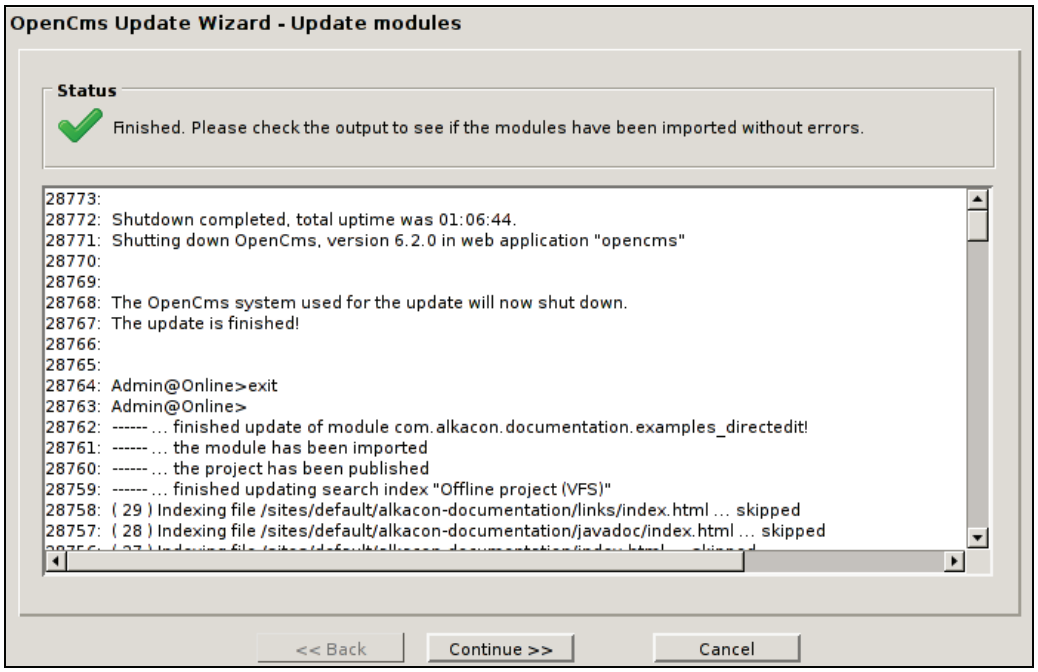
This page will list all of the updated XML configuration files. If you are migrating from the OpenCms 6.2 beta release (as is shown above), there is only one such file. Other versions of OpenCms may have more XML files in need of updating. Unless you have custom versions of these XML files, you should leave all of these selected.

Clicking Continue installs the XML updates and then loads the module selection screen.



This screen shows all the currently available modules, and all of them are checked by default. On a production server you may not want the documentation modules. For other reasons, you may have already identified other modules you do not want. If you know you do not want some of these modules and you know that uninstalling them will not cause problems, you can deselect the undesired modules. Otherwise, the best course of action is to install all of them. You can delete the unwanted ones later. (See Chapter 4.)

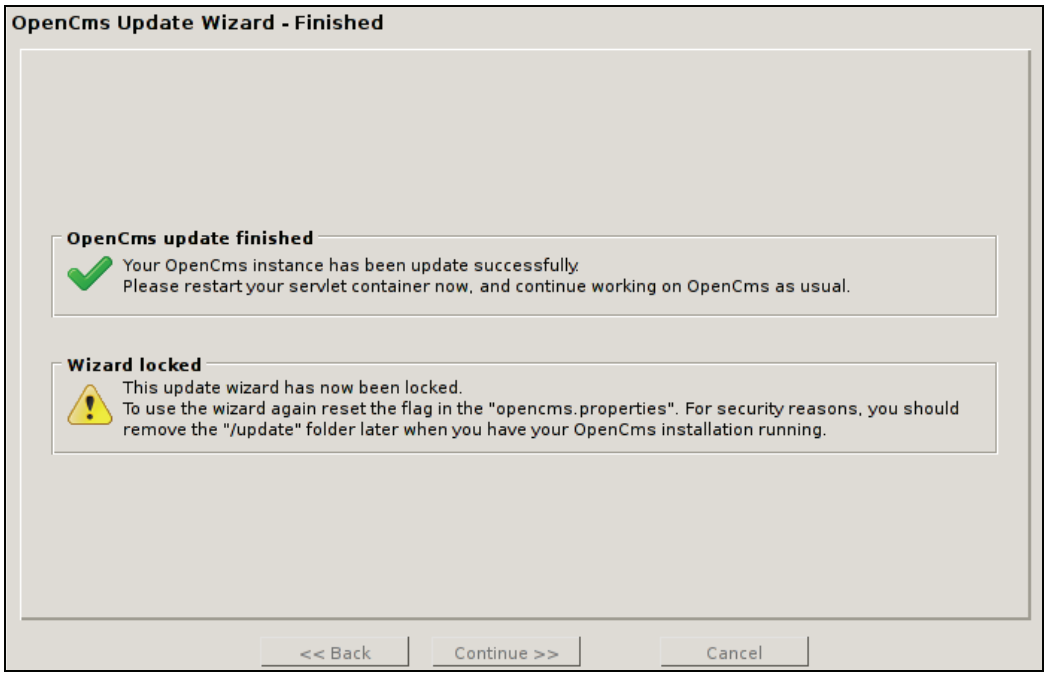
Clicking `Continue` will bring up a progress screen, which will display information as the application installs.



Even for an update, this process can take a long time.

Final Steps

The last screen of the upgrade wizard will prompt you to restart the server.



At this point, the update wizard is complete. You can safely close your web browser. But remember, you need to restart Tomcat to ensure that all of the changes take effect.

> OpenCms will automatically disable the wizard again, and you will not be able to rerun the wizard without changing the settings in the `opencms.properties` file.

You can now view your OpenCms site and log back into the OpenCms Workplace.

Summary

At this point, you should have version 6.2 of OpenCms running. Version 6.2 has many new features and modifications. Many of these new features—especially the ones in the OpenCms Workplace—are documented in this book.

Index

E

F

G

H

I

J

L

M

O

P

R

S

T

U

V

W