



Community Experience Distilled

Alfresco 3

Web Content Management

Enterprise Web Content Management made easy and affordable



Munwar Shariff
Pallika Majmudar

Amita Bhandari
Vinita Choudhary

[PACKT] open source*
community experience distilled
PUBLISHING

Alfresco 3 Web Content Management

Enterprise Web Content Management made easy and affordable

Munwar Shariff

Amita Bhandari

Pallika Majmudar

Vinita Choudhary



BIRMINGHAM - MUMBAI

Alfresco 3 Web Content Management

Copyright © 2010 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the authors, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: September 2010

Production Reference: 1150910

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

ISBN 978-1-847198-00-6

www.packtpub.com

Cover Image by Vinayak Chittar (vinayak.chittar@gmail.com)

Credits

Authors

Munwar Shariff
Amita Bhandari
Pallika Majmudar
Vinita Choudhary

Reviewers

Johnny Gee
Sivasundaram Umapathy

Acquisition Editor

Steven Wilding

Development Editor

Dhiraj Chandiramani

Technical Editors

Chris Rodrigues
Hithesh Uchil

Copy Editor

Lakshmi Menon

Editorial Team Leader

Akshara Aware

Project Team Leader

Ashwin Shetty

Project Coordinator

Zainab Bagasrawala

Indexer

Hemangini Bari

Proofreader

Lynda Sliwoski

Graphics

Geetanjali Sawant

Production Coordinator

Arvindkumar Gupta

Cover Work

Arvindkumar Gupta

About the Authors

Munwar Shariff is the CTO and EVP of EMEA at CIGNEX. CIGNEX is the leading provider of open source Enterprise Content Management (ECM) solutions for businesses and government agencies.

He has served as the chief architect and manager of engineering teams for 19 years in the areas of business application software, Internet applications, social media, and mobile applications for customers worldwide. He is an expert in Content Management Systems (CMS). Since co-founding CIGNEX in late 2000, he has successfully delivered more than 130 CMS applications using various open source technologies. He has written a number of articles on open source CMS, is an experienced trainer, and a frequent speaker at conferences related to this topic.

Munwar earned his M.S. in Digital Electronics and Advanced Communications from National Institute of Technology, Surathkal, India. He has authored three technical books on open source CMS: *Alfresco 3 Enterprise Content Management*, *Implementing Alfresco*, and *Plonelive*.

Amita Bhandari is a Senior Consultant at CIGNEX. She has extensive experience in implementing Enterprise Web Applications using J2EE technologies. For the past four years, at CIGNEX, she has rolled out numerous Alfresco deployments worldwide in the areas of Document and Web Content Management. She has worked on performance tuning projects where Alfresco was deployed in clustered and load balanced environments. She has also implemented Single Sign On for easy communication between two systems.

She has worked with clients in media, gaming, healthcare, and e-governance. She has trained many students in advanced Java technologies.

She holds a Masters in Computer Applications from Rajasthan University, India. She has co-authored a book titled *Alfresco 3 Enterprise Content Management* and has also worked as a Technical Reviewer for the book titled *Alfresco 3 Web Services*.

Pallika Majmudar is a Senior Consultant at CIGNEX Technologies. She has strong hands-on experience in Java-based technologies, such as J2EE, Object-oriented Architecture and Design Patterns, Frameworks, web services, and web scripts.

She has architected and led many content management projects for customers in the USA, Hong Kong, Singapore, and India. She has implemented Alfresco-based solutions for clients across verticals like media, healthcare, hi-tech, and communications.

Pallika has earned a Masters degree in Computer Applications from Gujarat University, India. She has co-authored a book titled *Alfresco 3 Enterprise Content Management*.

Vinita Choudhary is a senior consultant at CIGNEX. She has extensive experience working in a variety of environments with cross-functional, multicultural teams.

She has reorganized existing repositories of documentation; written guidelines for document creation, filing, and change control; and written reference material for software developers and published it. She is involved in providing pre-sales support to the sales team and has worked on process streamlining for the company and various documentation aspects.

Vinita holds a Masters degree in Computer Applications from Gujarat University, India. She has co-authored a book titled *Alfresco 3 Enterprise Content Management*.

Acknowledgement

We thank our CEO Jeff Colvin, President Paul Anthony, President Americas Amit Babaria, and all the employees of CIGNEX for making this book a reality. We would like to thank Manish, head of India operations and Ram, head of delivery, for their encouragement and continuous support.

Our consulting team at CIGNEX helped us with real-life examples from various Alfresco implementations that we would not have imagined possible. We are thankful to them. We thank our sales, pre-sales, inside sales, and marketing teams for giving us an opportunity to provide Alfresco-based solutions to many customers.

We sincerely thank and appreciate David Barnes of Packt Publishing for giving us the opportunity. We thank Steven Wilding, Dhiraj Chandiramani, Zainab Bagasrawala, and the entire team at Packt Publishing. It is a pleasure working with them.

Our special thanks to our families and friends.

About the Reviewers

Johnny Gee is the Chief Technology Officer at Beach Street Consulting, Inc. In that role, he is responsible for architecting solutions for multiple clients across various industries and building Content Enabled Vertical Applications (CEVAs) on the Documentum platform. He has over 12 years of experience in ECM system design and implementation, with a proven record of successful ECM project implementations.

In addition to earning his undergraduate degree in Aerospace Engineering from the University of Maryland, Johnny achieved two graduate degrees: one in Aerospace Engineering from Georgia Institute of Technology and another in Information Systems Technology from George Washington University.

Johnny is an EMC Proven Professional Specialist in Application Development in Content Management and helped co-author the EMC Documentum Server Programming certification exam. He holds the position of top contributor to the EMC Support Forums and is one of the twenty EMC Community Experts worldwide. He has been invited on multiple occasions to the EMC Software Developer Conference and has spoken at EMC World. He also has a blog dedicated to designing Documentum solutions.

Johnny was the technical reviewer for Pawan Kumar's revision to *Documentum Content Management Foundations: EMC Proven Professional Certification Exam E20-120 Study Guide* that will be published later this year. He is also currently reviewing *Alfresco 3 Business Solutions*.

Sivasundaram Umapathy is currently working as a technical architect with Sella Servizi Bancari, the IT division of Gruppo Banca Sella, Italy, where he is leading the organization's transition to Alfresco and Liferay technologies. He has a Postgraduate Program in Software Enterprise Management (PGSEM) from IIM, Bangalore and an M.S. in Software Systems from BITS, Pilani. He has an array of certifications ranging from CGEIT, TOGAF 8, PMP, SCEA, OCA, SCBCD, SCWCD, also SCMAD to SCJP. He has co-authored *SCMAD Exam Guide* (ISBN 9780070077881) and been a technical reviewer of *Head First EJB* (ISBN 9780596005719). His current interests are Enterprise Architecture, IT Governance, IT-Business mismatch, and Entrepreneurship. He can be reached at siva@sivasundaram.com or via his LinkedIn profile at <http://bit.ly/sivasundaram>.

Table of Contents

Preface	1
Chapter 1: A Publishing Style Web CMS	7
Good web content management pays big dividends	7
High labor costs and the shortage of qualified personnel	8
How long it currently takes to implement site changes	8
Potential problems caused by erroneous or out-of-date postings	8
Revenue losses attributable to an inability to respond	9
Competitive issues related to a lack of planning	9
Various WCM systems in the market	9
The Alfresco WCM model	11
Web projects	11
Sandboxes	11
Virtualization and In-context Preview	12
Transparent layers	12
Web forms	13
Rendition templates	13
Web scripts	14
Workflows	14
Content delivery concepts	14
Static delivery model	15
Dynamic delivery model	15
Overview of delivery models	16
The best of both worlds	16
Significant enhancements in Alfresco WCM with Version 3.3	17
Alfresco Web Editor	18
Summary	19
Chapter 2: Installation and Configuration	21
Installing a JDK	22
Verifying the JAVA_HOME environment variable location	23

Table of Contents

Installing MySQL	23
Verifying the MySQL installation	23
Alfresco WCM	24
Installation option that is suitable for you	26
Enterprise and Community Editions	26
Operating Systems: Windows, Linux, Unix, MacOS	27
Databases: MySQL, Oracle, MS SQL Server, PostgreSQL	27
Application Servers: Tomcat, JBoss	28
Portals (optional): JBoss portal, Liferay	29
Choosing a suitable software for your installation	29
Eclipse installation	30
Installing Alfresco	40
Installing Alfresco on Windows	41
Installing Alfresco on Windows (full installation)	41
Installing Alfresco on Windows (excluding JDK)	45
Installing the Alfresco Tomcat bundle on Windows	47
Installing Alfresco on Red Hat Linux	48
Installing the Alfresco Tomcat bundle on Linux	50
Installing Alfresco on Mac	50
Installing the Alfresco WAR on any platform	52
Modifying the directory paths for Tomcat 6.x	52
Downloading the extension samples	53
Deploying Share into a separate Tomcat instance	53
Installing Alfresco components	54
Installing Alfresco WCM	54
Verifying the WCM installation	55
Installing the WCM standalone deployment receiver	56
Compiling and deploying the customizations on top of the WCM core	57
Installing OpenOffice	58
Installing ImageMagick	60
Installing Microsoft Office add-ins	61
Installing Flash Player	63
Installing SWFTools	63
Installing TinyMCE language packs	66
Installing an Alfresco Module Package	66
Installing Microsoft Office SharePoint Protocol Support	68
Installing the SharePoint Protocol Support AMP	69
Configuring SharePoint Protocol Support	70
Configuring SharePoint Protocol for Online Editing	71
Running Alfresco	71
Starting the Alfresco server	72
Starting Alfresco Share	72
Starting Alfresco Explorer	73
Stopping the Alfresco server	73
Starting the Alfresco virtualization server	73
Stopping the Alfresco virtualization server	74

Table of Contents

Starting the deployment engine	74
Stopping the deployment engine	75
Starting and stopping Alfresco as a console application	75
Installation folder structure	76
Configuring Alfresco as a Windows service	77
Summary	78
Chapter 3: Getting Started with Alfresco WCM	79
 Understanding the basics of WCM	79
Log in to Alfresco WCM web interface	80
My Alfresco Dashboard	80
Web project Sandboxes	82
User Sandbox interface	84
Advanced Versioning Manager (AVM)	85
 The web project	86
Create the web project	86
Creating a site easily with web project	90
Listing User Sandboxes	93
Add content to the web project	94
Submit content to the Staging Sandbox	95
 Filesystem projection	97
 Virtualization server	99
Configuring the virtual server for preview	99
Virtualization URL format	99
Virtualization server access to the User Sandbox	100
Virtual server configuration	101
 Dynamic websites using WCM	101
Virtual server JSP support	102
Previewing WARs and getRealPath()	102
Virtual server configuration	102
 Search	103
 Summary	103
Chapter 4: Web Content Production with Web Forms	105
 Why web forms	106
 Introduction to web forms	107
 Creating web forms	107
Identifying the structure to be used for each web form	108
Defining a schema	108
Defining a complex element	109
Defining a simple element	109
Default and fixed values for elements	110
Optional and required values for elements	110

Table of Contents

Advanced schema attributes	111
File pickers	111
Tool tips and labels	117
Create a web form in Alfresco	123
Rendition templates	127
Using FreeMarker templates for renditions	127
FreeMarker template engine within Alfresco	128
Alfresco objects available to FreeMarker	129
FreeMarker template-node model API	130
FreeMarker directives	130
Defining and creating FreeMarker templates	130
Extensible Stylesheet Language	132
Using XSLT for renditions	132
Using XSL-FO for renditions	134
Associating rendition templates to web forms in Alfresco	134
Associating web forms and renditions for specific/multiple project(s)	139
Creating dynamic content	141
Edit web forms for renditions	147
Associating a .xml file to the web form	151
Static and dynamic include of content	153
Web publishing dashlets	156
Summary	158
Chapter 5: WCM Workflows	159
Why workflows are required	160
Introduction to the workflow	161
Workflow process	162
Out-of-the-box workflow	163
Configuring workflows	163
Associating workflows to web forms	164
Associating workflows to web projects	165
Submitting content to the Staging box	168
Using the Edit Web Content wizard	168
Using Submit Items Wizard	170
Dynamically changing workflow for each snapshot submission	178
Creating a custom WCM Workflow for a group	179
Defining the workflow process	181
Expiring content in WCM	198
Configuration	198
Summary	201
Chapter 6: Dynamic Deployment and Customizations	203
Dynamic deployment	203
Dynamic models	204

Table of Contents

Deploying a model file	204
Dynamic Resource Bundles	206
Deploying a Resource Bundle	206
Dynamic workflows	208
Deploying a Process Definition file	208
Dynamic Alfresco Explorer	213
Deploying Alfresco Explorer customizations	214
Customization of existing workflow to use e-mail notifications	217
Remove workflow for specific staging submission	219
ZERO Workflow	222
Workflow Viewer	227
Summary	228
Chapter 7: Content Delivery and Deployment	229
Introduction to content delivery	230
Live server vs. Test server	231
Static vs. Dynamic delivery model	231
FSR for static delivery	233
Installing FSR	233
Configuring your deployment targets	235
Start and stop deployment receiver	235
Using FSR from Alfresco WCM staging	236
Configuring a web project to use FSR	236
Deploying a snapshot to FSR manually	237
Viewing deployment report and history	239
Reverting or rolling back to an older snapshot	241
Deploying to multiple servers	242
Advanced topics on FSR	242
Configuring prepare and postCommit callbacks	242
Defining payload transformations	243
Defining transport adapters	244
ASR for dynamic delivery	245
Configuring WCM deployment service	245
Number of send threads	246
Number of deployments in parallel	246
AVM Deployment Target	246
Auto deployment	246
Deploying to a test server	247
Setting up a test server pool	248
Deploy to a test server	249
Preview the content	250
Release the test server	250
Deploying from workflow	251

Table of Contents

Deploying from Alfresco WCM to DM repository	252
Setting up Alfresco DM as the deployment target	252
Deploying to DM	253
Summary	254
Chapter 8: Managing Multiple Websites Using WCM	255
Multiple web projects	256
Reusing forms, templates, and workflows	256
Using a web project as a template	258
Managing multiple websites using a single web project	259
Setting up multiple URLs on the target server	259
Setting up FSR for each target website	260
Creating many webapp folders	260
Layered folders	261
Creating a transparent folder	262
Updating a source file	264
Updating the destination file	266
Deleting files	266
Adding new files	267
Summary	267
Chapter 9: Alfresco Surf and Web Editor	269
Alfresco Surf platform	269
Applications using the Alfresco Surf platform	271
Alfresco Surf architecture	272
MVC pattern	272
Surf model objects	276
Surf API	276
Rendering engines	280
Design site navigation	280
Design a page	281
Use of a component in a page	283
Design page navigation	285
Communicating with Web Content Management	291
Using YUI (Yahoo User Interface) library	293
Alfresco Web Editor	295
Deploying and using Alfresco Web Editor	295
Deploying Web Editor to a Spring Surf Application	297
Alfresco Web Editor tag library	298
Sample Web Application using Alfresco Web Editor	299
Web Editor Framework	302
Core WEF Components	303

Table of Contents

Core WEF Widgets	303
Summary	303
Chapter 10: Integrating WCM Using Web Scripts	305
Concepts of WCM web scripts	306
Overview of REST architecture	306
What is REST	306
REST's main principles	306
Alfresco web scripts overview	307
What is a web script	307
Why to use web scripts	308
Alfresco web script framework	308
What's new in Alfresco 3 web scripts	309
Using web scripts with Alfresco WCM	311
Implementing web scripts for WCM	312
Components of web scripts	312
Description document	312
Controller script	312
One or more response templates	313
Configuration document	313
Locale message bundle	313
Creating a description document	313
Basic elements of description document	314
Advanced configuration for a description document	314
Response templates (URI templates)	316
Response type formats	317
Root objects of FreeMarker	318
FreeMarker methods for the AVM repository	319
AVM API	319
AVM store API	320
AVM node API	320
Response status	321
Web script controller	321
Objectives of a controller	322
JavaScript controller	322
Root objects for an execution script	322
JavaScript methods for the AVM repository	323
AVM API	324
AVM store API	324
AVM node API	324
Java-backed controller	325
How to declare a Java Bean	325
Creating a Java Bean class	325
Implementing web scripts	326

Table of Contents

Creating a web script	326
Storing the web script	326
Storing it on the filesystem	326
Storing it in Alfresco Explorer	326
Registering the web script	328
Listing the web scripts for external access	329
Integrating WCM with external applications—case studies	330
Integrating Alfresco WCM and Liferay with a news portlet	330
Web script for getting news headlines	330
Description document	330
Execution script	331
Response template	331
Storing/registering a web script in Alfresco	332
Portlet in Liferay	333
Integrating Alfresco WCM and Drupal with monthly blogs	333
Web script for getting monthly blogs	333
Description document	334
Execution script	334
Response template	335
Storing / registering the web script in Alfresco	337
Calling the web script in Drupal	337
Integrating Alfresco WCM with any J2EE web application	338
Web script for getting the details of a particular news item	338
Description document	338
Java-backed Bean for a web script	339
Response template	339
Calling web scripts from a JSP page	342
Enhancing the news item web script	344
Web script for getting the details of a particular news item	345
Description document	345
Java-backed Bean for web scripts	345
Response template	346
Storing/registering the web script in Alfresco	348
Calling the web script from a JSP page	348
Integrating Alfresco WCM and a Surf-based web application	349
Response template	349
Integrating web scripts with a SURF application	350
Summary	351
Chapter 11: Leveraging Alfresco Framework for WCM	353
Membership and Security Mechanism	354
Configuring LDAP for centralized identity management	354
LDAP configuration with Active Directory	355
LDAP synchronization	357

Table of Contents

Daisy Chaining	358
User roles	360
Common repository	361
Integrating Alfresco with the FFMPEG Video Transcoder	361
Various options for video transcoding	362
Various options for audio transcoding	362
Integrating transformation as an Action in Alfresco	363
Configuring FFMPEG transformation as a business rule	365
Copying videos from DM to WCM	367
DM to WCM using business rule	367
DM to WCM using JavaScript	370
Image transformation in WCM	372
Image transformation APIs	373
Configuring new action for image transformation in WCM	373
Using image transformation action in WCM	375
Advanced search in WCM	376
Using JavaScript	377
AVM API to search in WCM store	378
Using FreeMarker template	378
Using the Node browser	379
Using Java	379
Case study: User Interface for Advanced Search in WCM	380
Metadata extraction for WCM	384
Summary	388
Chapter 12: WCM Administration	389
Data backup	389
List of items to back up	390
Content stored in filesystem	390
Metadata stored in a relational database	391
Customization files	391
Membership data	392
Logfiles	392
Backup frequency	392
Backup is based on Alfresco deployment	393
Alfresco deployed as a Repository Application Server	394
Alfresco deployed as a hot backup	394
Upgrading to new versions of Alfresco	395
Upgrading to a minor release	396
Upgrading to a major release	396
Cleaning up deployment history	398
Using Alfresco Explorer	398

Table of Contents

Using scheduler	399
Deployment report 1 day before	400
Deployment report 1 day after	401
General maintenance tips	401
Examine logfiles	401
Reset the administrator password	402
Providing administrator rights	402
Reset complete repository data	403
Migrating servers	403
Summary	404
Index	405

Preface

There are many web content management systems available in the market today, either proprietary or open source. They help you to design a website, create web pages, link all media assets, edit the pages inline, and manage the look and feel of using templates. However, most of them use a single system for authoring and delivery, do not provide a validation process, do not secure the websites in more detail, cannot reuse the content effectively, and cannot integrate with external system to share the web content.

Alfresco provides a robust, easy to use, and scalable web content framework for managing multiple websites leveraging a common web infrastructure. Alfresco provides a publishing style web CMS, where you can create, preview, and approve the content on staging instance and then deploy the websites to test and/or production environments. It allows the separation of content from the look and feel and thus provides multiple usage of the same content by many other applications.

Alfresco offers true Web Content Management (WCM) by providing an open source alternative to expensive proprietary systems such as Microsoft SharePoint, Interwoven, and IBM Content Manager. Alfresco WCM is a good fit for the customers who are also looking for cost savings.

This book will guide you through creating, managing, and publishing web content in staging, test, and production environments. It will help you set up an infrastructure for supporting multiple websites using Alfresco, enabling a shortened web development cycle and providing a high return on investment, despite a low cost of ownership.

This book takes a step-by-step approach for building a complete web content management system using Alfresco. A greater emphasis is given to the concepts of web content creation and distribution. Plenty of sample code and screenshots are used in the book to make you confident in applying these concepts in real production scenarios.

Your feedback is very valuable to us. You can contribute by reporting any errors you find in the book, making suggestions for new content that you'd like to see in future updates, commenting, and blogging about it.

What this book covers

Written in an easy-to-read and encouraged-to-try style, this book will take you from the basics of publishing style CMS—such as web forms, page templates, and staging—to the skills that will make you an Alfresco developer, covering advanced topics such as workflow, web services integration, and more.

The topics that this book covers are:

Chapter 1, A Publishing Style Web CMS, explains the Alfresco's Web Content Management architecture and key features of the software.

Chapter 2, Installation and Configuration, includes tips to choose the right installation for you, and also installation of the software and start using it.

Chapter 3, Getting Started with Alfresco WCM, introduces the basic concepts of Alfresco Web Content Management from a user perspective. It demonstrates how to set up and configure Alfresco Explorer for managing a web project and showcases a sample web publishing scenario.

Chapter 4, Web Content Production with Web Forms, includes advanced concepts of separating the web content from presentation. You do this by storing the web content in XML using Alfresco web forms and presenting its various formats such as HTML and text using Presentation Templates.

Chapter 5, WCM Workflows, explains the entire process of creating web content, getting it approved, and publishing it to a staging environment. You will learn and have extensive hands-on experience with the examples to create a flexible workflow.

Chapter 6, Dynamic Deployment and Customizations, explains the dynamic customization of workflow without requiring a restart of the Alfresco server.

Chapter 7, Content Delivery and Deployment, introduces you to the content delivery and deployment features of Alfresco. You will understand the concepts behind delivering static content as well as dynamic content to the external production servers.

Chapter 8, Managing Multiple Websites Using WCM, covers information about managing multiple web projects using one installation of Alfresco WCM.

Chapter 9, Alfresco Surf and Web Editor, introduces Surf, which is an application framework for developing and delivering dynamic websites. Alfresco web editor is an application developed using Surf and provides in-context editing capabilities for Alfresco repository content.

Chapter 10, Integrating WCM Using Web Scripts, teaches you the web scripts and the integration of Alfresco WCM with the external systems using web scripts.

Chapter 11, Leveraging Alfresco Framework for WCM, provides information to leverage Alfresco's Document Management features in WCM.

Chapter 12, WCM Administration, provides a high-level overview of administering and maintaining your Alfresco implementation.

What you need for this book

The default installation of Alfresco requires installing software downloaded from the SourceForge project location (http://sourceforge.net/project/showfiles.php?group_id=143373). Now Alfresco is hosting its own enterprise download area. You can also download this from <http://www.alfresco.com/products/ecm/enttrial/>. Select the download package. And you will be asked for the user name and password of the Alfresco content community.

To install and run Alfresco, you need at least 1 GB disk space and at least 1 GB RAM on the desktop or server.

Who this book is for

If you are a web developer or content manager and wish to build a website using Alfresco, and customize it as per your client's needs, then this book is for you. It will also help business users to migrate from the existing proprietary web development tools to standards based open source web content management. Although no knowledge of Alfresco is presumed, exposure to HTML, XML, JavaScript, Java, and related web technologies will help you to get the most from this book.

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.

Code words in text are shown as follows: "We can include other contexts through the use of the `include` directive."

A block of code is set as follows:

```
<cm:person view:childName="cm:person">
<cm:userName>fredb</cm:userName>
<cm:firstName>Fred</cm:firstName>
<cm:lastName>Bloggs</cm:lastName>
<cm:email>fredb@alfresco.org</cm:email>
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
</property>
<property name="stores">
    <list>
        <value>workspace://SpacesStore</value>
    </list>
</property>
<property name="queryTemplate">
    <value>PATH: "/app:company_home"</value>
</property>
<property name="cronExpression">
    <value>0 0/15 * * * ?</value>
</property>
<property name="jobName">
    <value>jobD</value>
</property>
<property name="jobGroup">
    <value>jobGroup</value>
</property>
```

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

```
> chmod a+x ./alfresco-<version>-linux-community.bin
```

New terms and important words are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "Go to a space and add a file by clicking on the **Add Content** line."



Warnings or important notes appear in a box like this.



Tips and tricks appear 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 send an e-mail to feedback@packtpub.com, and mention the book title via 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 e-mail 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 this book

You can download the example code files for all Packt books you have purchased from your account at <http://www.PacktPub.com>. If you purchased this book elsewhere, you can visit <http://www.PacktPub.com/support> and register to have the files e-mailed directly to you.

Errata

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

Piracy

Piracy of copyrighted material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at copyright@packtpub.com with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

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

1

A Publishing Style Web CMS

The Alfresco Web Content Management is a next generation tool that allows organizations to rapidly create and more effectively maintain dynamic Internet, intranet, and extranet sites, enabling a shortened web development cycle, providing high returns on investment, and low cost of ownership. WCM manages the content and structure of websites, including the framework and navigation, as well as the creation, editing, approval, and publication processes. By using Alfresco you can implement web content management solutions with a scalable content repository: Web 2.0 AJAX-based user interface, flexible workflow, multi-language support, and a robust search engine.

This chapter provides an introduction to Alfresco WCM, outlining the benefits of using it for your enterprise's web content management requirements. It also introduces the new features of the Alfresco WCM.

In this chapter, you will learn about:

- Alfresco WCM architecture
- The Alfresco WCM model
- Features of Alfresco WCM
- Benefits

Good web content management pays big dividends

When an organization's management team is reviewing budgetary proposals, the price tag associated with a new computing solution often elicits one question: "Can't we avoid this expense?" The answer is obvious: you can continue to operate your websites without the benefit of a content management solution. However, the real question should be, "What is the cost of not making this purchase?"

Here are some factors to consider:

High labor costs and the shortage of qualified personnel

Finding and holding on to qualified employees can be a challenge. All the more a reason to make the most of each individual's skill set. A web content management solution that empowers content providers to post directly to a live site increases their productivity. By the same token, IT personnel who are not ensnared in the posting process are free to apply their skills and knowledge to more challenging tasks. Providing state-of-the-art tools helps promote positive attitudes while improving productivity, which can go a long way in retaining skilled personnel.

How long it currently takes to implement site changes

Does it take hours or even days to post new content on your websites? That's often far too long to leave outdated or incorrect information on a site. After all, the beauty of the Internet is providing target audiences with 24x7 access to all of the latest and greatest information. When a site becomes stale, users become disenchanted and are less likely to return. Recapturing a user's interest is far more difficult than maintaining it with fresh, personalized content.

Potential problems caused by erroneous or out-of-date postings

When new information doesn't get posted quickly, what does it cost your organization? In case of a publicly-traded company, incorrect financial postings can have serious consequences. Providing only accurate, timely product information can prevent misunderstandings that lead to customer dissatisfaction. Giving distributors and suppliers incorrect, out-of-date, or partial information can have a negative effect on your bottom line. The right web content management solution, one that is easy to use and maintain, will help ensure that your organization provides site visitors with reliable content.

Revenue losses attributable to an inability to respond

A website that cannot be scaled to meet a business' emerging needs is just as serious a problem as an inability to hire more people, move to a larger facility, or acquire additional suppliers. Revenue can be lost and the future of the organization diminished. Why take such chances, especially when websites are becoming central to doing business?

Competitive issues related to a lack of planning

Hiring an experienced webmaster allows an organization to use that individual's skills beyond day-to-day site maintenance. A knowledgeable individual can help review site architecture, assess future site requirements, and implement upgrades. These are valuable activities for growing organizations that want to remain competitive in today's rapidly changing economic environment. However, when the webmaster must operate as a "web page processor", the time and skills of this valuable resource cannot be fully utilized.

The right web content management solution can allow your organization to:

- Make effective use of all internal resources
- Slash the time required to implement site content changes or redesign a site
- Ensure the availability of timely, accurate information
- Scale its website to keep pace with organizational growth
- Plan to accommodate new business initiatives and technological advances

Various WCM systems in the market

The worldwide Web Content Management market has been growing exponentially. The market maturity has homogenized much of the competition. Therefore, procurement decisions should be increasingly based upon vendor viability and the vendor's long-term product strategy. Maintaining your web assets is both a cost of doing business and a competitive differentiator. There are proprietary and open source WCMs available today for organizations to choose from. Alfresco is one of the leading choices when organizations look at the open source options available to them.

The Alfresco WCM engages customers through next-generation sites, enabling mass contributions from internal and external users, simple configuration via reusable web scripts, and low-cost massive scalability that uses commodity software and hardware.

Cutting edge technology, rich interface experiences, user participation, and effective costing are all factors that organizations seek to consider while selecting the best suited WCM solution for their organization.

From a high-level perspective, WCM solutions in the market today can be classified into two types:

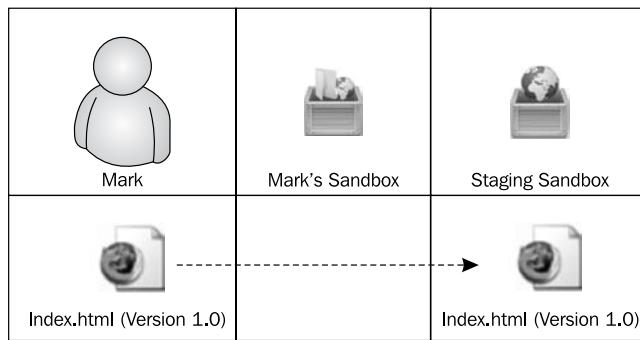
- Wiki style
- Publishing style

The following table shows the differences between the two types:

Features	Wiki Style Web CMS	Publishing Style Web CMS
Authoring and delivery	Single system for authoring and delivery	Authoring and delivery are separated
Content and presentation	Little separation of content and presentation Content = Page or page fragment	Separate content and presentation Content ≠ Page or page fragment
Validation	Little or no validation / QA process	Configurable editorial and approval process
Editing	In place editing of live web pages	Editing of separate editorial copy of content
Apt for	Smaller sites or those managed by smaller teams	Larger sites or those managed by larger teams
Examples	Wikis Joomla! Drupal PHP Nuke Portal Server CM portlets	Alfresco Interwoven Vignette Day

The Alfresco WCM model

Content Production and Content Delivery are separated in Alfresco Web Content Management, as shown in the following diagram:



It is important to understand the concepts that form the basis of the Alfresco WCM model.

Web projects

Web projects are the production-side representation of a site. This is what manages the content consumed by the site. Here the access rules and roles for content producers are defined. Every Alfresco server can have multiple web projects. Within a web project the user can:

- View content based on the state of their User and/or Staging Sandboxes
- Preview content based on the state of a sandbox with workflows
- Upload file-based content
- Create web forms and manage content
- Submit content to staging and deploy content to a live environment

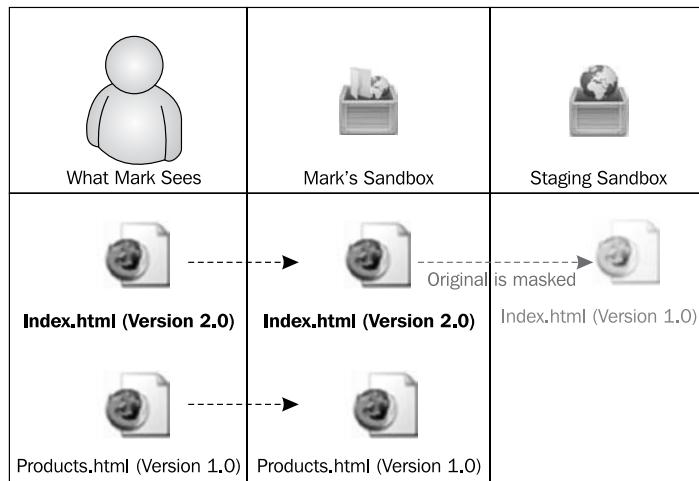
These actions can be controlled and managed through the use of workflows.

Sandboxes

Alfresco provides a sandboxed development model. Content producers make use of the sandboxes to make changes to a site in isolation from one another. The default configuration is as follows:

- One Staging Sandbox per web project
- One User Sandbox per user per web project

- One temporary workflow sandbox per active workflow instance per web project:



Virtualization and In-context Preview

Virtualization and In-context Preview is core to the sandboxing concept. Virtualization means that each user has a complete view of all current, approved, checked in content along with those unique modifications made within the context of their sandbox.

Alfresco provides a complete virtual view of the website as it would look if all changes in a sandbox were committed to the live site even when previewing any non-modified or modified asset in a sandbox. This is In-context Preview.

Each user in the context of their sandbox can do rigorous and thorough quality checks for all changes they are posting to the website.

Transparent layers

Transparent layers are the means to implement sandboxes in Alfresco. This layer is a central construct in the **Advanced Versioning Manager (AVM)** repository, very similar to the UnionFS Linux filesystem, and is used to define "composite" stores that can "read through" content from other stores. It can be defined at the store, directory, or file level.

From Alfresco 3.1 onwards, transparent layers can be configured by a Content Manager in the Staging Sandbox of a web project. This is useful for:

- Defining web project templates
- Reusing content across multiple web projects
- Explicitly segregating different groups of content producers for separate web projects

Web forms

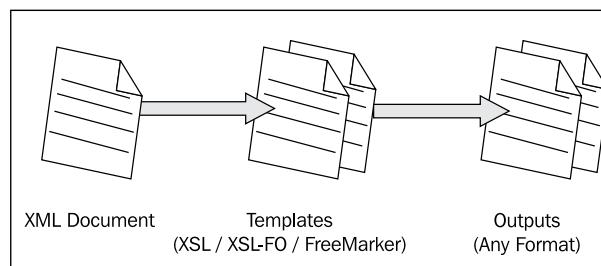
Web forms are used in Alfresco WCM to capture content from the user, and store as XML. An XML schema needs to be created by form developers for capturing content. It is then rendered automatically as a user-friendly web-based form for content contributors.

Alfresco uses the open source project **Chiba**, an XForms implementation used to transform the XML schema into an internal representation of a form (XForms), and then present UI controls for elements and attributes described in the schema. This helps to render the form entry UI to the end users.

Web forms are created and administered in the **Web Forms** space within the **Data Dictionary**. As they are located in Alfresco Spaces, they are accessible by the default CIFS, FTP, and WebDav interfaces. They can also be configured with rendering engine templates for generating renditions of the collected content.

Rendition templates

The web form-managed XML can be transformed with rendition templates and the corresponding content into rendered output. Server-side templating languages, such as FreeMarker, XSLT, and XSLT-FO are provided by Alfresco. After a content item (XML file) is created via a web form, each rendition template configured for that content type is executed, producing an output file per template (shown in the following diagram). Typical formats for renditions of web content include HTML, JSP, PDF, XML, and so on:



Web scripts

Web scripts provide RESTful access to content held within your Alfresco Enterprise Content Repository. You can therefore place controls on your enterprise content to manage it, and provide uniform access for a wide variety of client applications and services, such as browser, portal, search engine, or any custom application.

Web scripts allow you to:

- Easily access, manage, and cross-link your content via a customized RESTful API. You do not need any compilation, generators, server restarts, complex installs, tooling, or Java knowledge. All you need is your favorite text editor or the Alfresco Explorer web client.
- Build custom URI-identified and HTTP-accessible Content Management Web Services.
- Turn your Alfresco repository into a content management-powered HTTP server.

Workflows

Alfresco WCM uses JBoss jBPM for all workflows. There are three aspects of workflows in Alfresco:

- **Workflow definition:** The creation and deployment of the jBPM workflow into Alfresco repository.
- **Workflow association:** The assignment of a workflow to a web project, which specify the actors (reviewers identified).
- **Workflow instance:** Created when content that is specific to the associated change set is submitted. Additionally, Alfresco comes with Web Site Submission workflow out of the box, which allows for serial and parallel approval of content.

Content delivery concepts

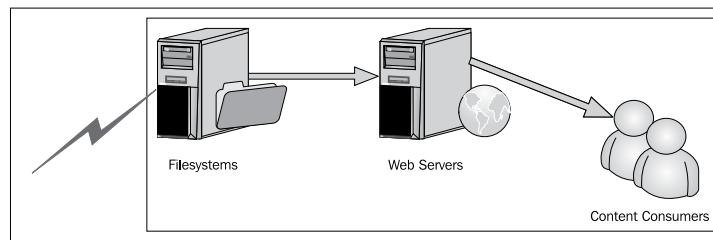
In Alfresco there are three delivery models: static, dynamic, and a hybrid of both static and dynamic. In a static delivery model, all requests to the web server return a static file of XHTML, XML, JSON, and so on to the web client without any additional processing (no CGIs, no SSI, and so on).

In a dynamic delivery model, all requests to the web server return objects of type XHTML, XML, JSON, and so on that are processed by some application server to render the resulting document.

Static delivery model

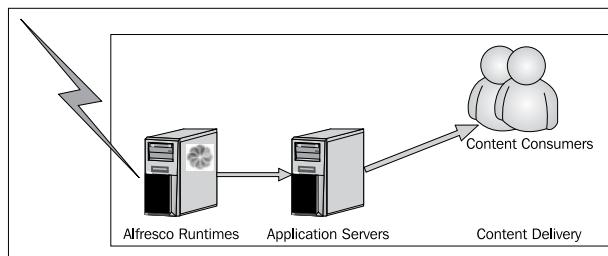
In such a model, pages are rendered as part of the content production process. The resulting HTML and associated assets (images, CSS, JS, and so on) are then published to the filesystem, typically a document root of a web server. This provides high levels of scalability on simplified production architectures (web server farms). This model, however, has limited personalization and there is a set number of rendering technologies (FreeMarker, XSLT, and XSLT-FO).

A **File System Receiver (FSR)** will need to be installed and configured to receive published static content from the Alfresco server. The FSR consists of a small server that receives updates from an Alfresco repository and publishes them to a flat filesystem, which is then typically served up by a web or application server. The following diagram illustrates this process:



Dynamic delivery model

A pure dynamic model publishes content to an Alfresco Runtime, thereby making the content available for dynamic queries with basically any web technology (PHP, Python, J2EE, AJAX, Flash, Cold Fusion, and so on). This provides ultimate flexibility in what and how content is displayed on a page. This provides the highest levels of personalization, but will require significantly more resources on the delivery servers for similar levels of traffic. For all but the smallest websites, significant effort is required in architecting, developing, and testing to ensure website or application stability. This is particularly the case during unexpected high-volume situations (for example, a Government website during a national disaster). The following diagram illustrates the dynamic delivery model:



An **Alfresco System Receiver (ASR)** will need to be installed on a server to facilitate the dynamic delivery model. The ASR is just another instance of the Alfresco server. The ASR allows a web project being authored in one Alfresco server instance to be deployed to another separate instance of Alfresco.

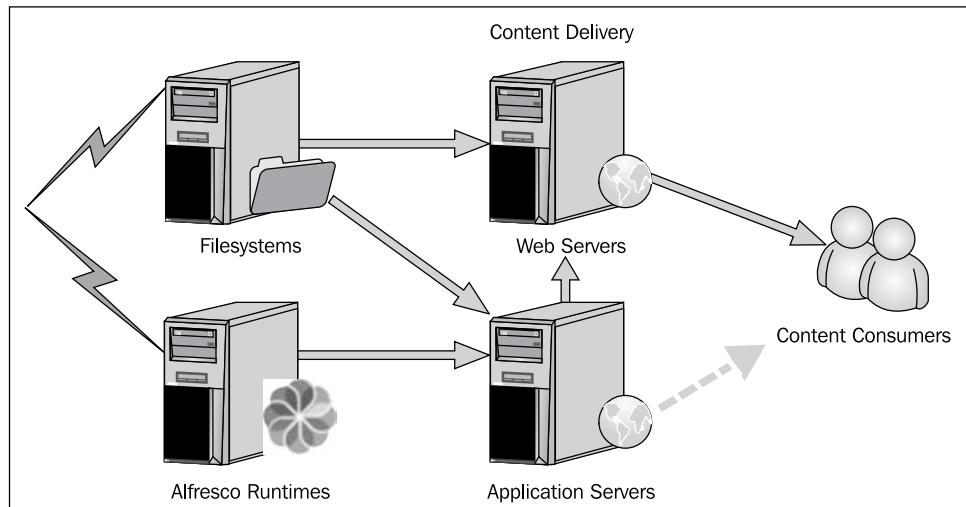
Overview of delivery models

The following is a summary of static and dynamic delivery models:

	Static "Bake" Model	Dynamic "Fry" Model
Delivery technology	Web servers	Application servers
Page compositing	Submission time	Request time
Content deployed to	Filesystem	Alfresco runtime
Personalization	Limited	Unlimited
Performance	Ultimate	Less than the "bake" model
Application developer skill sets	FreeMarker, XSLT, XSLT-FO	Any web technology

The best of both worlds

A hybrid approach is the preferred approach regardless of the WCMS and the underlying technologies. Determination of what is static and what is dynamic is highly dependent on the type of website and web applications.



Users also have the option of a hybrid delivery approach. This approach can be executed as follows:

- The web architecture model should be designed to support the dynamic model. This includes the ability to deploy content to both filesystems and Alfresco runtimes for flexibility.
- Leverage the static model wherever possible. If content must be personalized to a single user or a very small set of users with few "page" impressions, it most likely needs to be dynamic. Otherwise, it can be static.
- Choose a page composition model appropriate to the overall site and each page on the site:
 - **Outside-in:** Each page is static HTML with static components already embedded, but dynamic components or applications such as AJAX and Flash can be included.
 - **Inside-out:** Each page is dynamic and includes all page components dynamically regardless of whether those components are static or dynamic.

Significant enhancements in Alfresco WCM with Version 3.3

A bunch of new features focused on helping companies manage their web presence have been introduced in Version 3.3. A list of these is as follows:

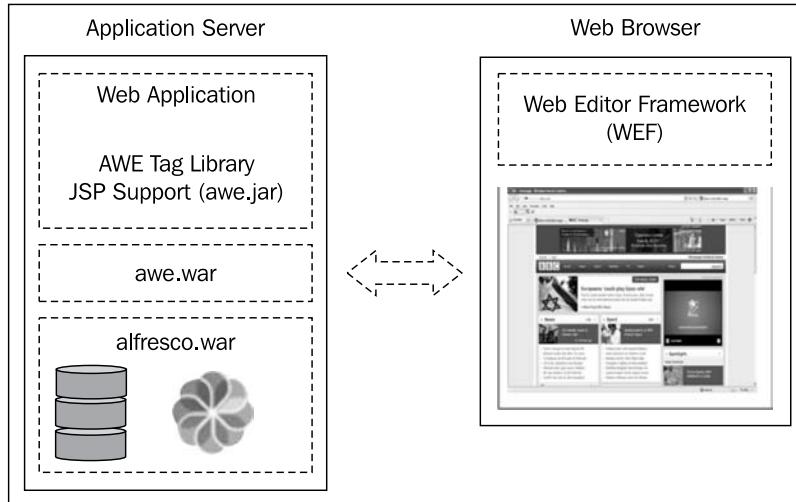
- **Alfresco Web Editor:** In-context editing to Alfresco (non-AVM) stored content has been introduced. This will allow content authors to edit content items stored within an Alfresco repository directly from the web page. Alfresco 3.3 also provides the Web Editor Framework, a JavaScript client-side framework, rendering a toolbar, and associated controls.
- **Transfer Service API:** Developers can build solutions that transfer content between Alfresco repositories (non-AVM) using the Transfer Service API. This is useful to WCM architectures where Alfresco provides both authoring and delivery tier components and allows rich-content structures and relationships to be maintained between Alfresco environments.
- **Rendition API:** The Rendition API will allow developers to build solutions for easily repurposing content for the Web. FreeMarker and XSLT templates can also be used as part of the Rendition API.

- **WCM deployment:** The Alfresco Deployment Receiver is configured as sub-system and a new Data Dictionary folder called Web Deployed is configured to default as the deployment target. AVM to DM-WCM deployment facilities have been enhanced to add an additional deployment target. This additional deployment receiver allows WCM content that is authored and stored within the AVM to be deployed to local and remote Alfresco repositories (Alfresco DM).

Alfresco Web Editor

The **Alfresco Web Editor (AWE)** is a Spring Surf-based web application that utilizes the Forms Service to provide in-context editing capabilities to Alfresco repository content (non-AVM). Alfresco 3.3 also introduces the **Web Editor Framework (WEF)**, which is a client-side JavaScript framework that is a dependency of the AWE.

With the initial release, the AWE will support **JavaServer Pages (JSP)**-based websites by providing a tag library. Additional languages will be supported in future releases with FreeMarker and PHP being on top of the list. The tags have been designed for easy implementation so that a developer can enable the AWE with minimal effort, and without effecting the CSS layout and design of the site.



The simplest and quickest way to deploy AWE is to use the prebuilt WAR (`awe.war`) file and deploy it in the same application server instance of your web application. Being a Spring Surf-based application, AWE does not have to be deployed in the same application server instance as the Alfresco repository. However, this section presumes that it is.

Summary

An easily navigated site, with information consistently organized in a logical fashion, is what most organizations want to provide. But delivering consistent organization with proper adherence to corporate branding and design standards can be difficult when several authors are contributing content. If more than one designer or Webmaster posts content, standards can easily become compromised and consistency diminished.

In this chapter, we have learned that Alfresco gives you a web content management solution that:

- Has content component architecture where content is separated from format; it is easier to reuse.
- Uses an open, object-based API—an open interface providing compatibility with new or emerging technologies.
- Is an open source alternative.

The next chapter focuses on installing Alfresco and various components around it. Installation on various operating environments is detailed therein. Also explained is the installation of various components like OpenOffice, ImageMagick, Microsoft Office Add-ins, Flash Player, and SWFTools.

2

Installation and Configuration

This chapter is aimed at enabling you to understand and carry out various aspects of the Alfresco WCM installation and configuring various components around it. A basic understanding of Alfresco architecture, various installation options, and the key terminologies used are all a part of this chapter. The Alfresco development environment needs to be set in place before we can work with the tools.

The installation procedures can also vary depending on the kind of system you have in place to set up and configure Alfresco. First, we will go through the installation procedures for a Microsoft Windows-based system. Linux-based systems also follow similar steps. We will go into detail later in the chapter.

At the end of this chapter, you will have learned the following:

- Components to install
- Configuring the Alfresco Setup
- The WCM Component
- Setting up the Alfresco WCM development environment

Following is the list of softwares you need on your machine before you install Alfresco:

Component	Recommendation
Java SE Development Kit (JDK)	JDK 6 is required.
Database	Alfresco comes preconfigured with the MySQL database. If you intend to use a different database, install and configure the database before you install Alfresco.

Component	Recommendation
OpenOffice.org	Alfresco uses OpenOffice for transforming documents from one format to another, for example, a text file to a PDF file. If you do not install OpenOffice, you will not have access to the transformation functionality.
Flash Player Version 10.x	Alfresco Share requires Flash Player Version 10.x to upload multiple files and view Flash previews. If you do not install Flash, you see the upload screen for single files.
SWFTools	Alfresco Share uses the pdf2swf utility for previewing PDF files. If you do not install SWFTools, you will not see PDF previews, but image previews will still be available.

Alfresco runs within an instance of the Tomcat application server. The installers and the Tomcat bundles are preconfigured with Tomcat. If you wish to install Alfresco within another application server, use the Alfresco WAR file.

Installing a JDK

A Java SE Development Kit (JDK) must be installed on your system before you install Alfresco. Some Alfresco installation wizards will detect whether you have a JDK on your machine and, if not, install a version for you. This task explains how to install JDK manually:

1. Browse to the Sun Microsystems Java download website:
<http://java.sun.com>.
2. Select and download Java Development Kit (JDK) 6 for your platform.
3. If prompted, specify a location in which to download.
4. Navigate to where you downloaded the JDK.
5. Install the JDK on your system.

After the JDK is installed on your system, verify that the `JAVA_HOME` environment variable is set.

Verifying the JAVA_HOME environment variable location

The `JAVA_HOME` environment variable location must be set to where the JDK is installed. This is done as follows:

1. Open the command prompt.
2. Enter the following:

Windows: `echo %JAVA_HOME%`

Linux: `echo $JAVA_HOME`

Installing MySQL

This section describes how to set up a MySQL open source Relational Database Management System (RDBMS) to use with Alfresco. Some of the Alfresco installation wizards install an embedded instance of MySQL that is configured with the correct settings. If you prefer to install MySQL database independently, this section describes the configuration settings that you should use:

1. Browse to the MySQL download site: <http://dev.mysql.com/downloads>.
2. Locate and select the appropriate package for your platform. Alfresco requires MySQL 5.0.67 or higher.
3. If prompted, specify a location on your system in which to download and install MySQL.
4. Browse to the location where you downloaded MySQL and double-click on the installer file.
5. The MySQL Server Setup wizard guides you through the MySQL installation, followed by the Configuration wizard.

Verifying the MySQL installation

Once you have installed MySQL, this task describes how to verify that it was installed correctly:

1. Open the command prompt.
2. In the prompt, enter:

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

3. Type the password that you set during the installation and press *Enter*. Information about the installed MySQL version displays. If no errors are reported, MySQL is installed and running.
4. In the `mysql>` prompt, type `exit` to exit MySQL. You have verified that the MySQL installation was successful.

Alfresco WCM

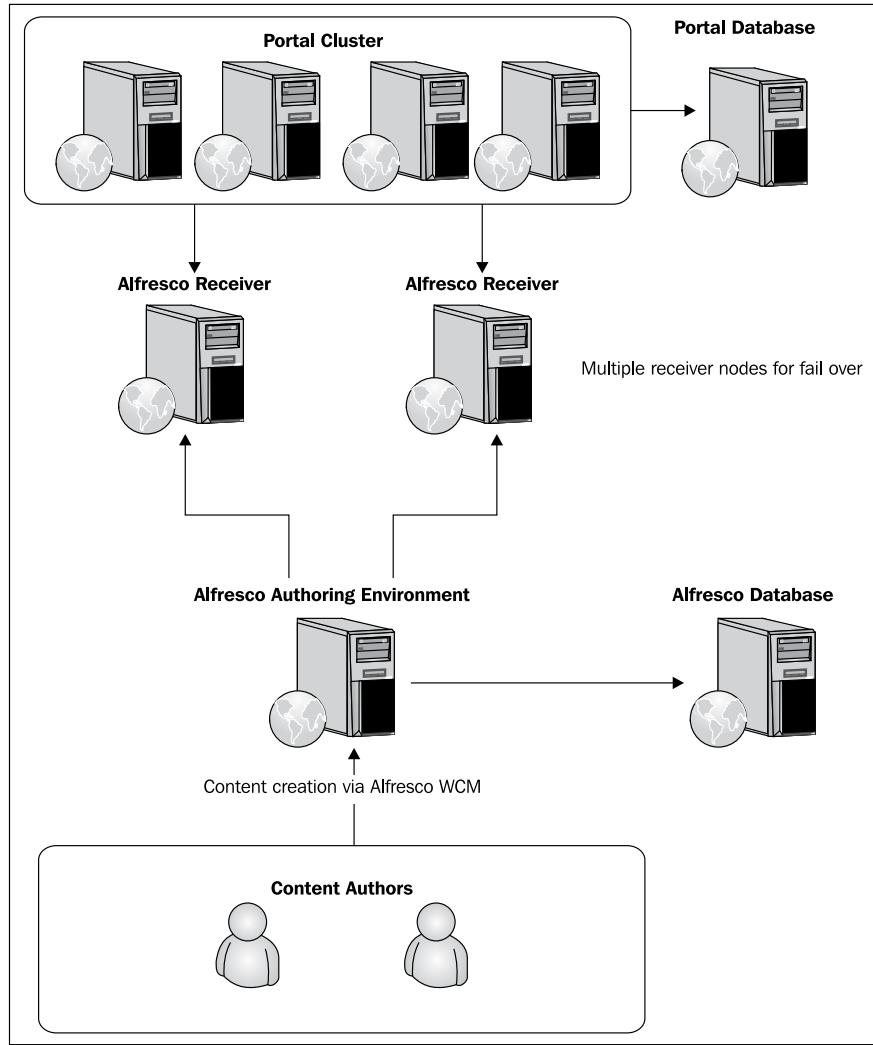
Alfresco is built on state-of-the-art open source components such as Spring, Hibernate, Lucene, and JSF – often the components of choice for website developers today. It offers one repository for the whole team. This repository is a modern platform for Web Content Management 2.0 with a variety of user and technical benefits as described and discussed through the course of this book.

The web development framework comprises:

Java	JSF, Struts, Wicket, among others
Ruby	Ruby on Rails
.NET	ASP.NET
PHP	Zend, CakePHP, CodeIgniter
Python	Django

Alfresco can be used by all of the previously mentioned frameworks. Each of them has different capabilities and often requires different implementation strategies. Using Alfresco WCM, you can provide a wealth of flexibility and choices when designing scalable web architecture.

The following diagram shows a possible physical architecture for a WCM solution:



WCM can be recommended to be used when:

- The authors are comfortable using the Alfresco Web Client and web forms
- Minimal time should be spent creating a robust presentation tier
- You need the ability to roll back change sets
- You only need to link to WCM content and basic URLs
- You need to have a staging environment where content is reviewed before it is published
- Simple "submit for approval" workflow is sufficient

Installation option that is suitable for you

Alfresco is a 100 percent open source software developed using open standards. It can be operated on various operating systems, relational databases, application servers, web browsers, portals, and supports various languages. We shall examine all of the choices and determine which option is right for you.

Enterprise and Community Editions

Alfresco currently provides two types of product download options:

- Alfresco Community Lab Network
- Alfresco Enterprise Network

The latest release of Alfresco has an Enterprise Edition as well as a Labs edition. Both the options have the same code base and features, and are 100 percent open source. For both the options, you can use Alfresco documentation (wiki), community support (forums), and community-contributed add-on products. The Alfresco Community Lab Network is free. The Alfresco Enterprise Network includes a per CPU license fee.

The Alfresco Community Labs Network is an unsupported product and is mainly designed for developers and technical enthusiasts. It is designed in non-critical environments. Alfresco Community Lab releases Early and Often and renders a daily build, offering the latest functionality. It doesn't provide scalability and high-availability certifications. I would recommend this to be used as the research vehicle for new features and as the platform for the Alfresco Community. And consider it as a test drive before you install.

If you are implementing Alfresco for a major corporation, financial, insurance, government, or healthcare organization, I would recommend you go for Alfresco Enterprise Network support. The primary benefit is that with the support of Alfresco and its certified partners you will get a stable, reliable, certified, and supported application with warranty and indemnity. Your Alfresco version will be certified on all available stacks such as Linux, Windows, MySQL, Oracle, and many more. You will benefit from Alfresco support, which includes problem resolution, compatibility advice, migration advice, and upgrade support. For mission-critical applications, you will get 24/7 support from Alfresco experts.

Operating Systems: Windows, Linux, Unix, MacOS

Choosing an operating system to run Alfresco will be based on various factors. For some companies it depends on in-house expertise. For example, if you have administrators and IT staff who can easily manage business applications running on a Microsoft Windows platform, then your choice could be to go with a Windows operating system. For some companies it is based on the integration requirements with the existing systems.

If you do not have any preferences, I would recommend you to go with the Linux operating system for production use. Linux source code is freely distributed. Tens of thousands of programmers have reviewed the source code to improve performance, eliminate bugs, and strengthen security. No other operating system has ever undergone this level of review. The key advantages of Linux are listed below:

- The best technical support available
- No vendor lock-in
- Runs on a wide range of hardware
- Exceptionally stable
- Supports many tools and applications you need
- Interoperates with many other types of computer systems
- Low total cost of ownership

Databases: MySQL, Oracle, MS SQL Server, PostgreSQL

The data access layer of Alfresco is implemented using an open source software component and **ORM (Object Relational Mapping)** tool called **Hibernate**.

Hibernate abstracts the database layer and provides seamless integration between the Alfresco repository and any relational database.

If the Microsoft Windows operating system is the OS of choice, then the natural choice for you can be MS SQL Server. If you already have an Oracle license, then Oracle database is also an option. Alfresco also supports the PostgreSQL database.

If you do not have any database preference, it is recommended that you go with the MySQL database, which costs nothing if you go with the open source version. The MySQL database has become the world's most popular open source database because of its consistency, fast performance, high reliability, and ease of usability. It's used in more than 10 million installations ranging from large corporations to specialized embedded applications. MySQL runs on more than 20 platforms, including Linux, Windows, OS/X, HP-UX, AIX, and Netware, giving you the kind of flexibility that puts you in control.

Application Servers: Tomcat, JBoss

Alfresco runs on any J2SE 5.0-compliant application server. Hence there are no application server-specific dependencies. However, it is important to make a choice of application server before moving into production.

Alfresco uses the **Spring** framework and not the **Enterprise Java Beans (EJB)** framework. So there is no dependency on JBoss or any other application server, which provides EJB container. **Spring** is an open source application framework for Java/JEE. The Alfresco repository uses the Spring Framework as the core foundation of its architecture. If you are developing a standalone application, then Tomcat might be a good option. Apache Tomcat powers numerous large-scale and mission-critical web applications across a diverse range of industries and organizations. It is the most widely-accepted web application server in the market.

On the other hand you must consider using the JBoss application server, which has the highest market capture (> 35 percent) in J2EE-based application servers in the world. JBoss internally uses Tomcat and hence you get the benefits of Tomcat servlet engine as well.

Alfresco utilizes JBoss cache's ability to distribute and maintain data caches, making it possible to build large-scale systems that outperform traditional enterprise content management systems. Alfresco also utilizes the clustering, failover, and load balancing facilities of the JBoss application server to increase scalability. Alfresco's business process management features are powered by the JBoss jBPM tool.

If you have already invested in JBoss, then Alfresco provides complementary industry-leading enterprise content management technology to the JBoss enterprise middleware system suite.

Portals (optional): JBoss portal, Liferay

It is optional for you to go with a portal of your choice; if you already have an enterprise portal, you can integrate Alfresco with it. If you do not have a portal in place and you would like to leverage the portal framework, then you can consider using either JBoss portal or Liferay portal. Both of these are based on J2EE technology; both of them are open source and open standards-based and both of them have Alfresco built-in support.

JBoss Portal provides an open source platform for hosting and serving a portal's web interface, publishing and managing its content, and customizing its experience. Although most packaged portal frameworks help enterprises launch portals more quickly, only JBoss Portal delivers the benefits of a zero-cost, open source license combined with a flexible and scalable underlying platform.

Liferay is the most downloaded and popular open source portal with 40,000 downloads per month. It runs on top of any J2EE servlet such as Tomcat. So a full installation of JBoss is not required, but it can run against most full application servers out of the box, including JBoss, JRun, BEA, WebLogic, and Orion. It has a full set of web service interfaces to the portal. Liferay supports more than 800 portlets (products) and has a wider adoption in the market.

Choosing a suitable software for your installation

You need to make the best choice of software to install Alfresco. If you do not have any specific requirements, you might consider a complete open source stack for production usage and go with **Alfresco Enterprise Edition** on the Linux operating system with a MySQL database running on a JBoss application server with Liferay Portal.

The examples in this book were created and tested with the following choice of Alfresco installation:

- Eclipse
- Alfresco Community Edition
- Operating System: Windows XP
- Database: MySQL 5
- Application server: Tomcat 6
- JDK 5
- Portal: None

Eclipse installation

As we will be using Eclipse as a development tool for this book, in this section we will talk about the installation of Eclipse. Download the Eclipse Europa for Java/J2EE developers from:

<http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/europa/winter/eclipse-java-europa-winter-win32.zip>.

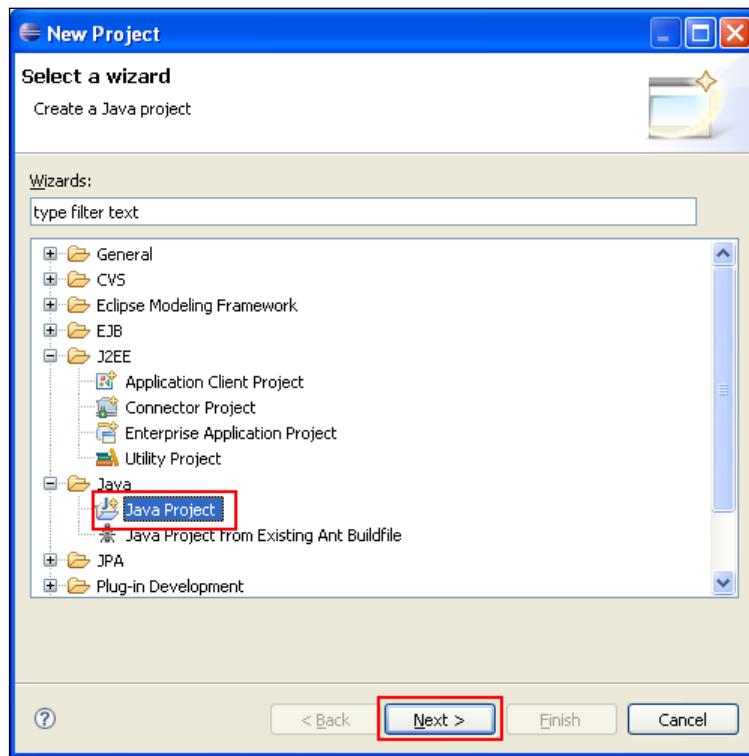
Configuring the project for development

In this section, we will learn to set up the development environment for any Alfresco WCM project. We will see how we can configure a project in Eclipse.

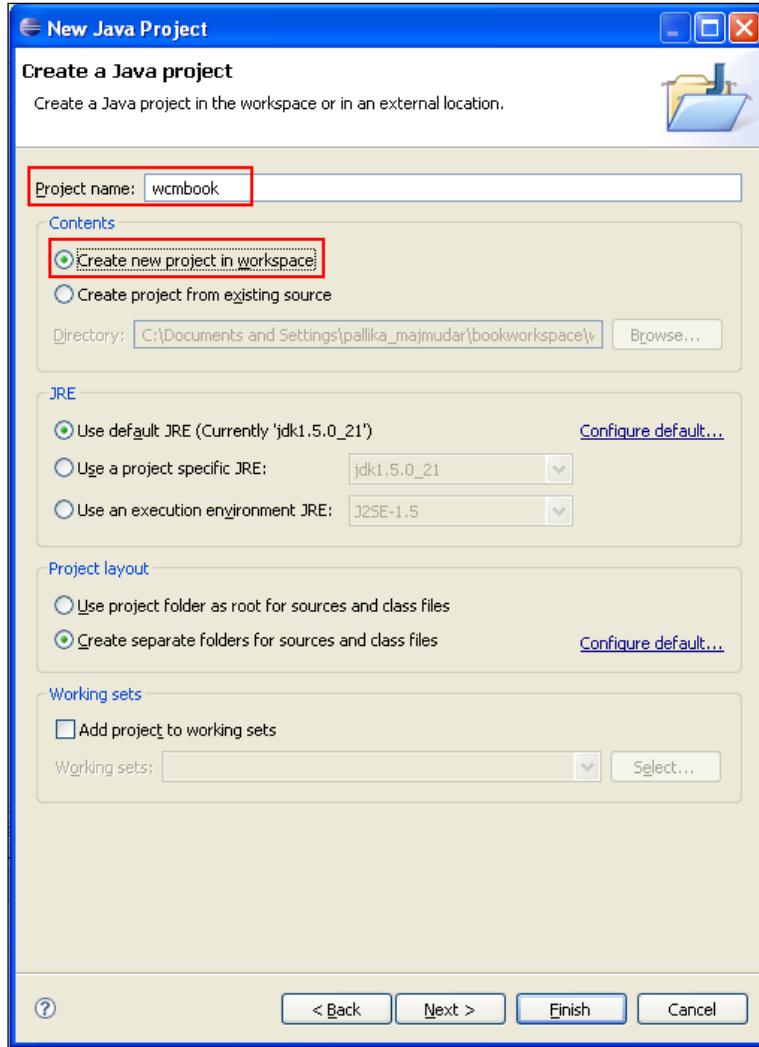
Open Eclipse by double-clicking on the Eclipse icon. It will ask you to select the workspace. Create a new folder called **bookworkspace** and browse to that folder when asked for the workspace.

Eclipse will now start. Now we need to set up a new project for the book in Eclipse. Go to **File | New | Project**.

1. Select **Java | Java Project** in the **New Project** wizard. Click on **Next**.

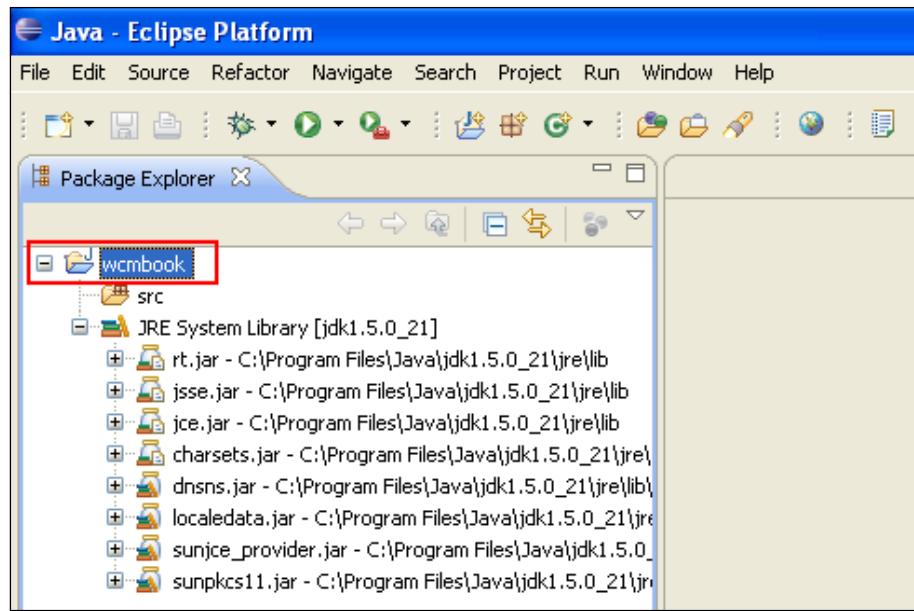


2. In the next screen, provide the **Project name** as **wcmbook** and choose the **Create new project in workspace** option. Click on **Next**.



3. Make sure you have **wcmbook/bin** in the **Default Output folder** in the next screen and then click on **Finish**.

4. The project named **wcmbook** will be created with a **src** folder and the JRE library with the required .jar files attached for this project (if JDK is already installed on your machine). You can check that in the **Package Explorer** window as shown in the following screenshot:



Configuring the build path

Now you need to configure the build path so that when you create any Java files, it can automatically be compiled by Eclipse. For that, make sure the **Project | Build Automatically** option is checked. To configure the build path for the project, follow these steps:

1. Right-click on project name, that is, **wcmbook** and select **Build Path | Add Libraries**.
2. Select **User Library** and click on **Next**.
3. Click on the **User Libraries** button. In the **User Libraries** screen, click on the **New** button.
4. Provide the name of the library as **alfresco_lib** and click on **OK**.
5. Select that library name and click on **Add JARs**. Select all the .jar files from the Alfresco installed folder, `tomcat/lib`, and add those to this library. Also select all the .jar files from the `tomcat/webapps/alfresco/WEB-INF/lib` folder and add those to this library. Once done, click on **OK**.
6. Then click on **Finish**.

Source code tree

Create the following folder structure for the project to store different types of files:

- src
 - <Java classes with package>
- config
 - alfresco
 - extension
 - <configuration files>
 - messages
 - <resource bundle files>
- web
 - jsp
 - css
 - scripts
 - images
 - ...
- /src: Put all your source Java classes in this folder with respective package structure.
- /config/alfresco/extension: Place all your extension configuration files in this folder.
- /config/alfresco/messages: Put Resource Bundle files (properties file) in this folder.
- /web/jsp: Place any custom or modified JSPs in this folder. It's a best practice to create an extension folder inside the jsp folder and put all the custom JSPs in that folder.
- /web/css: Place any custom or modified CSS files in this folder. It's a best practice to create an extension folder inside the css folder and put all the custom CSS files in that folder.
- /web/images: Place any images that you have modified or newly created in this folder.
- /web/scripts: Place any custom JavaScript files used by the user interface in this folder.

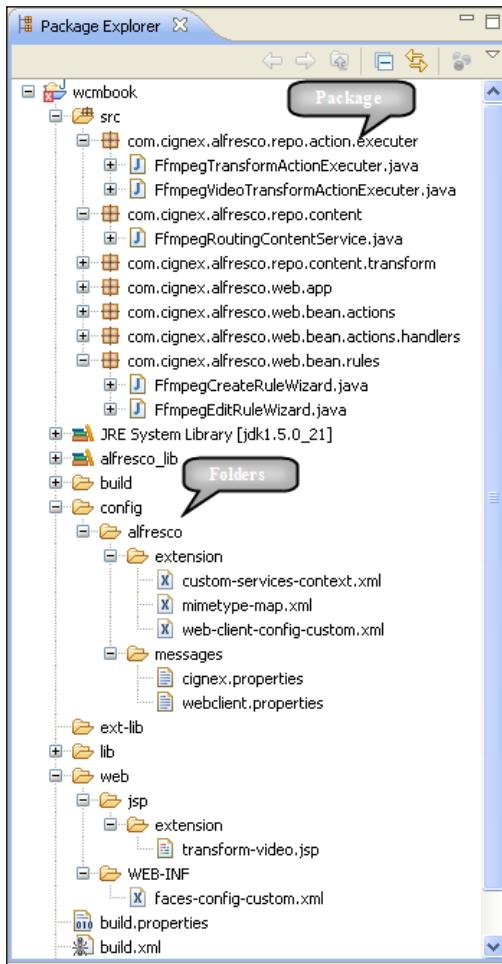
To create a folder in the Eclipse project:

1. Select the web project for root-level folder or select the parent-level folder and right-click, then click on **New | Folder**.
2. Provide the folder name and click on **Finish**.

To create a package in the eclipse project:

1. Select the source folder **src**, right-click, select **New | Package**.
2. In **Name**, provide the name of package, that is, `com.Cignex.web.bean`, and so on.

Once you are done with the package and folder structure creation, it will look similar to the following screenshot:



Build process

We will use Apache Ant, which is a Java-based build tool, for building the application from Eclipse.

Why to use a build tool

There are a number of steps required to transform the source into a deployable software solution. In general, a build tool allows developers and build managers to describe the build process. In a build, some steps may need to be executed before others or they may be independent of others. In summary, a build tool provides a mechanism to describe the operations and structure of the build process to deploy the code.

Integrating Ant with Eclipse

As we are using Eclipse as our development environment, we will learn how we can integrate Ant with Eclipse for the build process. First we need to create a new build file in Eclipse.

Creating the build.xml file:

1. Right-click on project name **wcmbook** in **Package Explorer**
2. Select **New | File**
3. In the **File Name** box, type **build.xml**
4. Click on **Finish**

Enter the targets for Ant Build in this file and save it. You can have build targets for compiling the code, packaging the .jar file, and so on. The build file we will use for our examples throughout the book will look like:

```
<project name="Cignex AMP Build File" default="package-jar"
  basedir="."> >

  <property file="build.properties" />

  <target name="clean" description="Cleans Build directory">
    <delete dir="${build.dir}" />
    <delete dir="${project.dir}/lib" />
  </target>

  <target name="clean-alfresco" description="Cleans alfresco
    directory">
    <delete dir="${alfresco.work.home}/tomcat/webapps/alfresco" />
  </target>
```

```
<target name="mkdirs" description="Creates Build directory">
  <mkdir dir="${build.dir}/dist" />
  <mkdir dir="${project.dir}/lib" />
</target>

<path id="class.path">
  <fileset dir="${alfresco.work.home}/tomcat/lib" includes="*.jar"/>
  <dirset dir="${alfresco.work.home}/tomcat/webapps/alfresco/
    WEB-INF/classes" />
  <fileset dir="${alfresco.work.home}/tomcat/webapps/alfresco/
    WEB-INF/lib" includes="*.jar" excludes="cignex.jar" />
</path>

<target name="compile" depends="clean,mkdirs" description="cmopiles
  the java source code">
  <mkdir dir="${build.dir}/classes" />
  <javac classpathref="class.path" srcdir="${project.dir}/src"
    destdir="${build.dir}/classes" debug="true"/>
</target>

<target name="package-jar" depends="compile" description="Package
  the classes into jar file">
  <jar destfile="${project.dir}/lib/${package.file}">
    <fileset dir="${build.dir}/classes" includes="**/*.class" />
  </jar>
</target>

<target name="package-amp" depends="package-jar"
  description="Package the Module">
  <zip destfile="${amp.file}" update="true">
    <fileset dir="${project.dir}" includes="web/**/*.*" />
    <fileset dir="${project.dir}" includes="config/**/*.*" />
    <fileset dir="${project.dir}" includes="lib**/*.jar" />
    <fileset dir="${project.dir}" includes="ext-lib**/*.jar" />
    <fileset dir="${project.dir}" includes="module.properties" />
    <fileset dir="${project.dir}" includes="file-mapping.properties" />
  </zip>
</target>

<target name="update-war" depends="package-amp,clean-alfresco"
  description="Update the WAR file.">
  <echo>Installing Cignex AMP into WAR</echo>
```

```

<java dir=". " fork="true" classname="org.alfresco.repo.module.
    tool.ModuleManagementTool">
    <classpath refid="class.path" />
    <arg line="install ${amp.file} ${alfresco.war.file} -verbose" />
</java>
</target>

<target name="package-extension" description="Package the extension
    files into zip file" depends="package-jar">
    <zip destfile="${package.file.zip}">
        <zipfileset file="${project.dir}/lib/*.jar"
            prefix="WEB-INF/lib" />
        <zipfileset dir="${web.dir}" />
        <zipfileset dir="${alfresco.dir}"
            prefix="WEB-INF/classes/alfresco"/>
    </zip>
</target>

<target name="war-backup" description="Package the extension files
    into zip file" depends="package-extension">
    <copy file="${alfresco.war.file}"
        tofile="${alfresco.war.backup}"/>
</target>

<target name="integrate-extension" description="Integrate
    the extension files to existing alfresco.war file"
    depends="war-backup,clean-alfresco">
    <available file="${alfresco.war.file}" type="file"
        property="alfresco.war.present" />
    <fail unless="alfresco.war.present"
        message="Could not find alfresco.war, please provide the
        correct path" />
    <zip destfile="${alfresco.war.file}" update="true">
        <zipfileset file="${package.file.jar}"
            prefix="WEB-INF/lib" />
        <zipfileset dir="${web.dir}" />
        <zipfileset dir="${alfresco.dir}"
            prefix="WEB-INF/classes/alfresco"/>
    </zip>
</target>

</project>

```

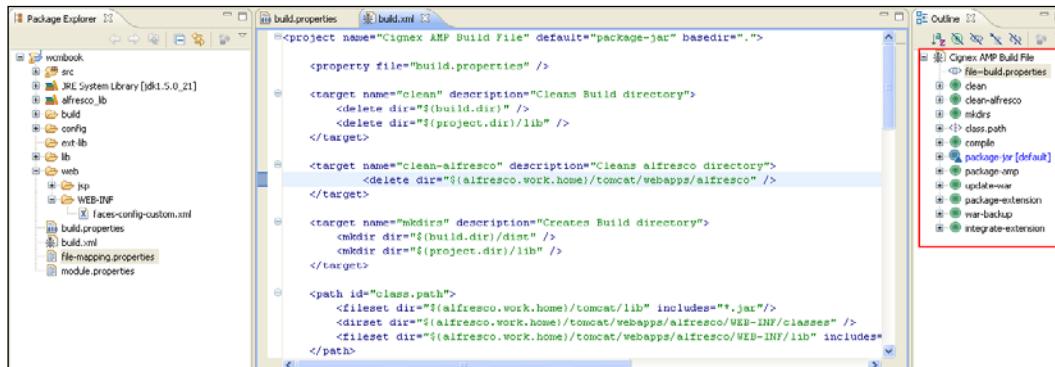
Here in this build file, we have used some of the properties that we need to define in the build.properties file.

Creating the build.properties file:

```
# Property file for ant Build
project.dir=.
build.dir=${project.dir}/build
package.file=cignex.jar
package.file.zip=${build.dir}/dist/cignex.zip
amp.file=${build.dir}/dist/cignex-install.amp
web.dir=${project.dir}/web
alfresco.dir=${project.dir}/config/alfresco
alfresco.work.home=C:/alfresco_3.2r
alfresco.war.file=${alfresco.work.home}/tomcat/webapps/alfresco.war
alfresco.war.backup=${alfresco.war.file}.backup_
```

In this file, you can change the values for the properties accordingly. The build.xml file will read the properties from this file.

When you create the build.xml file, you can see all of the targets for the build file in the **Outline** view at the right-hand side in Eclipse.



Running the Ant target:

There are three different ways to run a particular build target.

1st Approach:

1. Right-click on build.xml in the **Package Explorer**.

2. Select **Run | Ant Build**. This will run the default target for the Ant Build and gives you the results in Eclipse's Console view.

2nd Approach:

1. Right-click on `build.xml` in the **Package Explorer**.
2. Select **Run | Ant Build...** with ellipsis (three dots).
3. From the Ant launch configuration dialog screen, choose the target that you want to run from the available list of all targets of the build file and click on **Run**.

3rd Approach:

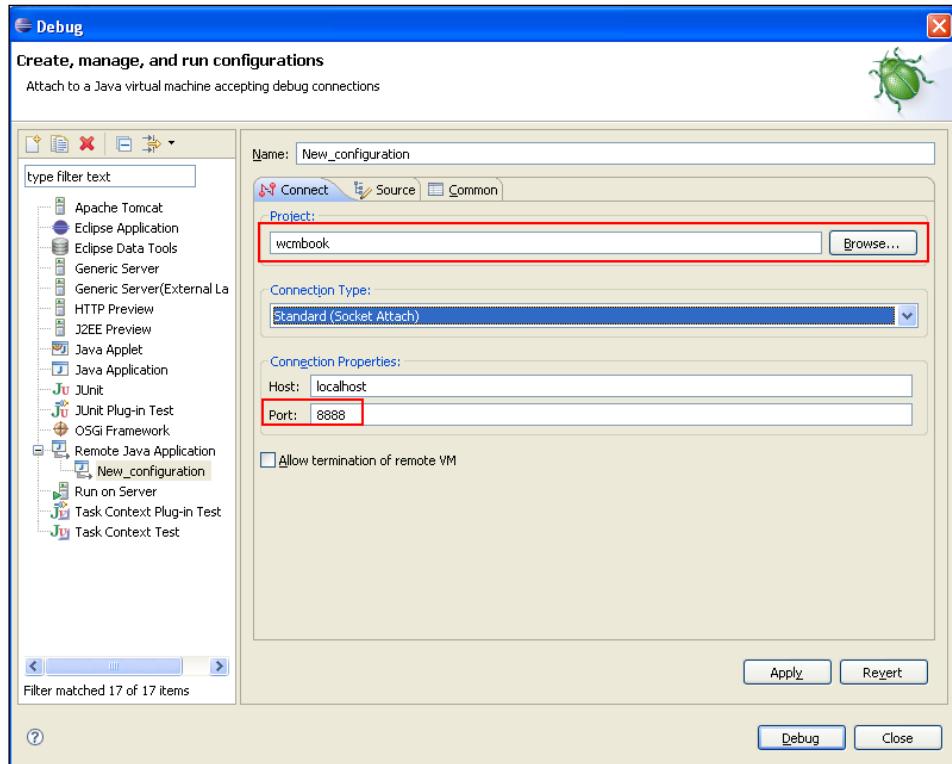
1. Go to **Windows | Show View | Ant** (if Ant is not available there, click on **Other...** and select **Ant | Ant** from there).
2. You will see the Ant view at the right-hand side in Eclipse.
3. Drag the `build.xml` file from **Package Explorer** to this Ant view.
4. Double-click on the target you want to run from the Ant view.

Debugging the Alfresco application in an Eclipse environment

We can configure Eclipse and start Alfresco in the debugging mode. This will help us in debugging the code for the Alfresco application. The steps to configure this are as follows:

1. Modify `catalina.bat` available at `<alfresco_home>\tomcat\bin` to add the following line for `DEBUG_OPTS=`:
`set DEBUG_OPTS=-Xdebug -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=8888`
2. Now in Eclipse, navigate to **Windows | Open perspective | Other | Debug**, which will open the debug perspective.
3. Click on **Run | Open Debug Dialog | Remote Java Application | New configuration**.

4. Browse the project and enter the port number similar to the debug port in the catalina.bat (8888) file as shown in the following screenshot:



5. Start the Alfresco server.
6. Put the breakpoint in the file, which you want to debug or where you want the control to stop.
7. Start the debug that you have just configured.
8. Use Alfresco (Web-Client) to get to the debug mode.

Installing Alfresco

This chapter provides information for installing Alfresco and its components. Depending on your system, you can install Alfresco using a number of different methods. For example, you can install Alfresco by:

- Using an installation wizard, which contains the required software and components you need

- Using a bundle that includes a preconfigured Tomcat server, the Alfresco Web Archive (WAR), batch files, database setup scripts, and a sample extensions folder
- Using a standard WAR file to deploy on your existing application server

A typical manual installation scenario includes the following procedure:

1. Install a Java SE Development Kit (JDK).
2. Install a supported database.
3. Install Alfresco.
4. Configure an Alfresco database.
5. Install Alfresco components.
6. Run Alfresco.

Installing Alfresco on Windows

This section describes how to install Alfresco using the following methods:

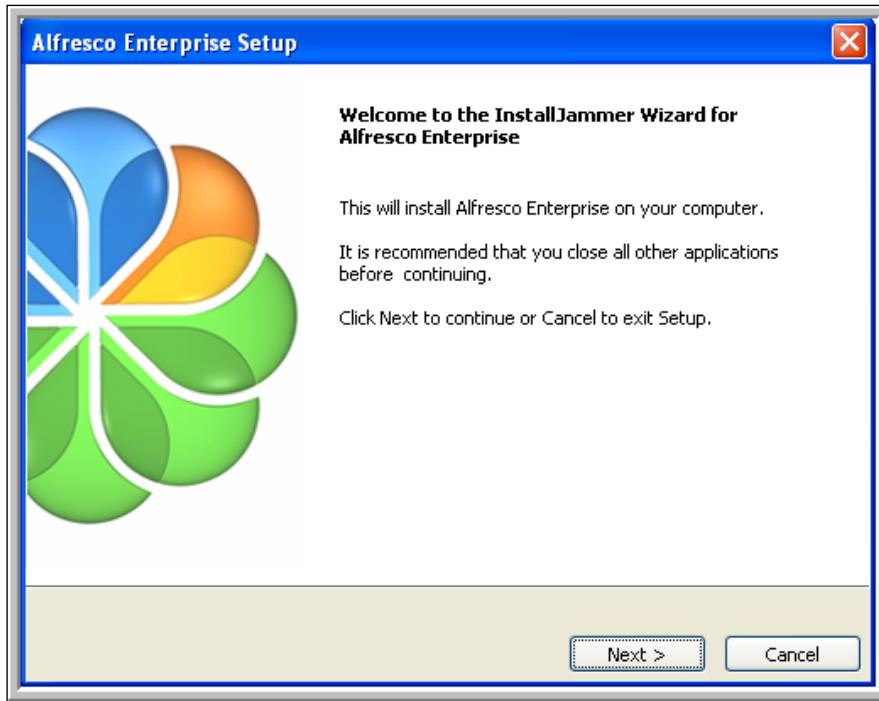
- Complete installation
- Installation excluding JDK
- Tomcat bundle installation

Installing Alfresco on Windows (full installation)

The installation wizard for Microsoft Windows installs all of the software and components that you require for running Alfresco.

1. Browse to the Alfresco Community Edition downloads area and download the following file: `Alfresco-Community-3.2-Full-Setup.exe`.
2. Double-click on the downloaded file. You may see an **Open File - Security Warning** message, prompting you to verify that you wish to run this software. To run the installation wizard, click on **Run**.
3. At the **Language Selection** prompt, select **English** and click on **OK**.
4. When prompted to confirm that you want to install Alfresco on your computer, click on **Yes**.
5. The installation wizard launches.

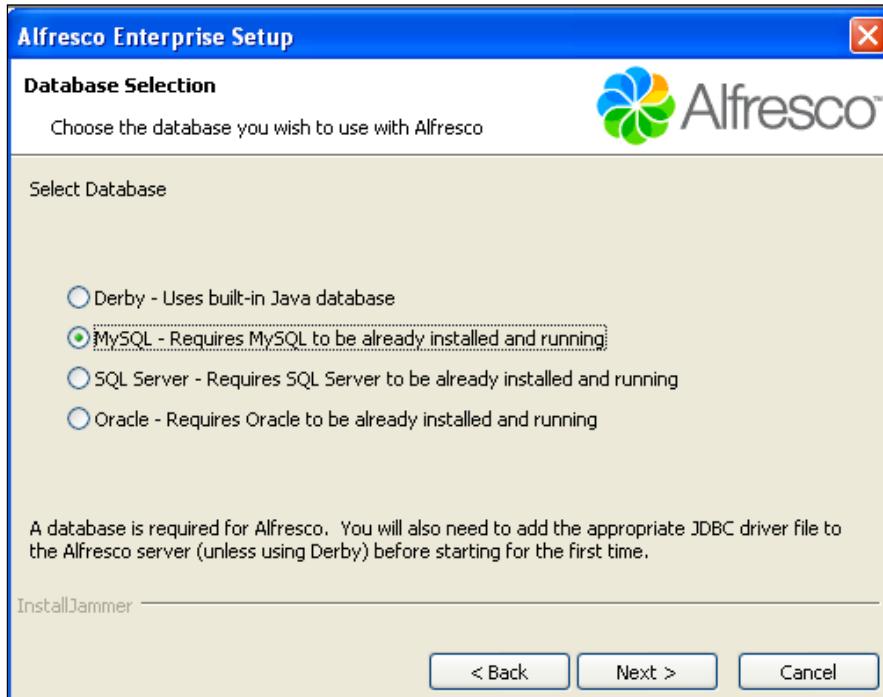
6. In the **Welcome to the InstallJammer Wizard for Alfresco Enterprise** window, click on **Next**.



7. In the **Setup Type** window, select one of the following options:

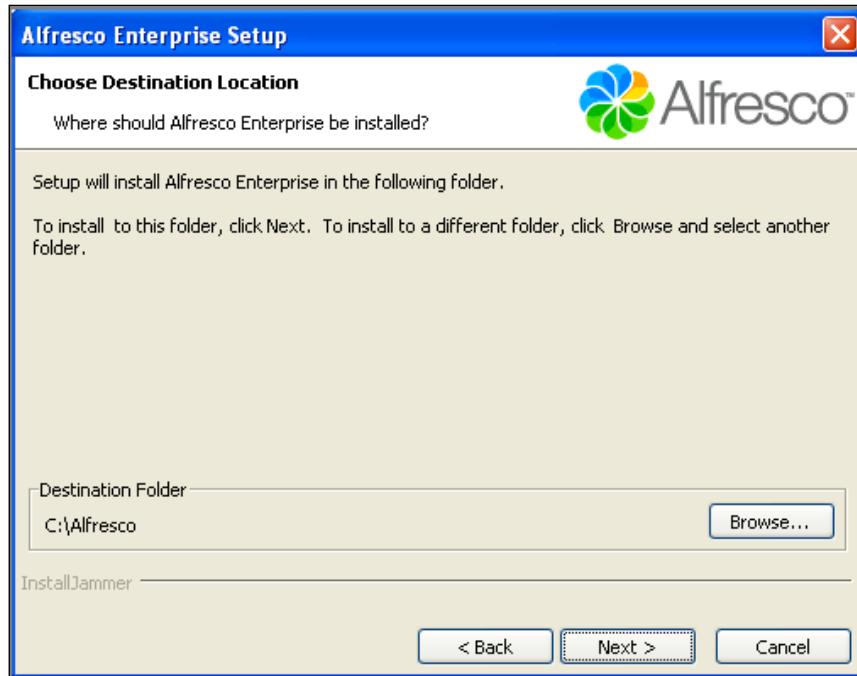
Option	Description
Typical	To select all of the components within the Alfresco full installation
Custom	To select individual components: <ul style="list-style-type: none">• Default Component (includes DM)• WCM• Java• OpenOffice• SharePoint Protocol• MySQL database

By default, Alfresco installs an embedded MySQL database instance, which is installed locally in the Alfresco directory and runs on port number 3306. (Optional) Select the database to use. You can choose one of the following:



8. If you wish to use an existing MySQL server rather than the embedded MySQL database, select the **Custom** option and deselect the **MySQL Database** option.
9. If you wish to install the embedded MySQL database in addition to your existing MySQL database, select the **MySQL Database** option. You will be prompted to specify an alternative port number after the installation starts.
10. If you wish to use a production database other than MySQL, deselect the **MySQL Database** option. Click on **Next**.

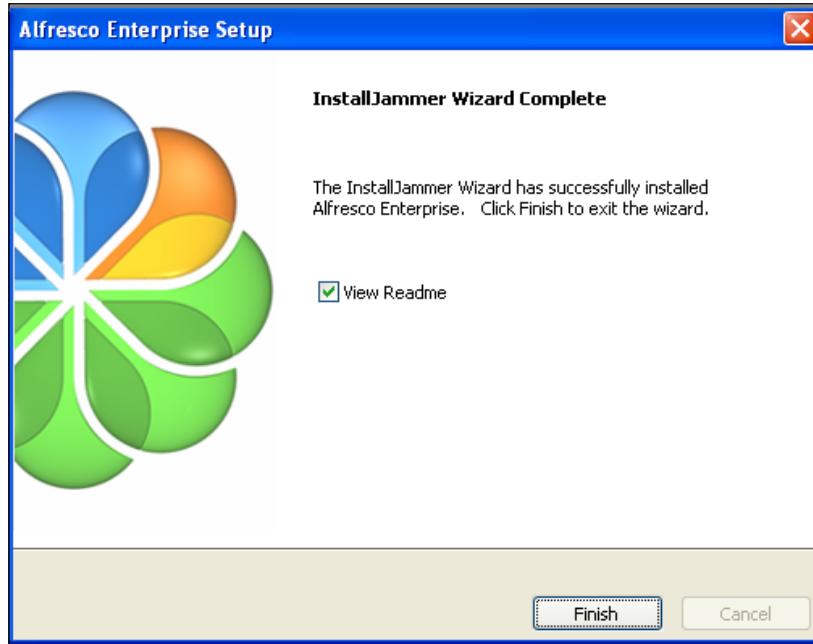
11. In the **Choose Destination Location** window, click on **Next** to accept the default location for Alfresco or click on **Browse** to choose another location.



12. In the **Start Copying Files** window, review the settings and click on **Next**.
13. The installation starts. If you chose the **Typical** option, a prompt notifies you when OpenOffice is being installed, and if you have an existing MySQL database (optional), the **Database Configuration** window notifies you that the default database port 3306 is already used by the embedded MySQL database and prompts you to enter the root password of your chosen database and an alternative database port. Then click on **Next** (optional).
14. The installation wizard writes the server name and host to the configuration file called `alfrescoglobal.properties`.

 If you are using your existing MySQL database, you need to modify and run the example scripts in the `extras` directory to create the Alfresco database and user for this instance of MySQL.

15. In the **InstallJammer Wizard Complete** window, click on **Finish**.



The installation is complete. The README file opens. When you close the README file, you are directed to the release notes.

Installing Alfresco on Windows (excluding JDK)

This section describes how to use the Alfresco OpenOffice installation wizard. This installation file has all of the required components for Alfresco, including OpenOffice, but it excludes the **Java Developer Toolkit (JDK)**.

1. Browse to the Alfresco Community Edition downloads area and download the following installation file: `Alfresco-Community-3.2-OOo-Setup.exe`.
2. Double-click on the downloaded file.
3. You may see an **Open File - Security Warning** message, prompting you to verify that you wish to run this software. To run the installation wizard, click on **Run**.
4. At the **Language Selection** prompt, select **English** and click on **OK**.
5. When prompted to confirm that you want to install Alfresco on your computer, click on **Yes**. The installation wizard launches.

6. In the **Welcome to the InstallJammer Wizard for Alfresco Community Edition** window, click on **Next**.
7. The **Browse for Folders** window displays, prompting you to locate your JDK installation. Browse to your JDK installation and click on **OK**. In the **Setup Type** window, select one of the following options:

Option	Description
Typical	To select all the components within the Alfresco full installation
Custom	To select individual components: <ul style="list-style-type: none">• Default Component (includes DM)• WCM• Java• OpenOffice• SharePoint Protocol• MySQL Database

Note that, by default, Alfresco installs an embedded MySQL database instance, which is installed locally in the Alfresco directory and runs on port number 3306.

8. (Optional) Select the database to use. You can choose one of the following:
 - If you wish to use an existing MySQL server rather than the embedded MySQL database, select the **Custom** option and deselect the **MySQL Database** option.
 - If you wish to install the embedded MySQL database in addition to your existing MySQL database, select the **MySQL Database** option. You will be prompted to specify an alternative port number after the installation starts.
 - If you wish to use a production database other than MySQL, deselect the **MySQL Database** option.
9. Click on **Next**.
10. In the **Choose Destination Location** window, click on **Next** to accept the default location for Alfresco or click on **Browse** to choose another location.
11. In the **Start Copying Files** window, review the settings and click on **Next**. The installation starts. You may see a prompt to notify you when the components are being installed.

12. (Optional) If you have an existing MySQL database, the **Database Configuration** window notifies you that the default database port 3306 is already used by the embedded MySQL database, and prompts you to enter the root password of your chosen database and an alternative database port.
13. (Optional) Click on **Next**.
14. The installation wizard writes the server name and host to the `alfresco-global.properties` file. You need to modify and run the example scripts in the `extras` directory to create the Alfresco database and user for this instance of MySQL.
15. In the **InstallJammer Wizard Complete** window, click on **Finish**. The installation is complete. The `README` file opens. When you close the `README` file, you are directed to the release notes.

Installing the Alfresco Tomcat bundle on Windows

This section describes how to install Alfresco using the Tomcat bundle on a Windows platform. Before you start, ensure that you have a JDK installed. Refer to the *Installing a JDK* section earlier in this chapter. Alfresco also requires Flash Player, SWFTools, and OpenOffice.org. For more information on installing these components, refer to the *Installing Alfresco components* section later in this chapter.

1. Browse to the **Alfresco Community Edition** downloads area and download the following installation file: `alfresco-community-tomcat-3.2.zip`.
2. Specify `C:\Alfresco` as the location for the download.
3. Extract the downloaded file into the location you specified.
4. Ensure that the `JAVA_HOME` environment variable points to the location of your JDK installation.
5. Open the `alfresco-global.properties` file.
6. Locate the property `dir.root`.
7. Change the property to show an absolute path for the `alf_data` directory. Replace backslashes with slashes. For example:
`dir.root=C:/Alfresco/alf_data`
8. This directory is created for you when you start the server.
9. Add the property settings for your preferred database.
10. Comment out the settings for the remaining databases.
11. Save the `alfresco-global.properties` file.

Installing Alfresco on Red Hat Linux

This section describes how to install Alfresco on Red Hat Linux using the following methods:

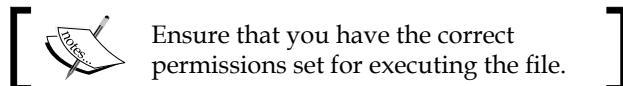
- Installation wizard
- Tomcat bundle

The installation wizard for Red Hat Linux installs the software and components that you require for running Alfresco. This installation wizard does not include JDK or an embedded database. Before you install Alfresco, ensure that you have the following softwares installed on your system:

- JDK 6
- MySQL database

These instructions are for working on a Graphical User Interface (GUI) in an X11 environment.

1. Browse to the Alfresco Community Edition downloads area and download the following installation file: `Alfresco-Community-3.2-Linux-x86-Install.`



2. Execute the downloaded file.
3. At the **Language Selection** prompt, select **English** and click on **OK**.
4. When prompted to confirm installation, click on **Yes**. The installation wizard launches.
5. In the welcome window, click on **Next**. The **Browse for Folder** window displays, prompting you to locate your JDK installation.
6. Browse to your JDK installation and click on **OK**.

7. In the **Setup Type** window, select one of the following options:

Option	Description
Typical	To select all of the components within the Alfresco full installation
Custom	To select individual components: <ul style="list-style-type: none">• Default Component (includes DM)• WCM• Java• OpenOffice• SharePoint Protocol

8. Click on **Next**.
9. In the **Choose Destination Location** window, click on **Next** to accept the default location for Alfresco.
10. In the **Start Copying Files** window, review the settings and click on **Next**. The installation starts. If you chose the **Typical** option, a prompt notifies you when OpenOffice and the SharePoint Protocol are being installed.
11. Click on **Next**.
12. (Optional) In the **Database Configuration** window, enter the root password of your MySQL database and a port number.
13. Click on **OK**.
14. In the **OpenOffice Location** window, browse to your OpenOffice location and click on **OK**. If you do not want to use OpenOffice for your document conversions, select the Alfresco installation folder as the location.
15. The installer looks for <path>/soffice.exe. As a temporary workaround, cp soffice.bin soffice.exe will help the installer finish correctly.
16. Click on **Next**. A prompt notifies you when the SharePoint Protocol is being installed.
17. In the **InstallJammer Wizard Complete** window, click on **Finish**.
18. The installation is complete. The README file opens. When you close the README file, you are directed to the release notes.

Installing the Alfresco Tomcat bundle on Linux

This section describes how to install the Alfresco Tomcat bundle on Linux.

Before you start, ensure that you have a JDK installed. Refer to the *Installing a JDK* earlier in this chapter. Alfresco also requires Flash Player, SWFTools, and OpenOffice.org. For more information on installing these components, refer to the *Installing Alfresco components* section.

1. Browse to the Alfresco Community Edition downloads area and download the following installation file: `alfresco-community-tomcat-3.2.tar.gz`.
2. Specify `/opt/alfresco/` as the location for the download.
3. If you change this location from `/opt/alfresco/`, edit `alfresco.sh` to point the `APP SERVER` variable to the Tomcat bundle location `/opt/alfresco/tomcat`.
4. Extract the downloaded file into the location you previously specified.
5. Ensure that the `JAVA_HOME` environment variable points to the location of your JDK installation.
6. Open the `alfresco-global.properties` file.
7. Locate the property `dir.root`.
8. Change the property to show an absolute path for the `alf_data` directory. For example: `dir.root=/opt/alfresco/alf_data`.
9. This directory is created for you when you start the server.
10. Add the property settings for your preferred database.
11. Comment out the settings for the remaining databases.
12. Save the `alfresco-global.properties` file.
13. (Optional) If you deployed previous versions of Alfresco, you can remove any temporary files created by your application server.

Installing Alfresco on Mac

The installation wizard for Mac installs all of the software and components that you require for running Alfresco. This installation wizard does not include JDK or an embedded database. Before you install Alfresco, ensure that you have the following softwares installed on your system:

- JDK 6
- MySQL database

If you have these software installed, proceed to the following steps:

1. Browse to the Alfresco Community Edition downloads area and download the following installation file:
`Alfresco-Community-3.2-MacOSXInstall.tar.gz`
2. Double-click on the downloaded file to unpack it.
3. Run the following executable file:
`Alfresco-Community-3.2-MacOSXInstall`
4. At the **Language Selection** prompt, select **English** and click on **OK**.
5. When prompted to confirm installation, click on **Yes**. The installation wizard launches.
6. In the welcome window, click on **Next**. The **Browse for Folder** window displays, prompting you to locate your JDK installation.
7. Browse to your JDK installation and click on **OK**.
8. In the **Setup Type** window, select one of the following options:

Option	Description
Typical	To select all of the components within the Alfresco full installation
Custom	To select individual components: <ul style="list-style-type: none">• Default Component (includes DM)• WCM• Java• OpenOffice• SharePoint Protocol

9. Click on **Next**.
10. In the **Choose Destination Location** window, click on **Next** to accept the default location for Alfresco.
11. In the **Start Copying Files** window, review the settings and click on **Next**. The installation starts. If you chose the **Typical** option, a prompt notifies you when OpenOffice and the SharePoint Protocol are being installed.
12. In the **InstallJammer Wizard Complete** window, click on **Finish**.
13. The installation is complete. The Readme file opens. When you close the Readme file, you are directed to the release notes.

Installing the Alfresco WAR on any platform

Use the Web Archive (WAR) file to install Alfresco on any platform. A WAR file is a JAR file used to distribute a collection of files (JavaServer Pages, servlets, Java classes, XML files, tag libraries, and static web pages) that together constitute a web application.

Use this method of installing Alfresco if you already have installed a JDK, a supported database, an application server, and the additional Alfresco components. The WAR zip includes the binaries for ImageMagick and pdf2swf. To download and install the Alfresco WAR file you need to carry out the following:

1. Browse to the Alfresco Community Edition download area.
2. Select and download one of the following files:
`alfresco-community-war-3.2.zip`
`alfresco-community-war-3.2.tar.gz`
3. Specify a location for the download.
4. Extract the downloaded file.
5. Copy the `alfresco.war` file and `share.war` file to the appropriate location for your application server, for example: `<TOMCAT_HOME>/webapps`.

If you deployed previous versions of Alfresco, you must remove any temporary files created by your application server.

Modifying the directory paths for Tomcat 6.x

If you install Tomcat 6.x separately, some of the directories that were present in Tomcat 5.x will not be present in the former. For example, Tomcat 6.x does not contain the `shared/classes` and `shared/lib` directories. Alfresco uses these directories to locate some of the configuration override files. This section describes how to configure Tomcat 6.x to use the correct directory structure and files for Alfresco:

1. Locate `<TOMCAT_HOME>`.
2. Create the `shared/classes` directory.
3. Open the `<TOMCAT_HOME>/conf/catalina.properties` file.
4. Change the value `shared.loader=` to the following:
`shared.loader=${catalina.base}/shared/classes,${catalina.base}/shared/lib/*.jar`
5. Copy the JDBC drivers for the database you are using to `lib/`.

6. Ensure that a copy of the commons-el.jar file is in the lib directory.
7. If you are using Java SE 6, copy any .jar files that needed to go into the Tomcat common/endorsed directory into the following directory:
...jdk6/jre/lib/endorsed.

Downloading the extension samples

Each Alfresco distribution includes a download containing sample extension files, such as the Spring configuration. You can use these sample extensions for advanced Alfresco customizations by downloading and installing the files mentioned below, as follows:

1. Browse to the Alfresco Community Edition downloads area.
2. Select and download one of the following files:
alfresco-community-sample-extensions-3.2.zip
alfresco-community-sample-extensions-3.2.tar.gz
3. Specify a location for the download. The extension samples are available for you to configure. Once you have downloaded the sample files, copy them into <classPathRoot>.

Deploying Share into a separate Tomcat instance

This task provides information for running Share in a separate Tomcat instance. These instructions are for Windows deployments, but Linux-based deployments can use the same methods:

1. Install a new Tomcat instance.
2. Modify the /conf/server.xml file for the new Tomcat instance as follows:
 - a. Change the port number in the line (for example, to 8006):
<Server port="8005" shutdown="SHUTDOWN">
 - b. Change the port number in the section (for example, to 8180):
<!-- Define a non-SSL HTTP/1.1 Connector on port 8080 -->
<Connector port="8080" ...>
3. Move the share.war file from the original Tomcat /webapps directory to the new Tomcat /webapps directory.

4. (Optional) Configure the original Alfresco Tomcat deployment.
5. Start the original Tomcat. You can use Alfresco-supplied batch files.
6. Ensure that a copy of the commons-el.jar file is in the Share Tomcat lib directory.
7. If you are running the Share Tomcat on a separate machine, you must modify the override file in the Share Tomcat web-extension directory, as follows:
 1. Open the webscript-framework-config-custom.xml file.
 2. Change any instance of the server and port to the correct name or IP address of the Alfresco server: `http://<server-name>:8080`.
 3. Start the new Share Tomcat. You can use copies of the Alfresco-supplied batch files, or your own.

Installing Alfresco components

This section describes how to install components that integrate with Alfresco. These can be installed any time before or after installing Alfresco. This section will take you through the installation of these components:

- Alfresco WCM
- WCM standalone deployment receiver
- OpenOffice
- ImageMagick
- Microsoft Office Add-ins
- Flash Player
- SWFTools
- TinyMCE language packs
- Alfresco Module Package
- Microsoft Office SharePoint Protocol Support

Installing Alfresco WCM

This section describes how to set up Alfresco Web Content Management (WCM) to an existing instance of Alfresco:

1. Browse to the Alfresco Community Edition download area.

2. Select one of the following files:
 - Windows: alfresco-community-wcm-3.2.zip
 - Linux: alfresco-community-wcm-3.2.tar.gz
3. Download the file into the Alfresco home directory. For example:
 - Windows: C:\Alfresco
 - Linux: /opt/alfresco
4. Browse to the Alfresco home directory and unzip the downloaded file.
5. If your unzip program asks about existing directories, allow this because no existing files will be overwritten.
6. In the root of the Alfresco home directory, copy the wcm-bootstrap-context.xml file to the <extension> directory.
7. Restart the Alfresco server.

This ensures that the Alfresco server starts to use the installed WCM components. To restart the Alfresco server, see the *Starting the Alfresco server* section further on in this chapter. WCM is installed and configured.

Verifying the WCM installation

Verify the WCM installation after you have installed, configured, and started the Alfresco server:

1. In the Alfresco home directory, open alfresco.log.
2. In alfresco.log, search for the following text: The Web Forms folder was successfully created: and The Web Projects folder was successfully created:
3. Check that the following additional spaces are in your Alfresco repository:
 - **Web Projects in Company Home**
 - **Web Forms in Data Dictionary**

WCM has been installed, configured, started, and verified. To use the Website Preview feature, start the Alfresco virtualization server (see the *Starting the Alfresco virtualization server* section later in this chapter).

Installing the WCM standalone deployment receiver

The standalone deployment receiver allows a web project from WCM to be deployed to a remote file server, typically a web or application server. The published files are then typically published by a web server such as Apache for static content, or an application server such as Tomcat or JBoss for dynamic content. To carry out the installation follow the steps mentioned below:

1. Browse to the Alfresco Community Edition download area.
2. Download one of the following files:
 - Windows: Alfresco-DeploymentCommunity-3.2-Setup.exe
 - Linux: Alfresco-DeploymentCommunity-3.2-Linux-x86-Install
3. At the **Language selection** prompt, click on **OK**.
4. At the **Install Alfresco Standalone Deployment Receiver** prompt, click on **Yes**.
5. In the **Welcome to the Alfresco Standalone Deployment Receiver** window, click on **Next**.
6. In the **Choose Destination Location** window, click on **Next** to accept the default location for Alfresco or choose another location. For example, C:\alfresco\deployment on Windows or /opt/alfresco/deployment on Linux.
7. In the **Deployment Settings** window, enter the following settings:

Deployment setting	Description
Temporary Data	The directory in which to store temporary data files
Log Location	The directory in which to store log data
Metadata Location	The directory in which to store metadata
Target Location	The directory in which to store deployment files
Name of default filesystem target	The default name of the filesystem target is default

8. For Windows directory locations, the backslashes need to be changed. For example, use C:\\directory1\\\\directory2. Alternatively, you can use the slash character as a separator, for example, C:/directory1/directory2.
9. Click on **Next**.
10. Enter a username and password for the user account that will administer the deployment receiver.

11. If you are using RMI as your transport protocol, enter the port numbers for the following:

Deployment setting	Description
RMI Registry Port Number	The port number for the RMI registry. Choose the default of 44101 to avoid conflict with the other services.
RMI Service Port Number	The port number to use for the RMI service. Choose this so that there are no conflicts with other services.

12. In the **Start Copying Files** window, click on **Next**.
 13. In the **InstallJammer Wizard Complete** window, click on **Finish**.

The deployment receiver, out of the box, is configured with a single filesystem deployment target.

Compiling and deploying the customizations on top of the WCM core

We already talked about the build process for the code deployment in the previous sections. So we have the build file ready and now we will see how we can deploy the code or customizations we have done for Alfresco WCM. There are two main approaches for this.

Integrating the code in the existing Alfresco WAR file

In this approach, we will integrate the files directly in the `alfresco.war` file. For this you need to run the `integrate-extension` target. This target has some dependent targets, which will compile the Java source files. Make the JAR file and then package everything in one ZIP file and then integrate it with the `alfresco.war` file.

Deploying the code as an AMP

AMP stands for **Alfresco Module Package**. Using this approach, you can install the code as a module. If you are installing using AMP, you should follow the folder-specific folder structure for that, which has already been described earlier in the *Source code tree* section. Apart from the folder structure, there are two other files that need to be created:

- `module.properties`: This will specify the metadata of the module with the ID and version number.

The sample `module.properties` file that we have used here will look like:

```
# Cignex module
module.id=Cignex
module.title=Cignex Website
module.description=Cignex Website module
module.version=1.0
```

- `file-mapping.xml`: This file will be used if you want to customize the way in which your AMP file is mapped into the WAR. If it is not provided, then the default mapping information will be used.

Here we have customized this for the `WEB-INF` folder, which generally contains the `faces-config-custom.xml` or any faces-related configuration files.

The content for `file-mapping.xml` file is:

```
# Custom AMP to WAR location mappings
/web/WEB-INF=/WEB-INF
```

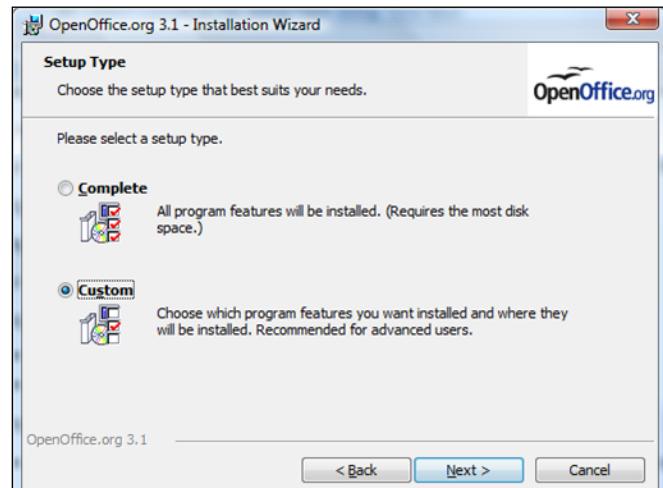
For deploying the code as an AMP, you need to run the `update-war` target. This target has some dependent targets, which will compile the Java source files. Make the JAR file and then create an AMP package with all of the customized files and finally install this AMP to the Alfresco WAR file and update the WAR file.

Installing OpenOffice

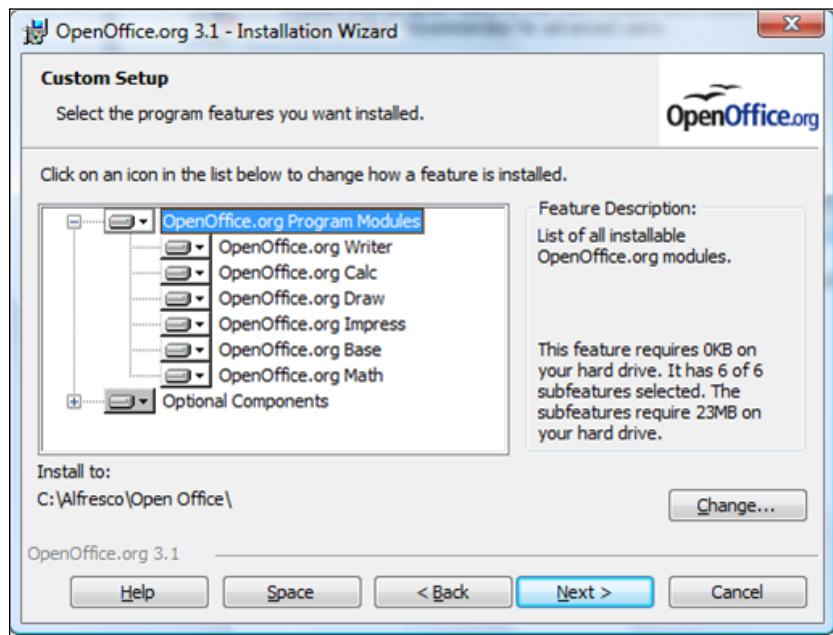
Within Alfresco, you can transform a document from one format to another, for example, a text file to a PDF file. To have access to these transformation facilities in Alfresco, you must install OpenOffice. This is optional and can be done any time after Alfresco is installed. If you installed Alfresco using an installation wizard, OpenOffice may already be installed.



1. Browse to the OpenOffice.org download site: <http://download.openoffice.org>.
2. Download the latest (stable) version of OpenOffice for your platform.
3. When prompted, specify a download destination.
4. Browse to the location of your downloaded file and install the application.
5. A wizard guides you through the installation.
6. Accept the license agreement and click on **Next**.
7. Enter customer information and click on **Next**.
8. Select the **Setup Type** as **Custom** and click on **Next**.



9. Change the installation directory to:
 - Windows: c:\Alfresco\OpenOffice
 - Linux: /opt/alfresco/OpenOffice



10. Optionally, select the files for which you want OpenOffice to be the default application and click on **Next**.
11. Start one of the OpenOffice programs for the initial registration and then close the program.
12. If the Alfresco server is running, stop and restart the server.
13. Modify the `ooo.exe=` property in the `<classPathRoot>\alfresco-global.properties` file to point to the OpenOffice binary `office.exe`.

Installing ImageMagick

To enable image manipulation in Alfresco, you must install and configure ImageMagick. Alfresco uses ImageMagick to manipulate images for previewing. If you installed Alfresco using one of the installation wizards, ImageMagick has already been installed. If however, ImageMagick has not been installed, you can choose to do so by downloading it and installing it to work with your Alfresco instance.

1. Verify if ImageMagick is already installed in your system. You can run the convert command, which is part of ImageMagick and is usually located in /usr/bin.
2. If ImageMagick is not in your system, browse to the ImageMagick download site and install the appropriate package for your platform.
3. Modify the img.root= and img.exe= properties in the <classPathRoot>/alfrescoglobal.properties file to point to the ImageMagick root directory.

For example, in Windows:

- Set the img.root= property to img.root=C:/Alfresco/ImageMagick.
- Set the img.exe= property to img.exe=C:/Alfresco/ImageMagick/bin/convert.exe.

And for Linux:

- Set the img.root= property to img.root=/ImageMagick.
- Set the img.exe= property to img.exe=/ImageMagick/bin/convert.exe.
- Ensure that you do not include a slash (/) at the end of the path. For example, /ImageMagick/.

Installing Microsoft Office add-ins

This task describes how to install Alfresco Add-ins for Microsoft® Office applications, such as Word, Excel, and PowerPoint. The Alfresco add-ins have been designed with Microsoft Office 2003 in mind. However, they are also compatible with Microsoft Office 2007. Before you start, make sure that:

- The .NET Programmability Support option is installed for each of the Office applications that you are installing the add-ins in (such as Word, Excel, and PowerPoint). To find these options, run the Office Setup program and expand the list of available options for each application. You may require your original Office 2003 install media to add these required components.
- The installing user has Windows administrator privileges.
- Any Microsoft Office applications on your system are NOT running, including Outlook if you use Word as an e-mail editor. If you are running Office 2007 on Windows Vista, note that Microsoft has rewritten the WebDAV parts of Vista, which means you will experience READ-ONLY access to the Alfresco repository over WebDAV. This is a known problem with Vista and affects many applications, including Microsoft's own SharePoint Server. There is no known workaround at the time of writing. CIFS access is unaffected and works as it does with Windows XP. Therefore, use CIFS to obtain read/write access to Alfresco using Windows Vista.

Now proceed to the following steps:

1. Browse to the Alfresco Community Edition download area.
2. Run the Microsoft setup file:
`alfresco-community-office2003-addins-3.2.zip`.
3. This example refers to the Office installer that installs all three add-ins.
You can also use individual installers to add Alfresco to one of the three Office applications.
4. These individual installers are:
`alfresco-community-word2003-addin-3.2.zip`
`alfresco-community-excel2003-addin-3.2.zip`
`alfresco-community-powerpoint2003-addin-3.2.zip`
5. Run `setup.exe`.
6. If required, the setup program will download the required components from the Microsoft website. These components are .NET 2.0 Framework and Visual Studio 2005 Tools for Office Second Edition runtime. The setup is complete.
7. Run the Office application, for example, run Word.
8. A welcome window with configuration options is displayed. You can return to the configuration options at any time using the link at the bottom of the add-in window.
9. In the **Web Client URL** field, enter the location of Alfresco Explorer.
For example: `http://server:8080/alfresco/`.
10. In the **WebDAV URL** field, append `webdav/` to the Alfresco Explorer URL.
For example: `http://server:8080/alfresco/webdav/`.
11. In the **CIFS Server** field, enter the path to the CIFS server. The add-in will try to auto-complete this value, but you should verify for the correct address.
For example: `\server_a\alfresco\` or `\servera\alfresco\`.
12. If you intend to use the CIFS interface to save documents via the add-in, it is very important that you are authenticated automatically. Limitations in the Microsoft Office APIs mean that an error is shown instead of an authentication dialog box if the Alfresco CIFS interface rejects your connection attempt. If you are not in an Enterprise environment, where it may not be possible to configure automatic authentication, you can map a network drive to the Alfresco CIFS interface instead.
13. In the **Authentication** area, enter your Alfresco username and password.

14. The add-in will always try to automatically log you into Alfresco in order to present your checked out documents and your assigned tasks. If you are using the CIFS interface, authentication is usually automatic. However, sometimes the add-in needs to present your Alfresco username and password for authentication. It is recommended that you enter and save your Alfresco credentials. All values are stored in the Windows registry and your password is encrypted against casual hacking attempts.
15. Click on **Save Settings**.

Installing Flash Player

Alfresco Share uses the Flash Player for viewing Flash previews and also when you use the multi-file upload facility. This is optional and may be installed after you have installed Alfresco.

1. Browse to the Adobe Flash Player download website: <http://www.adobe.com/products/flashplayer>.
2. Download the latest (stable) version of Flash Player for your platform.
3. Browse to the location of your downloaded file and install the application.
4. A wizard guides you through the installation.
5. When the installation is complete, click on **Close**.

Installing SWFTools

Alfresco Share uses the pdf2swf utility of the SWFTools for previewing PDF files. The pdf2swf utility generates one frame per page of fully-formatted text inside a Flash movie. To install the pdf2swf utility, you must install the complete SWFTools. This is optional and may be installed after you have installed Alfresco.

Installing SWFTools on Windows

Follow these steps to install SWFTools on Windows:

1. Browse to the SWFTools website: <http://www.swftools.org/>.
2. Download the latest (stable) version of the SWFTools for your platform. The Windows version is designated with the suffix .exe.
3. Download a version post 0.8.1 from 2007-02-28 because it does not support some functionalities Alfresco needs to render the preview.
4. Browse to the location of your downloaded file and install the application.
5. A wizard guides you through the installation.

6. Accept the license agreement and click on **Next**.
7. Select the installation directory.
8. Select whether you want to install the SWFTools for all users or only for the current user.
9. Click on **Next** to begin the install process.
10. By default, the options to **Create start menu** and **Create desktop shortcut** are selected.
11. Click on **Finish**.
12. Modify the `swf.exe=` property in the `alfresco-global.properties` file to point to the SWFTools root directory.
For example: `swf.exe=C:/Alfresco/bin/pdf2swf`.
13. Ensure that you do not include a slash (/) at the end of the path. For example, `/usr/bin/`.
14. The SWFTools are installed. For the most up-to-date instructions on installing the SWFTools, refer to the SWFTools website.

Installing SWFTools on Linux

Alfresco Share uses the features provided in the development snapshots of the tools. For Linux, there is no binary version, so you need to compile a development snapshot. Before you compile, ensure that the following packages are installed in your machine:

- `zlib-devel`
- `jpeg-devel`
- `giflib-devel`
- `freetype-devel`
- `gcc`
- `gcc-c++`

You can download and install all of these packages using the following command:

```
yum install zlib-devel libjpeg-devel giflib-devel freetype-devel gcc  
gcc-c++
```

To download and install the SWFTools:

1. Browse to the SWFTools website.
2. Download the latest version of the SWFTools for your platform. The Unix version is designated with the suffix `.tar.gz`.

3. Download a version post 0.8.1 from 2007-02-28 because it does not support some functionalities Alfresco needs to render the preview. The following version has been tested and verified by Alfresco as being fully functional:
<http://www.swftools.org/swf-tools-2008-10-08-0802.tar.gz> (you may have to copy this URL and paste it into a download manager).
4. Unpack the `tar.gz` file. The installation file contains detailed instructions on how to compile and install the SWFTools.
5. Change to the directory containing the source code.
6. Type the following command to configure the package for your system:
`./configure`
7. If you see a message on Red Hat Linux that states your operating system is unknown, then use the following setting: `./configure-build=x86_64-pc-linux-gnu`.
8. If you have an issue on Solaris with the lame libs, you can disable the making of portions of SWFTools that use lame by using the following setting: `./configure -disable-lame`.
9. Type the following command to compile the package:
`make`
10. Optionally, you can run the `make check` command to run any self-tests that come with the package.
11. Type the following command to install the programs, data files, and documentation:
`make install`
12. By default, the files are installed to the `/usr/local/bin` directory.
13. Modify the `swf.exe=` property in the `alfresco-global.properties` file to point to the SWFTools root directory, for example: `swf.exe=/usr/bin/pdf2swf`.
14. Ensure that you do not include a slash (/) at the end of the path. For example, `/usr/bin/`.

The SWFTools are installed. For the most up-to-date instructions on installing the SWFTools, refer to the SWFTools website.

Installing TinyMCE language packs

Translations in Alfresco use the language pack supplied in the default bundle. This default bundle includes English (en), French (fr), German (de), Japanese (jp), Spanish (es), and Italian (it). If you have a translation that is not supplied with Alfresco, then you must add the appropriate TinyMCE language pack for the translation to work correctly.

If you installed Alfresco using one of the installation wizards or bundles, the default language packs are already installed. You can also download and install the language packs from the TinyMCE website.

1. Browse to the TinyMCE website: http://tinymce.moxiecode.com/download_i18n.php.
2. Download the required TinyMCE language pack.
3. Unpack the language file:
 - For Share, unpack to: <TOMCAT_HOME>/webapps/share/modules/editors/tiny_mce.
 - For Explorer, unpack to: <TOMCAT_HOME>/webapps/alfresco/scripts/tiny_mce.
4. Ensure that the browser cache is cleared or refresh the page.

Installing an Alfresco Module Package

An Alfresco Module Package (AMP) is a bundle of code, content model, content, and the directory structure that is used to distribute additional functionality for Alfresco. This section describes how to install an AMP in an Alfresco WAR using the **Module Management Tool (MMT)**. The MMT is used to install and manage AMP files, and it is included in the Alfresco installers. The MMT is also available as a separately downloadable JAR file from the Alfresco release download area (`alfresco-mmt-3.2.jar`).

For Tomcat, alternatively, run the `apply_amps.bat` command, which is in the root Alfresco directory. This batch file applies all of the AMP files that are located in the `amps` directory. You can verify the correctness of installation of AMP as follows:

1. Browse to the `/bin` directory:
 - Windows: `C:\Alfresco\bin`
 - Linux: `/opt/alfresco/bin`

2. Run the following command:

```
java -jar alfresco-mmt.jar install <AMPFileLocation>  
<WARFileLocation>[options]
```

Where:

Option	Description
<AMPFileLocation>	The location of the AMP file that you want to install.
<WARFileLocation>	The location of the WAR file for your Alfresco installation.
-verbose	Install command [options]. Enables detailed output containing what is being updated and to where it is being copied.
-directory	Install command [options]. Indicates that the AMP file location specified is a directory. All AMP files found in the directory and its sub-directories are installed.
-force	Install command [options]. Forces installation of AMP regardless of currently installed module version.
-preview	Install command [options]. Previews installation of AMP without modifying the WAR file. It reports the modifications that will occur on the WAR without making any physical changes, for example, the changes that will update existing files. It is a good practice to use this option before installing the AMP.
-nobackup	Indicates that the WAR will not be backed up before the AMP is installed.

3. This command installs the files found in the AMP into the Alfresco WAR. If the module represented by the AMP is already installed and the installing AMP is of a higher release version, then the files for the older version are removed from the WAR and replaced with the newer files. The following command shows an example of how to install the `example-amp.amp`, and assumes that the AMP file is in the same directory as WAR file:

```
java -jar alfresco-mmt.jar install example-amp.amp alfresco.war  
-preview
```

4. Review the modification to check the changes that will update any existing files.

5. The following example will install the AMP file:

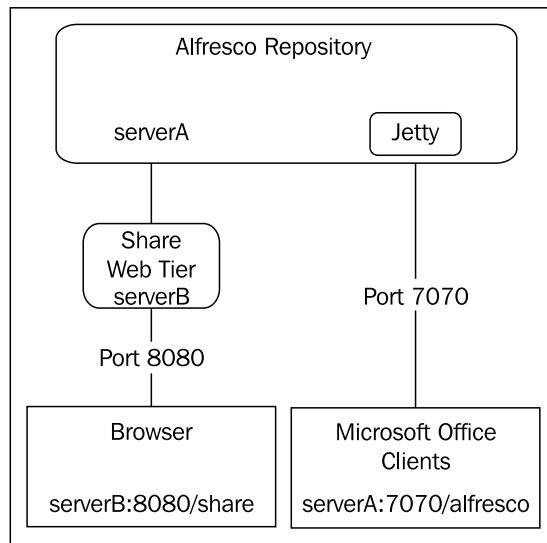
```
java -jar alfresco-mmt.jar install example-amp.amp alfresco.war  
-verbose
```
6. The modified Alfresco WAR can then be redeployed back into your application server. On restarting the application server, the console will show that the custom class was initialized during startup.
7. Verify that the AMP is installed using the MMT list command. For example:

```
java -jar alfresco-mmt.jar list <WARFileLocation>
```
8. This command provides a detailed listing of all of the modules currently installed in the specified WAR file. When the repository is next started, the installed module configuration will be detected, and the repository will be bootstrapped to include the new module functionality and data. It is not recommended that you overwrite an existing file in an AMP; however, it is sometimes necessary. The MMT makes a backup copy of the updated file and stores it in the WAR. When an update of the module occurs and the old files are removed, this backup will be restored prior to the installation of the new files. Problems may occur if multiple installed modules modify the same existing file. In these cases, a manual restore may be necessary if recovery to an existing state is required. Some application servers (notably Tomcat) do not always fully clean up their temporary working files and this can interfere with successful installation of an AMP file. To remedy this situation, it is recommended that you delete (or move) the Tomcat `work` and `temp` directories while Tomcat is shut down.

Installing Microsoft Office SharePoint Protocol Support

The Microsoft Office SharePoint Protocol Support offers Microsoft users greater choice by providing a fully-compatible SharePoint repository that allows the Microsoft Office Suite applications (for example, Word, PowerPoint, Excel) to interact with Alfresco as if it was SharePoint. This enables your users to leverage the Alfresco repository directly from Microsoft Office.

You can also use Microsoft Office SharePoint Protocol Support to enable online editing for Office documents within Alfresco Share. It enables your users to modify Office files without checking them in and out. Alfresco locks the file while it is being modified and releases the lock when the file is saved and closed. The following diagram shows the architecture of the SharePoint Protocol Support in relation to an Alfresco installation.



The SharePoint Protocol Support architecture embeds a Jetty web server within the Alfresco repository. Microsoft Office clients communicate directly with the Jetty server using WebDAV-like calls with proprietary extensions and on a different port number from Alfresco Share. This port number can be configured in the SharePoint Protocol Support properties files.

Installing the SharePoint Protocol Support AMP

The SharePoint Protocol support functionality is installed from an Alfresco AMP. If you use the Windows or Linux installers to install Alfresco, the SharePoint Protocol Support is installed by default. These instructions describe how to install the SharePoint Protocol Support into the Alfresco WAR. When you install this file, it responds to the SharePoint requests from Office, and therefore allows Alfresco to appear as the SharePoint server.



If you installed Alfresco using an installation wizard, the SharePoint Protocol may already be installed.

Follow the steps listed below to apply the SharePoint AMP to your Alfresco instance:

1. Shut down your Alfresco server.
2. Browse to the Enterprise download area.
3. Browse to the Alfresco Community Edition download area.
4. Download the vti-module.amp file.
5. Move the file to the amps directory.
6. Install the vti-module.amp file into the alfresco.war file using the Module Management Tool (MMT). The vti-module.amp file holds the functionality for the SharePoint connector. For Tomcat, alternatively, run the apply_amps.bat command, which is in the root Alfresco directory. This batch file applies all of the AMPS that are located in the amps directory.
7. Start your Alfresco server to expand the directory structure.
8. Verify that you have applied the SharePoint AMP to Alfresco by checking that you have the following directory:

```
/webapps/alfresco/WEB-INF/classes/  
alfresco/module/org.alfresco.module.vti/context
```

Configuring SharePoint Protocol Support

The SharePoint Protocol Support functionality uses the properties in the default configuration file called vti.properties in <TOMCAT_HOME>/webapps/alfresco/WEB-INF/classes/alfresco/module/org.alfresco.module.vti/context.

Custom properties and overrides can be set in the alfresco-global.properties file. Ensure that you have applied the SharePoint Protocol Support AMP. This can be done and verified as follows:

1. Open the alfresco-global.properties file.
2. Add the following properties:

```
vti.server.port=7070  
vti.alfresco.deployment.context=/alfresco  
vti.alfresco.alfresoHostWithPort=http://your-host:8080  
vti.share.shareHostWithPort=http://your-share-host:8080
```

The following table describes the properties.

Property	Description
vti-server.port	Use this property to configure on which port the SharePoint server listens. The default port number is 7070.
vti.alfresco.deployment.context	Use this property to specify the URL that you enter in the Microsoft Office Shared Workspace. The default is set to /alfresco. For example, the default /alfresco sets the URL to http://your-sharehost:7070/alfresco .
vti.alfresco.alfrescoHostWithPort	Use this property to specify the IP address or host name of the Alfresco server. Replace <code>your-host</code> with the location of your Alfresco server.
vti.share.shareHostWithPort	Use this property to specify the Share instance. Replace <code>your-share-host</code> with the location of your Share instance. This property includes the prefix <code>http://</code> , which allows for HTTPS support.

The `vti.properties` file in `<TOMCAT_HOME>/webapps/alfresco/WEB-INF/classes/alfresco/module/org.alfresco.module.vti/context` contains the full list of properties. Do not change the properties in this file; use only the `alfresco-global.properties` file. Now, perform the following:

1. Save the `alfresco-global.properties` file.
2. Restart your Alfresco server. The Microsoft SharePoint Protocol Support functionality is installed and configured.

Configuring SharePoint Protocol for Online Editing

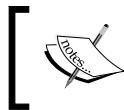
There is a known issue with Office 2003 and 2007 Office documents opening as read-only in Internet Explorer for all versions before Vista SP1. Refer to the knowledge base article 870853 on the Microsoft website to enable the Online Editing functionality.

Running Alfresco

This section describes how to start and stop the following:

- Alfresco server
- Alfresco Explorer
- Alfresco Share

- Virtualization server
- Standalone deployment receiver



Before running Alfresco, you may need to modify the configuration settings. If you have installed Alfresco using one of the installation wizards, the configuration is set for you.



Starting the Alfresco server

Once you have installed Alfresco, you can test the installation by starting the server.

For Windows:

1. Browse to C:\alfresco and double-click on alf_start.bat.
2. Or if you installed Alfresco using the installer, from the **Start** menu, select **All Programs | Alfresco Community Edition | Start Alfresco Server**.

A command prompt opens with a message indicating the server has started.

```
INFO: Server startup in nnnn ms
```

For Linux:

1. Browse to /opt/alfresco/ and run alfresco.sh start.
2. The default shell is sh. You can edit the alfresco.sh file to change to your preferred shell.

Starting Alfresco Share

Once you have installed Alfresco, you can start Alfresco Share using a browser.

1. Browse to the location of your Alfresco installation. If you installed Alfresco on your local machine, browse to http://localhost:8080/share. In Windows, alternatively, you can click on the **Start** menu and select **All Programs | Alfresco Community Edition | Alfresco Share**. Alfresco Share opens.
2. Log in using **admin** as the default username and password.

Starting Alfresco Explorer

Once you have installed Alfresco, you can start Alfresco Explorer using a browser.

1. Browse to the location of your Alfresco installation. If you installed Alfresco on your local machine, browse to <http://localhost:8080/alfresco>. In Windows, alternatively, you can click on the **Start** menu, and select **All Programs | Alfresco Community Edition | Alfresco Explorer**. Alfresco Explorer opens.
2. Log in using **admin** as the default username and password.

Stopping the Alfresco server

For Windows:

1. Browse to `C:\alfresco` and double-click on `alf.stop.bat`.
2. Or click on the **Start** menu and select **All Programs | Alfresco Community Edition | Stop Alfresco Server**.

The command prompt that opened during startup closes. Alfresco has now stopped.

For Linux:

1. Browse to `/opt/alfresco/` and run `alfresco.sh stop`.

Starting the Alfresco virtualization server

If you have installed Alfresco WCM, you can use the **Preview Website** feature by starting the Alfresco virtualization server.

For Windows:

1. Browse to `C:\alfresco` and double-click on `virtual.start.bat`.
2. Or click on the **Start** menu and select **All Programs | Alfresco Community Edition | Start Virtual Server**.

For Linux:

1. Browse to `/opt/alfresco/` and run `virtual.alf_sh start`.

Stopping the Alfresco virtualization server

For Windows:

1. Browse to C:\alfresco and double-click on virtual.stop.bat.
2. Or you can also click on the Start menu and select **Programs | Alfresco Community Edition | Stop Virtual Server.**

For Linux:

1. Browse to /opt/alfresco/ and run sh virtual_alf.sh stop.

Starting the deployment engine

The standalone deployment engine is implemented as a set of Java libraries and is multi-platform. Bourne shell scripts are provided for Unix, and Windows batch files are provided for Windows.

For Windows, to start the standalone deployment engine:

1. Open a command prompt and run the deploy_start script, or select **Start Menu | All Programs | Alfresco Standalone Deployment Receiver | Start Alfresco Standalone Deployment Receiver.**
2. The **Start Menu** action is available if you have used the deployment installer to install the Standalone Deployment Engine. This action is calling the deploy_start.bat script. It is also possible to install the standalone deployment engine as a Windows service, which can automatically start when Windows starts.

For Linux, to start the standalone deployment engine, open a command prompt and run the deploy_start.sh script.

1. When deploying to a deployment engine running on a multi-NIC system, it may be necessary to bind the RMI registry to a particular IP address. To do this, add the following to the Java command in deploy_start.sh or deploy_start.bat:
`-Djava.rmi.server.hostname=x.x.x.x`
Here x.x.x.x is the IP address assigned to the NIC to which you want to bind.

Stopping the deployment engine

The standalone deployment engine is implemented as a set of Java libraries and is multi-platform. Bourne shell scripts are provided for Unix, and Windows batch files are provided for Windows.

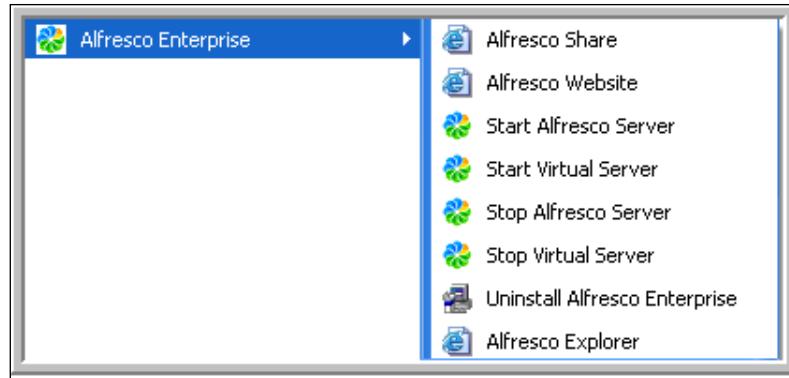
For Windows, to stop the standalone deployment engine:

1. Open a command prompt and run `deploy_stop.bat`, or select **Start Menu | All Programs | Alfresco Standalone Deployment Receiver | Stop Alfresco Standalone Deployment Receiver**.

For Linux, to stop the standalone deployment engine open a command prompt and run the `deploy_stop.sh` script.

Starting and stopping Alfresco as a console application

The options for starting and stopping Alfresco as a console application can be viewed by clicking on **Start | All Programs | Alfresco Enterprise** as shown in the following screenshot:



The options there are discussed next:

1. **Stop Alfresco Server:** This option is used to stop the Alfresco server. It stops the MySQL server and the Tomcat application server.
2. **Start Alfresco Server:** Use this option to start the Alfresco server as a console application. This will start the MySQL server and the Tomcat server.
3. **Stop Virtual Server:** This option is used to stop the Alfresco Virtual server. It stops the Virtual Tomcat application server.

4. **Start Virtual Server:** Use this option to start the Alfresco Virtual server, mainly used for Alfresco WCM. This will start the Virtual Tomcat Server.
5. **Alfresco Explorer:** This option is used to open Alfresco Web Client in the browser.
6. **Alfresco Share:** This option is used to open Alfresco Share in the web browser.
7. **Alfresco Website:** This option is used to open Alfresco's website in the web browser.

Alternatively, you can always start, stop, and restart the Tomcat application server and the MySQL database server manually by going to their respective directories. It gives more control to the user. However, the console option gives batch files to perform the start/stop procedures in a consolidated way, relieving the user of any unwanted errors.

Installation folder structure

Let's take a peek into the installation directory <alfresco_installation_folder> to look at the folders:

- **alfresco:** All of the shortcuts to installing, uninstalling, starting, and stopping Alfresco as a Windows service, and restarting, stopping, and starting of Alfresco as a normal console application from the **Start** menu of Windows points to this folder.
- **alf_data:** All of the Alfresco content and Lucene indexes are stored in this directory.
- **amps:** All of the AMP extensions files are required to be put here, and then use the `apply_amps` script to perform the update.
- **bin:** This directory contains the sub-installations of Alfresco. The main installation scripts in the `alfresco` directory calls the sub-scripts in this folder to start the sub-Alfresco Tomcat component of the installation, creating and setting up the permissions for the Alfresco MySQL database. This folder is very useful for people going for the manual installation rather than using an installer.
- **extras:** Contains additional files such as space template for records management file plan, which can be imported into the Alfresco repository.
- **java:** As is evident by the name, it contains the Java Development Kit. All of the Alfresco development is done using Java as the core programming language.
- **licenses:** This directory contains the licenses for Alfresco, MySQL, Apache, and licenses for the other third-party applications used inside Alfresco.

- **tomcat:** Again, as evident from the name, this directory holds the Tomcat installation where the Alfresco application is deployed as a WAR file. You can see the `alfresco.war` and `share.war` files in the `webapps` sub-folder of this directory.
- **openoffice:** This directory contains the entire portable Office suite installation that is used for word processing, spread sheet processing, and so on.
- **virtual-tomcat:** This folder contains the customized Tomcat, which is used for previewing the files in the WCM.
- The `README` file gives information about using CIFS and some troubleshooting tips.

You can uninstall the program by clicking on the `uninstall.exe` application.

Configuring Alfresco as a Windows service

You can also configure Alfresco as a Windows service in a standard Alfresco/Tomcat Installation. With the default installation, Alfresco is bundled as a web application that launches within Tomcat. To configure Alfresco to run as a Windows service, you need to set up Tomcat to run as a Windows service.

To configure Alfresco as a Windows service:

1. Open a command prompt.
2. Go to the `<alfresco_installation_folder>/tomcat/bin` directory.
3. Use the following commands:

```
service.bat install alfresco
tomcat5.exe //US//alfresco --DisplayName "Alfresco Server"
tomcat5.exe //US//alfresco --JvmMs=256 --JvmMx=512 --JvmSs=64
tomcat5.exe //US//alfresco --JavaHome=<alfresco_installation_
folder>/java
tomcat5.exe //US//alfresco --Environment ALF_HOME=<alfresco_
installation_folder>/
tomcat5.exe //US//alfresco --Environment PATH=<alfresco_
installation_folder>/bin;%PATH%
tomcat5.exe //US//alfresco --StartPath <alfresco_installation_
folder>--Startup auto
```

4. To uninstall the service, go to the `<alfresco_installation_folder>/tomcat/bin` and enter the following command:

```
service.bat uninstall alfresco
```

5. To edit your service settings, navigate to <alfresco_installation_folder>/tomcat/bin and enter the following command:
`tomcat5w.exe //ES//alfresco`
6. To start the service, locate the service named Alfresco Server in your Windows Service control panel and start Alfresco from this control panel.

Summary

In this chapter we learned:

- Alfresco can be installed using a number of different methods as per your preference.
- You can install Alfresco as a standard Web Archive (WAR) file to deploy on your existing application server, or as a bundle that includes a preconfigured Tomcat server, the Alfresco WAR (`alfresco.war`), batch files, database setup scripts, and a sample extensions folder.
- The various flavors of installations mentioned in this chapter are aimed to help you select what is best suited for your purpose and optimize your development experience with Alfresco WCM.

The next chapter will take you through creating web forms and web projects and the various aspects. You shall also learn about creating, editing, and updating content and the Advanced Versioning Manager.

3

Getting Started with Alfresco WCM

This chapter introduces the basic concepts of Alfresco Web Content Management from a user's perspective. It demonstrates how to set up and configure the Alfresco Explorer for managing a web project and showcases a sample web publishing scenario.

At the end of this module, you will be able to:

- Create a web project in Alfresco
- Bulk load an existing website content into a web project
- Configure the Virtualization Server to enable the previewing of content
- Publish the content from User Sandbox to Staging Sandbox after an approval process

Understanding the basics of WCM

We assume you are familiar with Alfresco Explorer, which is a web-based user interface to the Alfresco WCM. Before beginning the tutorial, you must ensure that:

1. The Alfresco ECM and WCM applications are installed.
2. The Alfresco Web Server is running.
3. The Alfresco Virtualization Server is running.

Log in to Alfresco WCM web interface

Using any web browser, you can connect to the Alfresco Explorer application. You will be able to manage users, security, content, business rules, and everything related to the enterprise content stored in Alfresco through the web client.

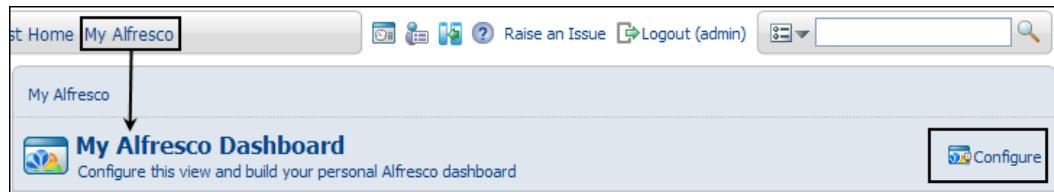
1. Access Alfresco web interface using the URL `http://localhost:8080/alfresco`. Or use the URL as per your installation.
2. Log in as Administrator using the default credentials:
 - **Username:** admin
 - **Password:** admin
3. Click on **Login**.

Once you log in, you will notice **My Alfresco Dashboard** as the home page.

My Alfresco Dashboard

The **My Alfresco Dashboard** is a configurable area where you can select from a list of preconfigured dashlets and components to construct your own page.

To start configuring your dashboard, click on the **Configure** icon given in **My Alfresco Dashboard** as shown in the following screenshot. The **Configure Dashboard Wizard** will open up allowing you to select the dashboard layout and dashlets:



When configuring the dashboard, each component you select is displayed as a pane on the dashboard. The available components are as follows:

Component	Description
Getting Started	This displays helpful information for getting started, including links to an Alfresco demonstration, a feature tour, and the Alfresco online help system. Descriptions of some common tasks you may want to perform in Alfresco are also included.
My Tasks To Do	Here all incomplete tasks assigned to you are listed. In this pane, you can manage and reassign your tasks.

Component	Description
All Active Tasks	The All Active Tasks component displays all active tasks – those assigned to us and those assigned to other users. In this pane you manage and reassign tasks.
My Pooled Tasks	All pooled tasks relevant to us are displayed here. These will be pooled tasks assigned directly to us or tasks assigned to a group that you belong to. In this pane you can manage and reassign your tasks.
My Completed Tasks	The My Completed Tasks pane displays all tasks that you have completed. In this pane you can view and cancel workflows related to a completed task.
OpenSearch	This component displays the OpenSearch pane on your dashboard, which provides the ability to search across multiple repositories. This is the same search pane that is displayed in the sidebar.
My Document List	The My Document List component displays the documents within your home space. In this pane you can view the document details, preview the document, and perform various actions on the content item without leaving the dashboard, including Check Out , Check In , Update , Delete , Edit Details , and Download . Use the document type filter in this pane to specify the information to display: All Items , Word Documents , HTML Documents , PDF Documents , or Recently Modified .
My Spaces List	This displays the current repository. In this pane you can navigate the repository from the dashboard and perform various actions, including creating spaces, uploading content, and viewing content items. Use the filter in this pane to specify the type of information to display: All Items , Spaces , Documents , My Items , or Recently Modified .
My Tasks	Displays all incomplete tasks assigned to you. In this pane you can filter the tasks by due date (Due Today , Next 7 days , No due date , Overdue) and manage the tasks. This component is similar to the My Tasks To Do component but with a different user interface.
My Web Forms	This displays the web projects you have been invited to and the web forms associated with each project. The My Web Forms pane enables you to preview a web project, create web content from a web form, and display the Sandbox view for a project – all from the dashboard.
My Web Files	All the web projects you have been invited to and the Modified Items list from your sandbox for each project are listed here. This pane enables us to edit and preview individual content items, preview a web project, and display the Sandbox view for a project – all from the dashboard.

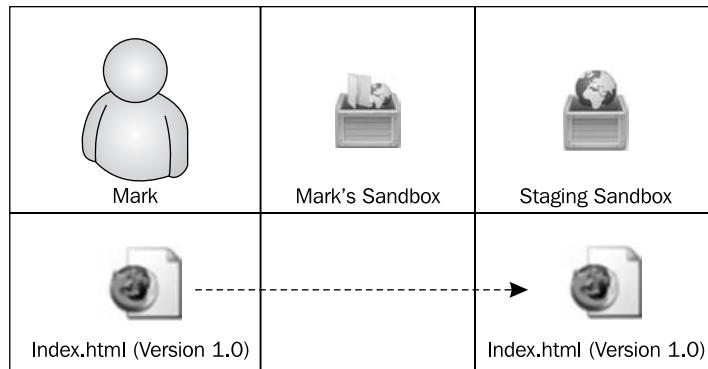
Web projects are created in the **Company Home | Web Projects** space. This space is automatically created if either WCM was included with the base install of Alfresco or if WCM is installed separately.

Web project Sandboxes

Before you delve into the details of web projects, it is worthwhile to understand the concept of sandboxes.

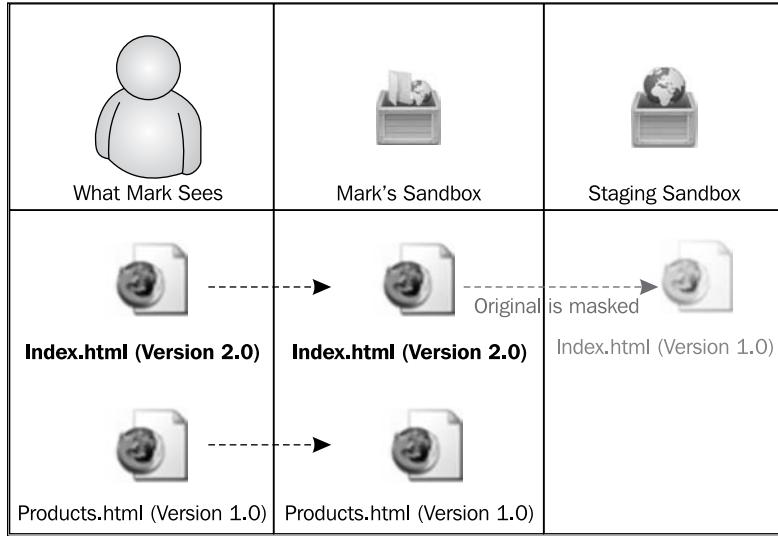
- Each user in a web project is provided their own sandbox.
- Each web project has a Staging Sandbox. All user changes, such as uploading a file in the system, are stored in their sandbox until they are submitted to the Staging Sandbox.
- The User Sandbox also maps all files in the Staging Sandbox, which have not been modified by the current user, providing a complete view of the web project for that specific user with their current changes.
- The Staging Sandbox has all files submitted by each user, where the current view is the most recently submitted version of a specific file.

The following diagram illustrates what user Mark sees with no modified file in his sandbox. Note how he sees the Staging Sandbox's version of `Index.html`:

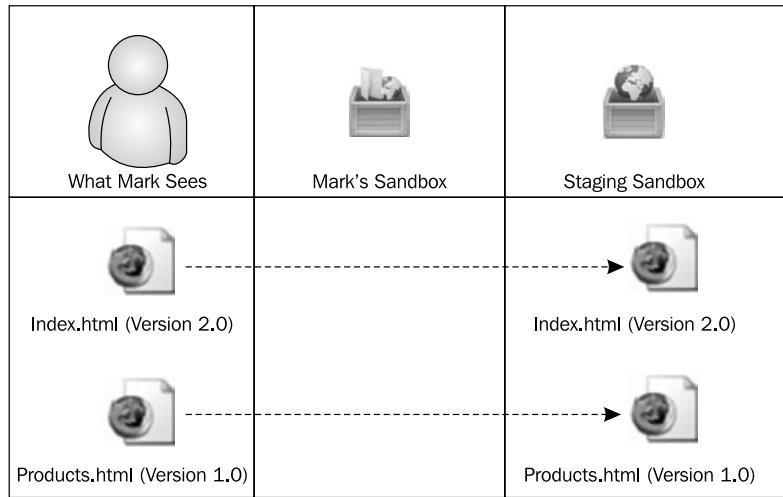


Mark creates or uploads a new products page. He will see the new `Products.html` file and the staging's version of the `Index.html` file.

Now Mark modifies the `Index.html` file to link to the `Products.html` file. This creates a new version of the `Index.html` file in his sandbox. He no longer sees the current Staging Sandbox's version of the file as illustrated in the following diagram:



Mark is satisfied with his changes and submits them through workflow and the two files are promoted to staging. His versions of the `Index.html` and `Products.html` files become the Staging Sandbox's current versions as shown in the next diagram:



User Sandbox interface

The following screenshot shows the WCM interface for the user's sandbox at the bottom and the **Staging Sandbox** at the top:

The screenshot displays the WCM interface for managing web projects. At the top, a breadcrumb navigation shows 'Company Home > Web Projects > CIGNEX'. Below this is a header bar with the project name 'CIGNEX' and a globe icon. A sub-header 'Use this view to browse the staging area and user sandboxes for a web project.' is present. On the right, there is an 'Actions' dropdown menu.

The interface is divided into two main sections:

- Staging Sandbox:** This section contains a summary of the 'Staging Sandbox'. It includes a small icon of a folder with a globe, the title 'Staging Sandbox', and details like 'Created On: 20 February 2010' and 'Created By: admin'. It also indicates that 'There are 3 users working on this web project.' Below this, there are two expandable sections: 'Recent Snapshots' and 'Content Awaiting Launch'.
- User Sandboxes:** This section lists 'My Sandbox (Content Manager)' with its own set of actions: 'Browse Website', 'Preview Website', 'Submit All', 'Undo All', and 'More Actions'. Below this, there are two more expandable sections: 'Modified Items' and 'Web Forms'.

Using this interface users can:

- Browse their sandbox and the Staging Sandbox web project hierarchy using the appropriate **Browse Website** action.
- Preview the website based on the state of their sandbox or the Staging Sandbox using the appropriate **Preview Website** action.
- Submit all modified files from their sandbox to a workflow using the **Submit All** action.
- Roll back all modified files in their sandbox using the **Undo All** action.
- Refresh their sandbox or the Staging Sandbox using the appropriate **Refresh** action. This is useful to view a new file submitted in another fashion (CIFS) in the user's sandbox, or new submission in the Staging Sandbox from other users.

Advanced Versioning Manager (AVM)

Sandboxes are backed by the **Advanced Versioning Manager (AVM)**. The AVM is an advanced repository implementation designed to support the version control requirements of large websites and web applications. It is a completely different implementation from the stores used by Alfresco Document and Records Management. Think of the AVM as a virtual time machine. For example, if you want to look at how a particular file appeared one month ago, you will be able to see it against the backdrop of how everything else looked one month ago. The state of the entire repository will be recorded in an efficient manner at each moment in its transaction history. The AVM provides the following features, including some found in other software tools like Subversion and TeamSite:

- Directories, renames, and file metadata will be versioned
- Every moment in transaction history is available immediately as a read-only filesystem
- All directories are first-class versioned objects
- Branching will be a constant-time operation
- Choice of database backends (for example, MySQL, Oracle, PostgreSQL)
- Access through CIFS

Alfresco AVM is an advanced store implementation designed to support the version control requirements of managing large websites and web applications. The AVM consists of a forest of content stores that can be used to manage the development and staging of web content and associated source code. Each AVM store is loosely modeled on Subversion, providing source code control for websites. Each AVM store supports an extended versioning model, including:

- File-level version control
- File-level branching
- Directory-level version control
- Directory-level branching
- Store-level version control (snapshots)
- Store-level branching

In addition to these extended versioning operations, the AVM also supports the following capabilities:

File version comparison

- Between two file versions on the same branch within a single store
- Between two file versions on different branches within the same store
- Between two file versions between any two stores

Directory version comparison

- Between two directory versions on the same branch within a single store
- Between two directory versions on different branches within the same store
- Between two directory versions between any two stores

Store version comparison

- Between two versions of any single store
- Between any two stores

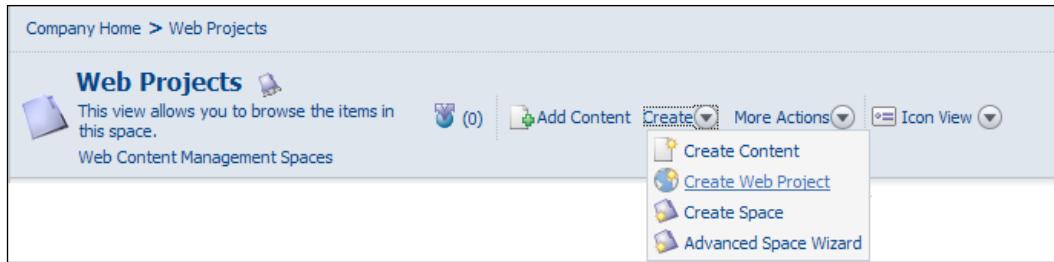
AVM's support for file, directory, and store comparison provides the basis for synchronization operations, for example, to synchronize any two branched versions of a source tree within an AVM store or maintained in separate AVM stores.

The web project

A web project contains all of the content and assets (files and images) required for the website(s). All actions for managing a specific website, creating content, uploading assets, reviewing and previewing content, workflows, managing web forms, managing rendering templates, and publishing are associated to a web project.

Create the web project

To create a new web project, go to the **Company Home | Web Projects** space, select the **Create** drop-down menu and click on the **Create Web Project** link as shown in the following screenshot:



You will see the **Create Web Project Wizard**, which includes seven steps in creating a web project as follows:

1. Specify the basic web project details.
2. Indicate if you are creating a new project or using the structure of an existing project.
3. Configure deployment servers.
4. Select and configure web forms for generating site content.
5. Select and configure workflow for content added that is not generated by a web form.
6. Add users and assign user roles to them.
7. E-mail a notification to the selected users.

The **Create Web Project Wizard** is shown in the following screenshot for your reference:

The screenshot shows the 'Create Web Project Wizard' interface. At the top, it says 'Company Home > Web Projects' and 'Create Web Project Wizard'. Below that, a sub-header says 'This wizard helps you create a new web project space.' On the left, a sidebar titled 'Steps' lists the following steps: 1. Web Project Details (which is selected and highlighted in blue), 2. Create From Existing Web Project, 3. Configure Deployment Receivers, 4. Configure Web Forms, 5. Configure Workflow, 6. Add Users, 7. Email Users, and 8. Summary. The main panel is titled 'Step One - Web Project Details' with the sub-instruction 'Enter the information about the web project.' It contains several input fields and dropdown menus:

- Web Project Details** section:
 - Name:
 - DNS name:
 - Default Webapp:
 - Title:
 - Description:
 - Use as a template?
 - Preview Provider:

 At the bottom of the main panel, it says 'To continue click Next.'.

Step one: Web Project Details

Use some sample web project information to complete the step.

Name	Unique name of your web project. This is used as unique ID for your web project.
DNS name	DNS name for deployment. Usually it is your website DNS entry such as Cignex.com.
Default Webapp	Your web project can have a web application folder name. By default it is called as ROOT in WCM. You can have more than one web application in your web project. The details of creating more web applications are given in <i>Chapter 8, Managing Multiple Websites Using WCM</i> .
Title	Title of your web project.
Description	Brief description of the web project.
Use as a template?	This web project can also be used as a template project to create many such web projects with the click of a button.
Preview Provider	By default Alfresco provides Virtualization Server Preview. You can customize it to have your own preview server.

Step two: Create From Existing Web Project

Select the **Create a new empty Web Project** option and click on the **Next** button.

Step three: Configure Deployment Receivers

Click on **Add Deployment Receiver** to display the configuration details. Use the following information to complete the page. More information about the deployment receiver and the deployment process is explained in *Chapter 7, Content Delivery and Deployment*.

Type	Live Server
Host	localhost
Port	44100
Username	admin
Password	admin

Step four: Configure Web Forms

Web form is a very important functionality of WCM. We are going to skip this step as web forms are covered in detail in *Chapter 4, Web Content Production with Web Forms*. Click on the **Next** button to skip this step.

Step five: Configure Workflow

This step helps you to add and configure a new workflow for the web project. The workflow can also be associated with a web form (from the previous step).

Workflows are covered very extensively in this book in Chapters 5 and 6. Refer these chapters for understanding more about adding and configuring workflows. For this web project, do not select any workflow. Click on the **Next** button to skip this step.

Step six: Add Users

This step allows you to select the content managers for this web project from the list of available users.

Using the search feature provided, locate and select the user. Select **Content Manager** as the role for this user and click on **Add to List**. You can add as many users as you want to a specific web project:

Step Six - Add Users

Select users and their roles.

Specify Users/Groups

1. Select user/group and their role(s)

Users amit

Results for 'amit' in 'Users'. Clear Results

Amita Bhandari [amita]

Role

Content Manager
Content Publisher
Content Contributor
Content Reviewer

2.

Selected users/groups and their role(s)

Name

Amita Bhandari (Content Manager)

To continue click Next.

Name	Amita Bhandari (Content Manager)	<input type="button" value="Delete"/>
------	----------------------------------	---------------------------------------

Step seven: Email Users

Accept the default option **No** so that a notification e-mail is not sent.

Summary

Review the summary screen and click on the **Finish** button once you are fine with the information provided in the summary.

You can always add or modify the web project details at any given point of time.

The new web project appears in the **Web Projects** space. Completing the wizard automatically creates a source repository called a **Staging Sandbox**, and a set of development repositories called **User Sandboxes**. There is one User Sandbox for each user invited to work on the project, as well as an administrator sandbox.

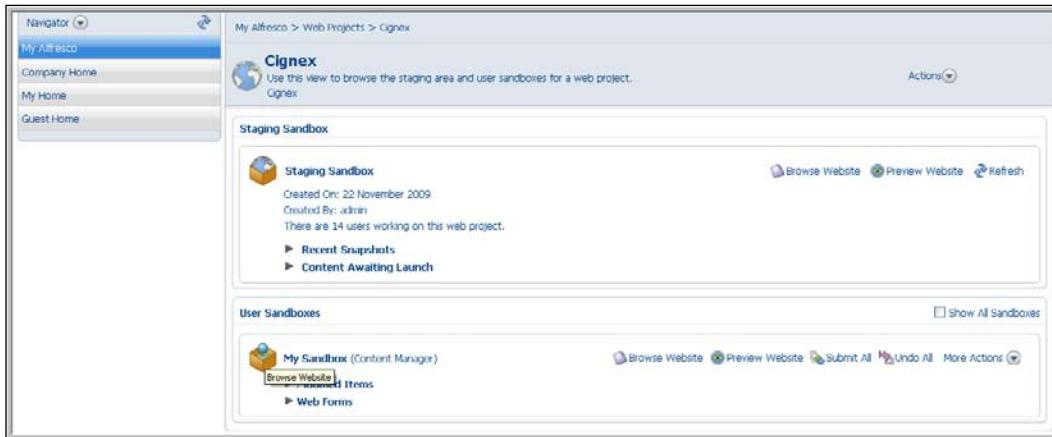
Creating a site easily with web project

In this section we shall see how we can easily create a website with a web project. For this we will use an existing ROOT folder, and import it into the web projects folder. The details of creating a website from scratch are given in the subsequent chapters of the book.

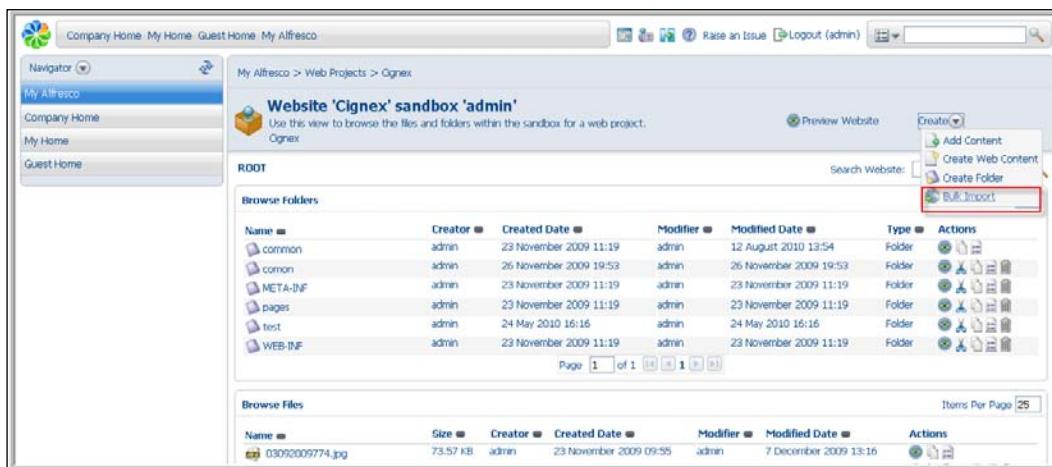
1. Create a new web project CIGNEX Technologies:



2. Import the .zip file of the existing ROOT folder into the newly created web project:

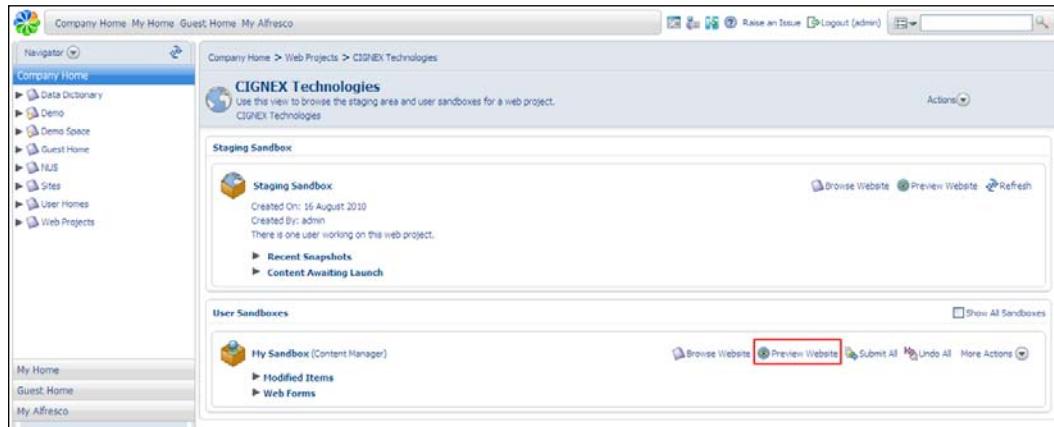


3. Click on the created website, **CIGNEX Technologies**.



Getting Started with Alfresco WCM

4. Click on **Preview Website** to view the website.



5. In the browser the website will appear as follows:



The code bundle for the corresponding files is available as a downloadable for this chapter.

Listing User Sandboxes

On clicking on the web project, the sandbox view appears displaying the Staging and User Sandboxes. Each User Sandbox contains the existing web project content. The sandbox labeled **My Sandbox** belongs to the currently logged in user (in this case, the administrator). As the administrator, you have access to the sandboxes of all users associated with the web project.

By default, only the Staging Sandbox and the User Sandbox display; however, you can select the **Show All Sandboxes** option in this view to display all the sandboxes available to you, as shown in the following screenshot:

Company Home > Web Projects > CIGNEX

CIGNEX
Use this view to browse the staging area and user sandboxes for a web project. Actions ▾

Staging Sandbox

Staging Sandbox
Created On: 20 February 2010
Created By: admin
There are 3 users working on this web project.
▶ Recent Snapshots
▶ Content Awaiting Launch

User Sandboxes

Show All Sandboxes

- My Sandbox (Content Manager)** Browse Website Preview Website Submit All Undo All More Actions ▾
 - ▶ Modified Items
 - ▶ Web Forms
- User: amita (Content Manager)** Browse Website Preview Website Submit All Undo All More Actions ▾
 - ▶ Modified Items
 - ▶ Web Forms
- User: munwar (Content Manager)** Browse Website Preview Website Submit All Undo All More Actions ▾
 - ▶ Modified Items
 - ▶ Web Forms

Refresh your browser page to view all the User Sandboxes if the browser is not refreshed automatically.

You cannot add the content directly to the Staging Sandbox. Here is the process you need to follow in order to create the content.

1. Create content in your User Sandbox either by uploading the files or using bulk import feature.
2. Preview the content to ensure the accuracy.
3. Optionally verify the broken links using the links checker.
4. Submit the approved content to the Staging Sandbox.
5. Deploy the content from Staging Sandbox to external servers using the filesystem deployer.

Add content to the web project

There are multiple methods of adding and creating content for a web project. In addition to creating web content within a project, you can also upload individual files from your computer or perform a bulk import of a ZIP file with web project contents as shown in the following screenshot:



When you install WCM, you will also get sample files to create web projects. Refer to the file `alfresco-sample-website.war`, which is provided as a default example in the Alfresco bundle, from `c:\alfresco\extras\wcm`.

To bulk import content of this file, select **Bulk Import** in the **Create** menu. Use the **Browse** button to locate and upload the `alfresco-sample-website.war` file. Once uploaded, click on **OK** to begin the import.

The `.war` file, which you imported into the current directory (the ROOT webapp), appears in expanded form in your User Sandbox.

If you don't have the sample `.war` file, then add few sample files to your User Sandbox from your local computer using the **Add Content** option. In the Alfresco WCM, a lock is automatically placed on content items created, imported, modified, or deleted, in order to prevent editing clashes. In our sandbox, notice that all of the items (not folders) currently display a padlock icon with a key (). This indicates that you own the lock and can perform actions on the content. A lock owned by another user appears as a plain padlock (). You can position the mouse cursor over the icon to display a tooltip indicating the lock owner.

Click on **Preview Website** in the sandbox header to see the website in its current state.

The preview window displays how the website or web application will look with the submitted User Sandbox content. Click on **About us** and then on **News** to display the empty **Alfresco Press Releases** page. You will come back to this again later once you have added some content.

Close the preview and return to User Sandbox. Click on **Project Name** in the breadcrumb path to return to the sandbox view.

Submit content to the Staging Sandbox

At this point, the web project is populated with content items, some imported and some created, some submitted and some not. You must now deal with the submission of the remaining items to staging.

In the User Sandbox, expand the **Modified Items** list to display the imported content.

The screenshot shows the 'Staging Sandbox' view for the 'CIGNEX' web project. In the 'User Sandboxes' section, 'My Sandbox (Content Manager)' is selected. Below it, the 'Modified Items' list is expanded, showing a table of imported files. The 'Actions' column contains icons for each item, including a red box around the 'Submit All' icon. A black arrow points from the text above to this 'Submit All' button.

Name	Created Date	Modified Date	Size	Actions
about	5 April 2010 23:44	5 April 2010 23:44		
accessibility	5 April 2010 23:45	5 April 2010 23:45		
assets	5 April 2010 23:45	5 April 2010 23:45		
customers	5 April 2010 23:45	5 April 2010 23:45		
document.gif	5 April 2010 23:45	5 April 2010 23:45	2.21 KB	
index.jsp	5 April 2010 23:45	5 April 2010 23:45	13.04 KB	

Click on **Submit All** and, on the **Submit Items** page, provide the information that will display as the name and description of the corresponding snapshot in staging:

Label: Initial Import

Description: Sample Website

The screenshot shows the 'Submit Items' page. In the 'Submission Info' section, 'Label' is set to 'Initial Import' and 'Description' is 'Sample Website'. The 'Workflow' section indicates 'No suitable workflows could be found for the modified item below'. The 'Content Expiration' section notes 'Set expiration date for all modified items.' The 'Modified Items' section lists the same files as the previous screenshot, with a note 'The following items will be submitted'. The 'Actions' column includes a red box around the 'Submit' icon. A black arrow points from the text above to this 'Submit' button.

Name	Description	Path	Modified Date	Expiration Date	Actions
about		/ROOT/about	5 April 2010 23:44		
accessibility		/ROOT/accessibility	5 April 2010 23:45		
assets		/ROOT/assets	5 April 2010 23:45		
customers		/ROOT/customers	5 April 2010 23:45		
document.gif		/ROOT/document.gif	5 April 2010 23:45		
index.jsp		/ROOT/index.jsp	5 April 2010 23:45		
legal		/ROOT/legal	5 April 2010 23:45		
login		/ROOT/login	5 April 2010 23:45		
media		/ROOT/media	5 April 2010 23:45		
META-INF		/ROOT/META-INF	5 April 2010 23:45		

Click on **OK**. The submission takes place in the background and each content item remains in the **Modified Items** list until its submission is complete.

Refresh the page as necessary until the **Modified Items** list is empty. In the Staging Sandbox, expand the **Recent Snapshots** list to view the snapshot you have created as shown in the following screenshot:

The screenshot shows the Alfresco Staging Sandbox interface. At the top, there's a header with 'Company Home > Web Projects > CIGNEX' and a 'Actions' dropdown. Below the header, there's a 'CIGNEX' section with a globe icon and a brief description: 'Use this view to browse the staging area and user sandboxes for a web project. Sample Alfresco Web Project to describe features.' On the right of this section is another 'Actions' dropdown. The main content area is titled 'Staging Sandbox'. It contains a 'Recent Snapshots' section which is highlighted with a red border. This section has a table with columns: Name, Description, Date, Submitted By, Version, Status, and Actions. One row is visible: 'Initial Import' with 'Sample Website' as the description, '5 April 2010 23:55' as the date, 'admin' as the submitted by, '3' as the version, and a status of 'Content Awaiting Launch'. There are also 'Browse Website', 'Preview Website', and 'Refresh' buttons above the table. Below the table, there's a link 'Content Awaiting Launch'. Further down, there's a 'User Sandboxes' section with a 'My Sandbox (Content Manager)' entry, 'Show All Sandboxes' link, and 'Actions' buttons. Underneath this, there are sections for 'Modified Items' (with a note 'No modified items') and 'Web Forms'.

Filesystem projection

The AVM repository and its virtual stores are made available by projection through CIFS.

To mount the CIFS directory in Windows Server/XP, follow these steps:

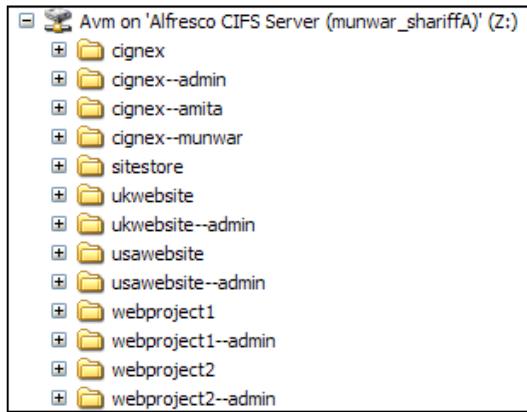
1. Open up Windows Explorer.
2. Under the **Tools** menu, **Select Map Network Drive....**
3. Select an available drive letter. Example: **Z:**

4. For **Folder** enter the CIFS server name. Example:

`\\A\AVM`



Windows will open the mapped drive and you should see something similar to the following that includes the current web projects:



AVM CIFS projection provides:

- Access to all available sandboxes (Read/Write)
- Access to all available layers (Preview)
- Access to all snapshots in read-only mode under top-level Versions Directory
- Access to all file metadata under top-level METADATA directory
- Mounting of AVM virtual store "docroots" to test servers

Bulk loading of content XML files and previously rendered files (HTML, XML, and RSS feeds) will not recreate mapping between source file -> rendering template -> output file. This is also the case for bulk uploads into a web project.

There is a Web Projects tool on the Alfresco Forge, but it is for Version 2.2. These tools provided import and export features, and re-association of renditions with content and templates, among others.

Virtualization server

A Virtualization server is a customized version of Tomcat that treats each User Sandbox, Staging Sandbox, and workflow instance as a separate Tomcat web application. It understands an "overloaded" hostname format that allows it to map requests to specific sandboxes. These "overloaded" hostnames use wildcard DNS entries to map all request to the same Alfresco Virtualization server.

Configuring the virtual server for preview

The configuration files for the virtual server can be found in the <alfresco home>/virtual-tomcat/conf directory. To enable previews on a development machine, you need to modify the alfrescovirtserver.properties file to utilize the Alfresco wildcard DNS service. The alfresco.virtserv.domain value needs to be updated to replace 127-0-0-1 with the IP of the Alfresco local machine, substituting all ":" with dashes:

```
# Alfresco's free/public wildcard domain service ("EchoDNS")
# returns resource records with a TTL of 1 day. Therefore,
# the run-time overhead is quite minimal. Once the machine
# has done a name lookup, it will cache the result for the
# next 24 hours (or until you reboot).
#
alfresco.virtserver.domain=127-0-0-1.ip.alfrescodemo.net
```

If the IP of the Alfresco machine is 192.168.1.25, update alfresco.virtserver.domain to alfresco.virtserver.domain=192-168-1-25.ip.alfrescodemo.net.

Alfresco Virtual Server needs to be restarted at this point.

Virtualization URL format

The general format for Alfresco virtual hyperlink is:

```
http://virtual-hostname.www--sandbox.virtualization-domain:port/
request-path
```

where the `virtual-hostname` is defined as the DNS name when creating a web project, and the `virtualization-domain` is the wildcarded domain name entry that maps to the Alfresco Virtualization Server. The default port for a virtualization server is 8180. Given that a web project was created with "test" as the DNS name, (`*.virtualserver.unico.com` is a DNS entry for Virtualization Server) and is installed on the default port, the following are true:

- `http://unicom.www--sandbox.virtualserver.unico.com:8180/` is the preview of the Unicorn web project for the Staging Sandbox.
- `http://admin.unicom.www--sandbox.virtualserver.unico.com:8180/` is the preview of the root of the Unicorn web project for the admin user's sandbox.
- `http://jsmith.unicom.www--sandbox.virtualserver.unico.com:8180/index.html` is the preview of the file `/index.html` of the web project in the `jsmith` user's sandbox. There are variations of this for previewing new content before the initial persistence and previews for items within a workflow.

Virtualization server access to the User Sandbox

The AVM repository and its virtual stores are mapped into the virtual Tomcat working directory:

```
$VIRTUAL_TOMCAT_HOME/work/Catalina/avm.alfresco.localhost/.
```

This working directory will contain directories like the following, which are similar to the naming CIFS convention for virtual stores:

```
TABLE [  
$-1$mysite--admin--preview$ROOT/ $-1$silly--admin--preview$ROOT/  
$-1$mysite--admin$ROOT/ $-1$silly--admin$ROOT/  
$-1$mysite--alice--preview$ROOT/ $-1$silly--alice--preview$ROOT/  
$-1$mysite--alice$ROOT/ $-1$silly--alice$ROOT/  
$-1$mysite--bob--preview$ROOT/ $-1$silly--preview$ROOT/  
$-1$mysite--bob$ROOT/ $-1$silly--$ROOT/  
$-1$mysite--preview$ROOT/ host-manager/  
$-1$mysite$ROOT/ manager/  
$42$mysite$ROOT/]
```

Virtual server configuration

Configure the virtual server to allow the previewing of content and review some of the content in the CIGNEX web project in the user's sandbox. Once you are satisfied that the preview is working, submit the whole project to staging. After the submission is complete, preview the website in the Staging Sandbox.

1. Determine the current IP of the Alfresco machine using one of the following:
 - Windows – open CMD prompt and run the ipconfig command
 - Unix(s) – open terminal and run ipconfig -a
2. Open the `virtserver.properties` file found in the `<alfresco home>/virtual-tomcat/conf` directory.
`c:\Alfresco\virtual-tomcat\conf\virtserver.properties`
3. Modify the following line to use the IP found in step one, replacing all `"."` with `"-"`:
`alfresco.virtserver.domain=127-0-0-1.ip.alfrescodemo.net`
For IP of 192.168.1.100 the line would be:
`alfresco.virtserver.domain=192-168-1-100.ip.alfrescodemo.net`
4. Save and exit the file.
5. Start or restart the Virtual Server.
6. Test previewing the file in the user's sandbox.

Click on **Submit All** to send all files to the staging environment. Test preview in the Staging Sandbox. Previewing in the User and Staging Sandboxes should work.

Dynamic websites using WCM

A web project is not limited to just static files. Web projects can include server-side scripting files including JSP, PHP, CF, or Python files. In the end, these files would be published to the appropriate application server or web server enabled to evaluate these files. Specifically for Java, Alfresco supports the evaluation of JSPs and exploded WARs within a web project with support for preview of JSPs by the Virtual Server.

You could upload archived WARs, have Alfresco deploy them to a web application server for unpacking, and so on. For Tomcat, it could place them in the `webapps` directory.

Virtual server JSP support

Create a simple "hello world" JSP, load it up, and test. The JSP will evaluate without creating the WEB-INF and web.xml files. But more complex WARs will need WEB-INF and web.xml files for Tomcat to evaluate.

Previewing WARs and getRealPath()

For JSP and Java code that use the `getRealPath()` method or the filesystem path for a file, Alfresco requires additional configuration for the virtual server preview to function. This is the case for Spring-based web applications, where configuration and properties files are required to be read on startup.

To enable preview in this case, Alfresco needs to be modified to:

- Enable Alfresco to use CIFS (refer *Chapter 2* for details)
- Mount the CIFS directory
- Have the virtual server mount the AVM CIFS projection

Virtual server configuration

The virtual server needs to be configured to mount the new drive. Follow these steps:

1. Open up the `alfresco-virtserver.properties` file in the directory:
`c:\Alfresco\virtualtomcat\conf.`
2. Locate the following section:

```
#  
# On Windows, the ".unix" properties are ignored.  
# On Linux, the ".win" properties are ignored.  
#  
alfresco.virtserver.cifs.avm.versiontree.win=v  
alfresco.virtserver.cifs.avm.versiontree.unix=/media/alfresco/  
cifs/v
```
3. Modify `alfresco.virtserver.cifs.avm.versiontree.win=v` to use the drive letter "Z" used above when mapping to the CIFS server, as follows:
`alfresco.virtserver.cifs.avm.versiontree.win=z..`

Search

Searching against a WCM store is the same as searching against a full repository. However, the scope of the search will be limited to that web project only.

A search against an AVM store can be performed via Java, JavaScript, FreeMarker template, or the node browser. It is not yet exposed via open search. As the API is the same, searches via web services or something similar are the same apart from the store context.

Search is only available for the latest snapshot of the head revision for staging. It is not available for User Sandboxes, workflow sandboxes, and so on. In a lot of cases, the design principle followed is to use Lucene queries minimally in your project. Other methods such as navigating via AVM Paths and others to get hold of the content you are trying to search can suffice in such a scenario.

Summary

Alfresco tracks all content modifications made within a user's sandbox and maintains those changes in isolation from other user's working within their own sandbox environment. In this way, large teams of users can work independently on changes to the website without stepping over one another's work.

Sandboxed development, along with virtualization and in-context preview, means that a large, diverse web team can easily collaborate on changes to the website with reduced risk and higher overall quality.

The next chapter will explain in detail the concepts we have covered in this chapter. Creation of web forms from scratch, web projects, and content creation using the same is explained in great detail. It will take you through concepts of templating, associating templates with web projects, dashlets, dynamic content, and so on.

4

Web Content Production with Web Forms

In the previous chapter, we learned about creation of a web project. To set up a website you must have both web forms and a web project. Web forms provide the facility to manage content through a user-friendly and technology-neutral (as much as possible) interface. It allows you to separate content from code and presentation logic. It allows non-technical content owners to manage, approve, and deploy their own web content. Static content is required for a few websites. However, for many sites, dynamic and flashy content is required. Dynamic content can be managed easily with web forms, as content can be created once and rendered in many formats. In this chapter, you will learn about basic and advanced web form concepts and the ways to extend it as per your business requirements.

By the end of this chapter, you will have learned how to:

- Create web forms
- Create rendition templates
- Create FreeMarker templates
- Create Extensible Stylesheet Language transformations
- Associate web forms and renditions for specific or multiple projects
- Update web forms and rendition templates
- Create dynamic content
- Create web publishing dashlets

Why web forms

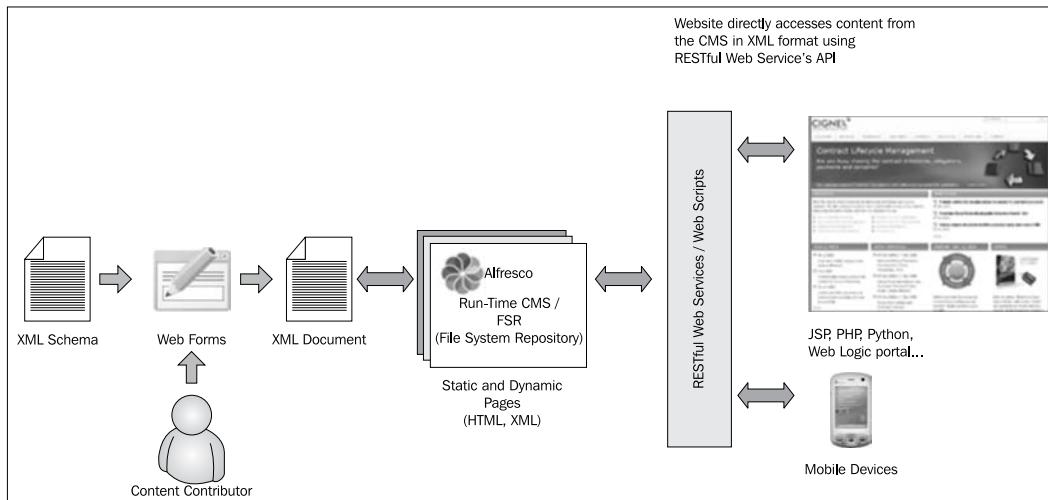
Consider a website, say, www.cignex.com, which has some primary sections, such as news, blogs, events, trainings, and solutions. The news, blog, and training elements have to be updated every month. Let's assume that in the structure of news element there should be a title, news headline, sub headline, page content, image, image title, and news date. For each content of the news, you must have these elements. To update these sections, each time you have to create new HTML/JSP files from scratch. This becomes quite clumsy. We need a way to manage it easily. Assume the concept where the structure is in place and you have to just put in the content. The concept can be called web forms. Another important thing to focus on is the fact that web forms are always stored in XML (Extensible Markup Language) format. Using web scripts, you can extract the web content (which is in XML format) and serve external applications. Thus you can consider the user interface to be in various technologies that will enable content to be used for various purposes wherever required. Content can be transactionally deployed to static content servers or Alfresco runtime repositories, providing complete architectural flexibility for web forms.

These enhanced capabilities make it easier for authorized users to preview and edit various file types, including those with multi-language content, before publishing. The new capabilities would allow contributors to preview pages rendered with different file formats, including HTML, PDF, RTF, XML, and mobile, as well as any UTF-8 compatible foreign language formats.

Following are the benefits of web forms:

- **Easily extensible solutions:** This will allow users to separate content from code and presentation logic. Thus, content contributors can easily manage the website without developer intervention.
- You can display the content in various formats.
- **Easy integration of external systems:** Web forms are stored in XML format, which gives the advantage of platform and language-independent technology, thus making the content available to any web technology (PHP, Python, J2EE, AJAX, Flash, Cold Fusion, and so on).
- **Asset reusability:** Static and dynamic include of the content makes the structure simpler.
- It provides a rapid learning curve. Developing and maintaining Alfresco web forms require basic skills in XML, XSD, XSLT, and FreeMarker.

The following diagram shows how to access content from the Alfresco repository to an external application:



Introduction to web forms

Web forms are also known as WCM forms. Web forms are used to capture content from the user and store it as **Extensible Markup Language (XML)** in Alfresco. The XML content is created based on an **XML Schema Definition (XSD)**. Web forms can be rendered as user-friendly web-based forms. Web forms can be associated with multiple rendition templates to render web content in various formats, such as plain text, HTML, JSP, and PDF. Out of the box, it supports three types of rendition engines: **Extensible Stylesheet Language Transformations (XSLT)**, **FreeMarker**, and **Extensible Stylesheet Language Formatting Objects (XSL-FO)**. Web forms are stored in the **Web Client Extension** space within **Company Home | Data Dictionary | Web Forms**. Web forms can be configured to any number of projects. As web forms are located in the Alfresco space, they can be accessed by the default CIFS, FTP, and WebDAV interfaces.

Creating web forms

Let's take the same example as mentioned previously, of the Cignex website. We will create web forms for the blogs and news sections. Also, we will associate these web forms with the rendition templates. Finally, we will create content using web forms and preview it.

The process to define and create web forms in Alfresco is as follows:

1. Identify structure to be used for each web form.
2. Define your XSD file.
3. Create a web form using XSD.

Identifying the structure to be used for each web form

To create a web form you are required to create an XML Schema Definition (XSD) file with an extension as .xsd. XSD specifies how to formally describe the elements in an XML document. Before creating an XSD file you will need to decide what all user interface components you want to display in a web form. To create a schema for a web form, you analyze its structure defining each structural element as you encounter it. As mentioned previously, we will create web forms for the news and blogs sections. Consider that some of the elements that are required in creating an XSD file for news are: Brief Title, News Page Headline, News Page Sub Headline, News Page Content, News Article Graphic, Image Caption, Image Title, and News Date. Now that you have decided the elements, you will learn how to create an XSD file for simple user interface controls. You will know more about advanced controls in the short while.

Defining a schema

To create an XSD file we have to start with the standard XML declaration followed by the xs:schema element that defines a schema:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:trn="http://www.alfrescobook.com/webforms"
  xmlns:alf="http://www.alfresco.org"
  targetNamespace="http://www.alfrescobook.com/webforms"
  elementFormDefault="qualified"
  >
</xs:schema>
```

In the previous schema, we use the standard namespace (xs), and the URI associated with this namespace is the schema language definition, which has the standard value of <http://www.w3.org/2001/XMLSchema>. It is important to specify a namespace (xmlns:trn) for your web forms to eliminate collisions. You can have the same namespace for a specific development project. We will use this element while generating renditions. The alf namespace is added to enable label, alert, and appearance elements while writing an XSD.

Defining a complex element

Now we can define elements inside a complex element. First we have to define a root element. Let's define news as a root element. This element has an attribute and it contains other elements, therefore we consider it as a complex type. The child elements of the news element are surrounded by an xs:sequence element that defines an ordered sequence of sub-elements. The syntax for defining a complex element is:

```
<xs:element name="news">
  <xs:complexType>
    <xs:sequence>
      ...
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Defining a simple element

We can also define a few simple elements inside the complex element. Simple elements are those elements that do not contain any attribute. Each element is defined with a name and type. Let's define a few simple elements such as shortTitle, contentHeader, contentSubHeader, contentText, and so on.

The syntax for defining a simple element is:

```
<xs:element name="shortTitle" type="xs:normalizedString" />
```

Here shortTitle is the name of the element and normalizedString is the data type of the element. Following is an example of simple elements defining inside complex elements.

```
<xs:element name="news">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="shortTitle" type="xs:normalizedString" />
      <xs:element name="contentHeader" type="xs:normalizedString" />
      <xs:element name="contentSubHeader" type="xs:string" />
      <xs:element name="contentText" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

An XML schema has a lot of built-in data types. Each of these data types has a default widget rendering within the generated XForm in Alfresco.

The following table shows the associated widgets with the most commonly used XSD data types:

XSD Data Type	User Interface Widgets
xs:normalizedString	Textbox
xs:string, xs:anyType	WYSIWYG editor, Text area
xs:date	Calendar control
xs:time	Time picker
xs:anyURI	Asset picker
xs:boolean	Checkbox
xs:integer, xs:float, xs:double, xs:decimal	Slider control
xs:enumeration	Radio button, Drop down

Now we will learn a few attributes that can be applied to an element in order to make it mandatory, optional, default value, or fixed value.

Default and fixed values for elements

An attribute may have a default or fixed value. A default value is automatically assigned to the element and you can specify any other value. A fixed value is also automatically assigned to the element, but you cannot specify another value. This can be considered as a read-only value. Following is the code snippet for defining default value.

```
<xss:element name="contentHeader" type="xs:normalizedString"  
default="News Full Headline">
```

Optional and required values for elements

We can define the number of possible occurrences for an element with the `maxOccurs` and `minOccurs` attributes. `maxOccurs` specifies the maximum number of occurrences for an element and `minOccurs` specifies the minimum number. The default value for both `maxOccurs` and `minOccurs` is 1, which means the field is mandatory.

To set the element as optional, you have to set the value of `minOccurs` to 0 as follows:

```
<xss:element name="contentSubHeader" type="xs:string" minOccurs="0"  
maxOccurs="1">
```

The element can appear multiple times by setting the `maxOccurs` attribute of the element to "unbounded", which means that there can be as many occurrences of the element as the content contributor wishes. We can also fix the occurrences of the element by specifying any number.

Advanced schema attributes

There are some advanced schemas attributes provided that will enhance the capabilities provided within a form. For instance, you may want to display images in the web form. You may also think of validation in the web forms that will validate your content. For this, you are required to customize some of the widgets in web forms.

File pickers

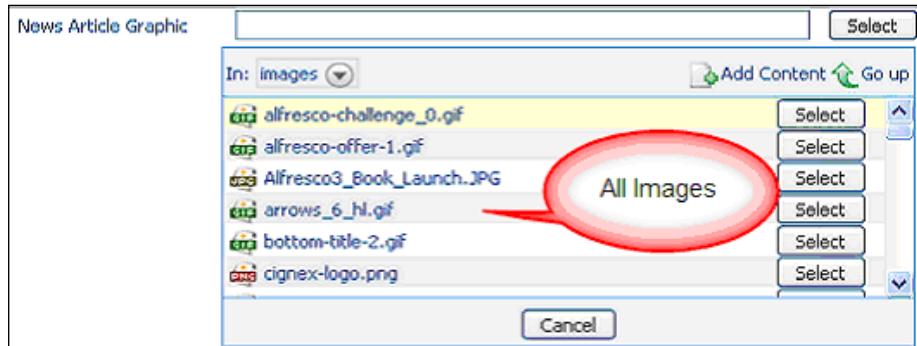
Alfresco uses the `xs:anyURI` data type to overcome limitations such as inter-document relationships. Although XML standards do exist for defining relationships (XPointer and XLink), both are stagnant (no updates since 2001) and neither has achieved widespread adoption:

```
<xs:element name="contentGraphic" type="xs:anyURI" />
```

This results in an "asset picker" widget, which will allow the content contributor to browse and select content from the web project. The following code snippet provides the ability to specify an image picker to get the images from any folder in the web project:

```
<xs:element name="contentGraphic" type="xs:anyURI" minOccurs="0"
  maxOccurs="1"
  <xs:annotation>
    <xs:appinfo>
      <alf:label>News Article Graphic</alf:label>
      <alf:appearance>image_file_picker</alf:appearance>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

On implementation of this code snippet, a **Select** button (shown in the following screenshot) will be displayed for each of the images:



We also have a few other preconfigured appearance options defined in the `web-client-config-wcm.xml` file, which is placed under `<install-alfresco>/tomcat/webapps/alfresco/WEB-INF/classes/alfresco/`.

Here are some examples of preconfigured appearances from the `web-client-config-wcm.xml` file:

Appearance	Description
<code>folder_restricted_file_picker</code>	Picks files from a specific folder
<code>search_restricted_file_picker</code>	Searches files from a specific folder
<code>image_file_picker</code>	Uploads images
<code>html_file_picker</code>	Uploads HTML files
<code>folder_picker</code>	Uploads folders
<code>file_picker</code>	Uploads any file

If you want to upload HTML files, you have to put `html_file_picker` inside the `<alf:appearance>` tag. For files and folders you can have `file_picker` and `folder_picker` respectively.

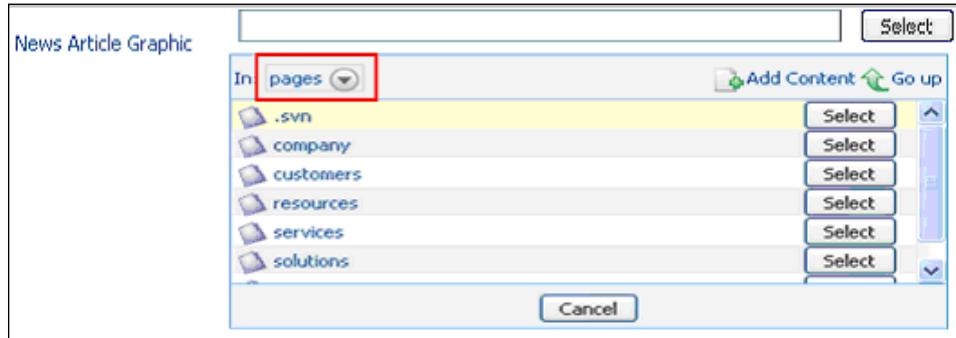
Suppose you want to provide restrictions to file pickers and be relative to the root of the web project folder. Thus, the contents of the specified folder will be available for selecting in the file picker. This can be achieved by uncommenting the section in the code below (which is highlighted in bold) in the same web-client-config-wcm.xml file:

```
<widget xforms-type="xf:upload"
    appearance="folder_restricted_file_picker"
    javascript-class-name="alfresco.xforms.FilePicker">
<param name="selectable_types">wcm:avmcontent,wcm:avmfolder</param>
<param name="folder_restriction">/common/pages</param>
</widget>
```

Define this widget in the XSD file, as follows:

```
<xs:element name="contentGraphic" type="xs:anyURI" minOccurs="0"
    maxOccurs="1"
<xs:annotation>
<xs:appinfo>
    <alf:label>News Article Graphic</alf:label>
    <alf:appearance>folder_restricted_file_picker</alf:appearance>
</xs:appinfo>
</xs:annotation>
</xs:element>
```

This customized file picker will search for only folders inside common/pages folder as shown in the following screenshot:



Another important feature, which will make search faster and time saving, is the reusability of search. You don't have to browse every time for the files.

The config_search_name parameter must be set to the name of a configured search. The file picker contents will be restricted to the results of this named configured search. The configured search must be stored in the Public Saved Searches folder.

The search is Lucene based and it will only query content of the Staging Sandbox. The public saved search is actually stored as an XML document in the Alfresco repository. Therefore, we can modify it with any Lucene query that is supported by Alfresco WCM search. The following code snippet will search for "training" as a text in all of the files of the web project submitted to the staging server:

```
<widget xforms-type="xf:upload"
    appearance="search_restricted_file_picker"
    javascript-class-name="alfresco.xforms.FilePicker">
<param name="config_search_name">training</param>
</widget>
```

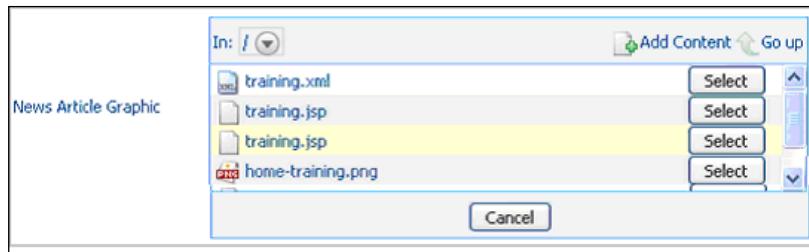
Define this widget in the XSD file, as follows:

```
<xss:element name="contentGraphic" type="xs:anyURI" minOccurs="0"
    maxOccurs="1"
<xss:annotation>
<xss:appinfo>
    <alf:label>News Article Graphic</alf:label>
    <alf:appearance>search_restricted_file_picker</alf:appearance>
</xss:appinfo>
</xss:annotation>
</xss:element>
```

The following screenshot shows how to save 'config_search_name' text in a public search folder so that this can be used while searching in the web project:

The screenshot shows the Alfresco search interface. At the top, there is a search bar with the query "training". Below it, a "Search Results" section displays a single result: "training related content". To the right of the search results, there is a "Save New Search" dialog box. The "Name" field is set to "training", and the "Description" field is "training related content". A checked checkbox says "Save as a public search available to all users." A red box highlights the "Save" button. A red callout points to the "Save" button with the text "Click to save search in Public search folder". Another red callout points to the "Save" button in the dialog with the text "Lucene Query Created".

The file picker will search all files having the word "training" within its name. The following screenshot shows the search result:



You can also customize file pickers. You have to define all customizations in `web-client-config-wcm.xml`. Using the following code snippet you will have the option of selecting only JPEG files from the web project. The `selectable_types` parameter must be set to restrict which types of content are selectable in the widget.

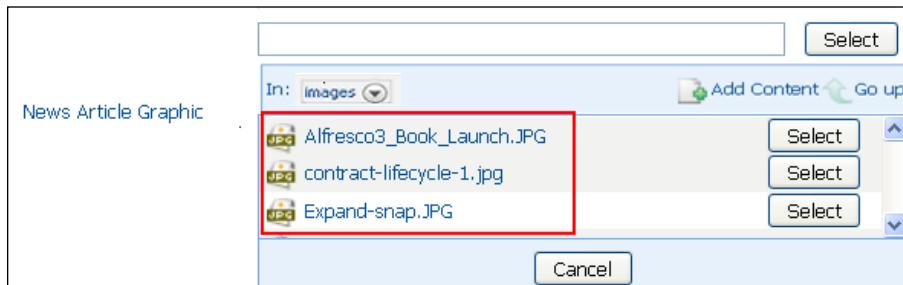
The `filter_mimetypes` parameter can be used to filter selectable files by MIME types:

```
<widget xforms-type="xf:upload"
    appearance="custom_image_file_picker"
    javascript-class-name="alfresco.xforms.FilePicker">
<param name="selectable_types">wcm:avmcontent</param>
<param name="filter_mimetypes">image/jpeg</param>
</widget>
```

Define this widget in the XSD file, as follows:

```
<xss:element name="image" type="xs:anyURI" minOccurs="0" maxOccurs="1">
<xss:annotation>
<xss:appinfo>
    <alf:appearance>custom_image_file_picker</alf:appearance>
</xss:appinfo>
</xss:annotation>
</xss:element>
```

Since we have customized file picker, in the following screenshot you will find only JPEG files:



With the previous code snippet, we have customized `image_file_picker`. In this manner, you can customize other file pickers also.



Please note all of the customization related to file pickers has to be provided in the specified file—`<install-alfresco>/tomcat/webapps/alfresco/WEB-INF/classes/alfresco/web-client-config-wcm.xml`.

Tool tips and labels

You can also provide a label for an element specified in your XSD file. By default, the label created by the form processor for each element is simply the element's name. You could also get a tool tip for the label you have provided. Hence, the label becomes News Page Sub-Headline instead of contentSubHeader. Whenever you drag a mouse over the label you will get help for the label:

```
<xs:element name="contentSubHeader" type="xs:string" minOccurs="0"
maxOccurs="1">
<xs:annotation>
<xs:appinfo>
<alf:label>News Page Sub-Headline</alf:label>
<alf:hint>Please enter the news page sub-headline if the news has a
sub-headline.</alf:hint>
</xs:appinfo>
</xs:annotation>
</xs:element>
```



Localization

You can also localize the label or alert by specifying a message bundle key in the annotation:

```
<xs:element name="emailId" type="xs:string">
<xs:annotation>
<xs:appinfo>
<alf:label>${email}</alf:label>
</xs:appinfo>
</xs:annotation>
</xs:element>
```

This can be achieved by creating the following message bundle file. Create a property file with name `strings.properties` in the **Company Home | Data Dictionary | Web Forms | <form space name> | strings.properties** with the following code:

```
email = Please enter a proper Email Id .
```

The property files can also be internationalized by having different versions for language or locale combination. The default configured property file locations from highest to lowest precedence are:

1. **Company Home | Data Dictionary | Web Forms | <form space name> | strings.properties.**
2. **Company Home | Data Dictionary | Web Forms | strings.properties.**
3. **webclient.properties placed in <install-alfresco>/tomcat/webapps/alfresco/WEB-INF/classes/alfresco/messages/.**

Validation

You can provide validations to any of the fields. For instance, if you provide the correct order ID, the element will be highlighted in red. This indicates that the value of the element must be a string, it must be exactly six characters in a row, and those characters must be a number from 0 to 9:

```
<xs:element name="orderId">
<xs:simpleType>
<xs:restriction base="xs:string">
    <xs:pattern value="[0-9]{6}" />
</xs:restriction>
</xs:simpleType>
</xs:element>
```

Customizing the WYSIWYG editor

You can customize the WYSIWYG editor easily. The WYSIWYG editor is the TinyMCE editor. When we define a simple element of type string, we can see Rich Text Editor with limited functionalities. We can also have many other features available with Rich Text Editor. See the following code for the advanced editor. With this we can have features such as alignment, font family, and font size:

```
<xs:element name="contentSubHeader" type="xs:string" minOccurs="1"
maxOccurs="1">
<xs:annotation>
<xs:appinfo>
<alf:label>News Page Sub-Headline</alf:label>
<alf:hint>Please enter the news article sub-headline if the article
has a sub-headline.</alf:hint>
<alf:appearance>full</alf:appearance>
</xs:appinfo>
</xs:annotation>
</xs:element>
```

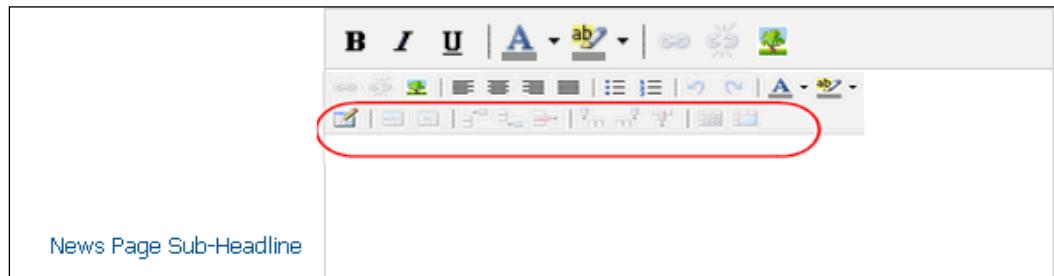


If you want more advanced features for the editor, such as table, you can use the following code in the XSD file:

```

<xs:element name="contentSubHeader" type="xs:string" minOccurs="1"
maxOccurs="1">
<xs:annotation>
<xs:appinfo>
<alf:label>News Page Sub-Headline</alf:label>
<alf:hint>Please enter the news article sub-headline if the
article has a sub-headline.</alf:hint>
<alf:appearance>custom</alf:appearance>
</xs:appinfo>
</xs:annotation>
</xs:element>

```



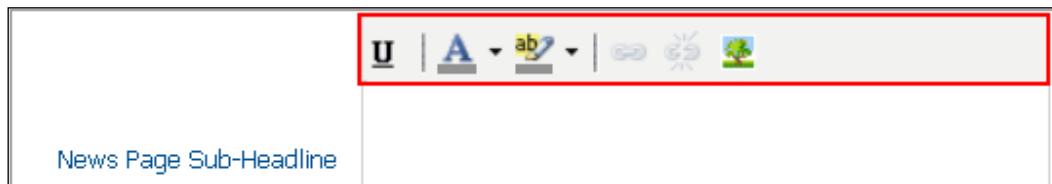
Suppose you want to customize these editors. You want to hide bold, italic feature and reduce the height of the editor. The customization code has to be placed in the <install-alfresco>/tomcat/webapps/alfresco/WEB-INF/classes/alfresco/web-client-config-wcm.xml file.

Place the following code:

```
<widget xforms-type="xf:textarea"
    appearance="customnews"
    javascript-class-name="alfresco.xforms.RichTextEditor">
    <param name="theme_advanced_buttons1">underline,separator,
        forecolor,backcolor, separator,link,unlink,image</param>
    <param name="height">100</param>
</widget>
```

Place this code in the XSD file:

```
<xss:element name="contentSubHeader" type="xs:string" minOccurs="1"
    maxOccurs="1">
    <xss:annotation>
        <xss:appinfo>
            <alf:label>News Page Sub-Headline</alf:label>
            <alf:hint>Please enter the news page sub-headline if the news
                has a sub-headline.</alf:hint>
            <alf:appearance>customnews</alf:appearance>
        </xss:appinfo>
    </xss:annotation>
</xss:element>
```



Another customization that will allow you to see the source code of the HTML and insert media files in the Rich Text Editor if required is as follows. Open the web-client-config-wcm.xml file. Replace the code with the following highlighted code:

```
<widget xforms-type="xf:textarea" appearance="custom"
    javascript-class-name="alfresco.xforms.RichTextEditor">
    <param name="theme_advanced_buttons1">bold,italic,underline,
        strikethrough,separator,fontselect,
        fontsizeselect,code</param>
    <param name="theme_advanced_buttons2">link,unlink,image,
        separator,justifyleft,justifycenter,justifyright,
        justifyfull,separator,bullist,numlist,separator,
        undo,redo,separator,forecolor,backcolor</param>
    <param name="height">600</param>
    <param name="mode">exact</param>
```

```

<param name="force_p_newlines">true</param>
<param name="apply_source_formatting">true</param>
<param name="plugins">table,paste,media</param>
<param name="theme_advanced_buttons3">tablecontrols,media</
    param>
</widget>

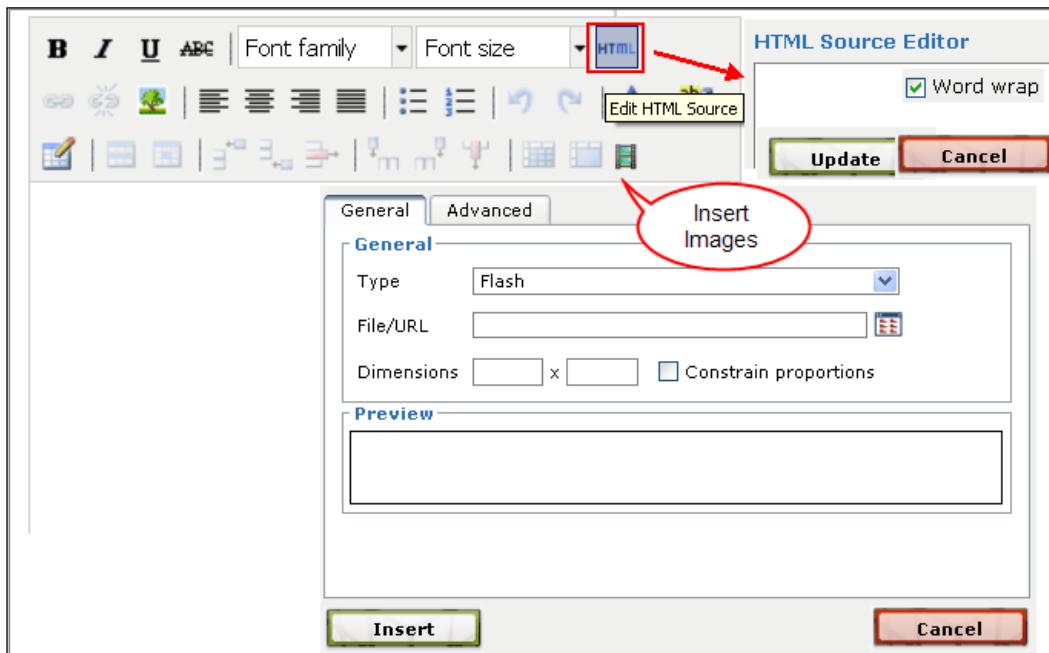
```

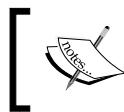
Place the following code in the XSD file:

```

<xss:element name="contentSubHeader" type="xs:string" minOccurs="1"
    maxOccurs="1">
    <xss:annotation>
        <xss:appinfo>
            <alf:label>News Page Sub-Headline</alf:label>
            <alf:hint>Please enter the news pagesub-headline if the page
                has a sub-headline.</alf:hint>
            <alf:appearance>custom</alf:appearance>
        </xss:appinfo>
    </xss:annotation>
</xss:element>

```





We can also easily install additional plugins by putting them in the <install-alfresco>/tomcat/webapps/alfresco/scripts/tiny_mce/plugins folder.

Dynamically populating lists (in drop-down lists) or conditional drop downs

A drop-down list can be handled with the xs:enumeration element within the xs:restriction element that defines the type of enumeration. For example, all the values in the list should be string, as follows:

```
<xs:element name="disclaimer" minOccurs="0" maxOccurs="1">
<xs:annotation>
<xs:appinfo>
<alf:label>Disclaimer</alf:label>
<alf:appearance>minimal</alf:appearance>
</xs:appinfo>
</xs:annotation>
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="news" />
<xs:enumeration value="blogs" />
</xs:restriction>
</xs:simpleType>
</xs:element>
```

Reuse of common XSD and JSP

Use of includes will enable you to reuse the common structure across multiple files. You can also call a web script in include schemaLocation. Please note: for this, the virtual server should be running. We will see example of includes in the later sections.

```
<xs:include schemaLocation="/common/inc/contentType.xsd" />

<xs:include schemaLocation="/common/inc/contentType.jsp" />

<xs:include schemaLocation="webscript://path/to/my/
webscript?storeid={storeid}" />

<xs:import namespace="imported_namespace" schemaLocation="webscript://
path/with/the/{storeid}/embedded?ticket={ticket}" />
```



For more information about XSD, refer to the website <http://www.w3.org/XML/Schema>.



In order to create the XSD file, follow these steps:

1. Navigate to <installed-alfresco>/extras. Create folder wcm/forms.
2. Create a file named news.xsd in the above-specified path and populate it with the downloaded code from Packt's website.
3. Create a file named blogs.xsd in the above-specified path and populate it with the downloaded code from Packt's website.
4. Create a file named training.xsd in the above-specified path and populate it with the downloaded code from Packt's website.



Download the complete code samples from the Packt website.



The XSD file is now ready to use. The next task is to create web forms. For this you first have to create a web project. This section uses the web project that you have already created as a part of your Cignex sample application in *Chapter 3*. As a part of the sample application, you will manage content in the Cignex web project. For more information about the web project and sample application, refer to *Chapter 3*.

Create a web form in Alfresco

You can create a web form using two ways:

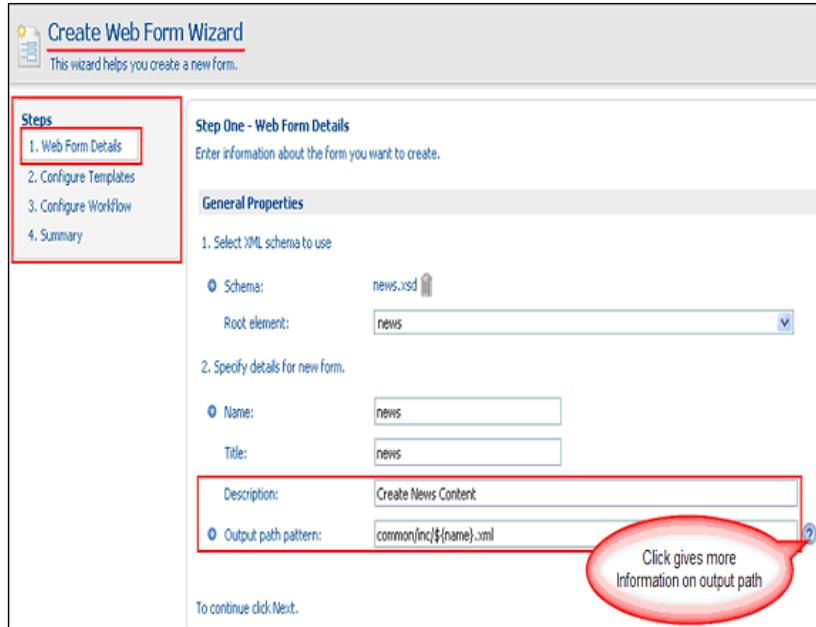
- Directly from **Company Home | Data Dictionary | Web Forms** space (**Create Web Form**). Using this first option we can only create web forms; it is not associated to any project.
- Using **Edit Web Project setting** wizard in a web project (assuming you have already created a web project). It gives the flexibility to associate a web form with the web project.

The process to create web forms is as follows:

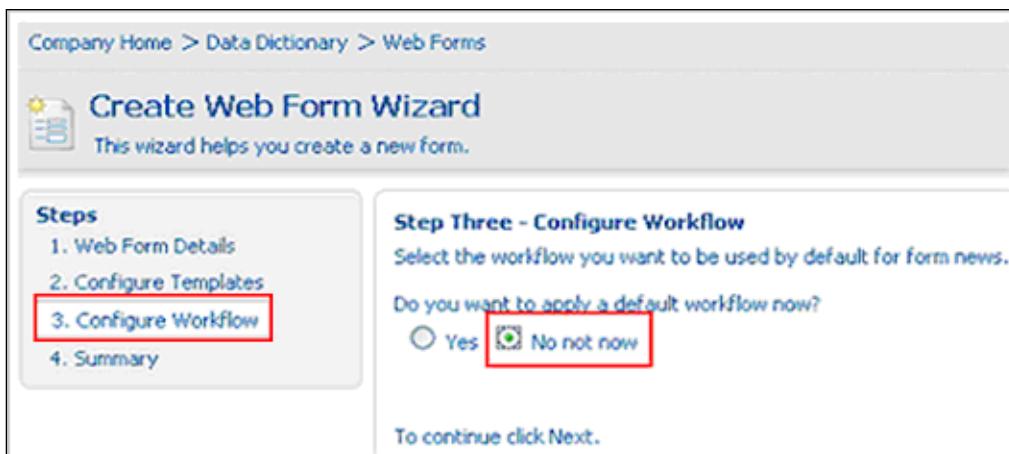
1. Ensure that the Alfresco Server is up and running.
2. Go to **Company Home | Data Dictionary | Web Forms**.



3. Clicking on the **Create Web Form** link opens the **Create Web Form Wizard** window. This is the **Create Web Form** pane, as you can see from the list of steps at the left of the pane. You may also notice that the wizard window has four steps, as shown in the following screenshot. The first step is to upload the XSD file, the second step is to configure template for rendition output, the third step is to define the workflow for form data, and the fourth step is to confirm the web form.
4. In the **Step One** window, you will notice a **Browse** button to upload the XSD file. Click on **Browse** to locate and upload the news.xsd file created in the folder <installed-alfresco>/extras/wcm/forms. After uploading, it will automatically fill up the values like **Root element**, **Name**, **Title**, and **Output path pattern**. You can change the output path pattern. This means the XML file will be created in the specified path. Take a look at the following screenshot and change the **Output path pattern** and **Description** as specified. Then click on **Next**.



5. In the **Step Two** window, the **Configure Template** page is shown. This step is optional. Leave all values blank and click on **Next**. (This step allows you to generate forms in various outputs. We will examine this in the next section of this chapter.)
6. In the **Step Three** window, the **Configure Workflow** page is shown. Select the **No not now** option as shown in the following screenshot (this step allows the creation of default workflows for form data. You will know more about this in the next chapter):



7. Clicking on the **Next** button will take you to the **Step Four** window, which displays a summary of the web form. Click on the **Finish** button and this will create a web form. You will notice a new space called **news** inside **Company Home | Data Dictionary | Web Forms**. Inside this space you will see the news.xsd file.

The screenshot shows the Alfresco interface with the following details:

- Header:** Company Home > Data Dictionary > Web Forms
- Left Panel - Web Forms:**
 - Web Forms**: This view allows you to browse the items in this space.
 - Web Content Forms**
- Right Panel - news Space:**
 - news**: This view allows you to browse the items in this space.
 - Create News Content**
 - Browse Spaces**: No items to display. Click the 'Create Space' action to create a space.
 - Content Items**:
 - news.xsd**: 2.96 KB, 21 November 2009 16:47

8. Continue the previous steps to upload the blogs.xsd and training.xsd files.

Please provide the **Output path pattern** as common/inc/\${name}.xml for creating blogs and training web forms when the **Web Form Details** window is opened.

Do not proceed to the subsequent sections without first creating the web forms. The remaining sample solution is based on these web forms. Please specify the **Output path pattern** as shown in a previous screenshot for news, training, and the blogs web form.

The Alfresco Wiki website (http://wiki.alfresco.com/wiki/Forms_Developer_Guide and http://wiki.alfresco.com/wiki/Creating_XForms_Widgets) contains information about forms and widgets.

Rendition templates

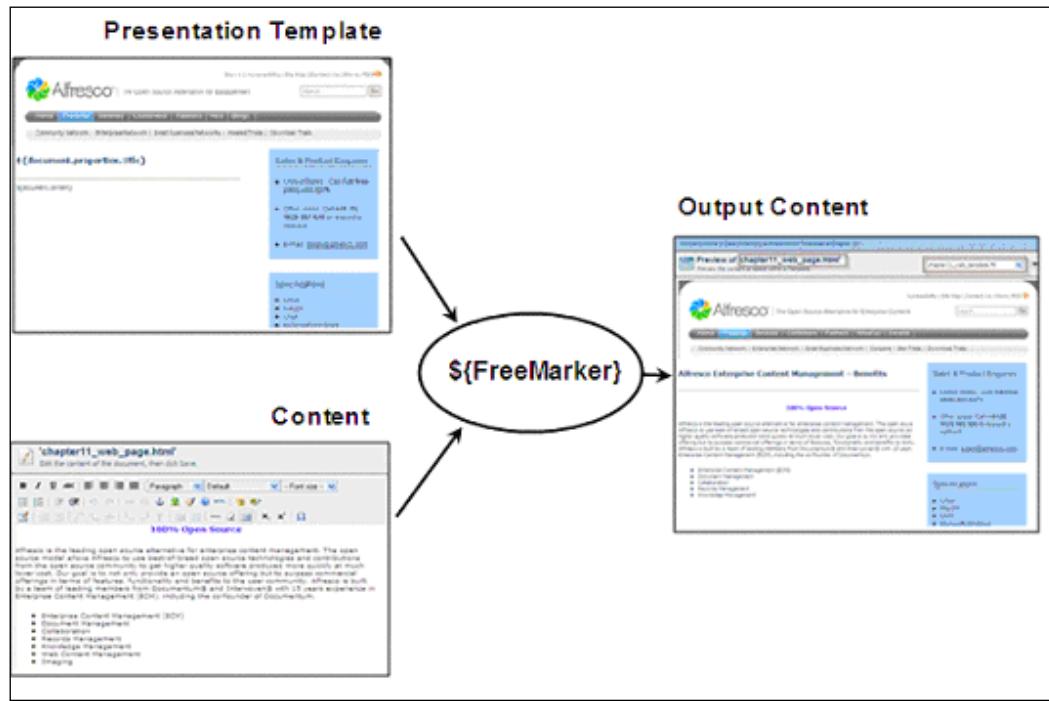
Rendition templates simplifies and accelerates the web publishing process by transforming the web form-managed XML content into web content. Rendition templates can be written in the FreeMarker template language (with an extension `.ftl`), Extensible Stylesheet Language Transformations (with an extension `.xslt`), and Extensible Stylesheet Language Formatting Objects (with an extension `.xsl-fo`). You can leverage the built-in renditioning engine for FreeMarker, XSL, and XSL-FO template languages. After a content item (XML file) is created with a web form, each rendition template is configured for that content type to produce an output in a desirable format. It mainly includes plain text, JSP, PDF, and HTML.

The second step of the Create Web Form wizard suggests configuring a template for rendition output. In this section, you will learn in detail about how to create and configure a template while creating web forms. We will use the FreeMarker template for rendition in this chapter and will see an overview about XSLT and XSL-FO.

Using FreeMarker templates for renditions

FreeMarker is an open source template engine. It is a generic tool for generating text output (which can be anything from HTML to auto-generated source code) based on templates. FreeMarker is designed to be practical for the generation of HTML web pages, by following the MVC (Model View Controller) pattern. The idea behind using the MVC pattern for dynamic web pages is that you separate the content authors from the programmers. This separation is useful, even for projects where the programmer and the HMTL page author are the same person, as it helps to keep the application clear and easier to maintain. If you want that page to be more dynamic, then you begin to put special parts into the HTML, which will be understood by FreeMarker. For instance, `"${ . . . }"` will produce output with the actual value of the thing inside the curly brackets. Presentation templates are written in FreeMarker template language and will have a `.ftl` extension.

In the following figure, the content authors create document content in Alfresco. The programmers create the presentation template file with stylesheets and HTML code, taking care of look and feel requirements. The final content will be generated by the FreeMarker engine (which is embedded in Alfresco) by applying the presentation template on the document content as shown in the following figure:



FreeMarker template engine within Alfresco

The FreeMarker template engine is embedded within Alfresco. FreeMarker takes schema as input and generates text (HTML or XML) as output. FreeMarker also supports XSLT to translate XML content.

Alfresco objects available to FreeMarker

The default model provides a set of named objects that wrap Alfresco Node objects to provide a rich, object-oriented layer, suitable for scripting usage. If you are accessing the templates through the web-client UI, then the following named objects are provided by default:

Named object	Description
avm_sandbox_url	This provides the URL to the root of the current sandbox.
form_instance_data_file_name	This provides the name of the file containing the form instance data being used.
rendition_file_name	This provides the filename of the rendition being produced.
parent_path	This provides the parent_path of the rendition being produced.

For example, consider the following FreeMarker template. In FreeMarker, all variables and functions are in a hash called alf:

```
<html>
<head>
<title>Whats New</title>
</head>
<body>
    <h1>${alf.parent_path}</h1>
</body>
</html>
```

At runtime, the value of variable alf.parent_path will be the path of XML file, which is generated while creating content.



The Alfresco Wiki website (<http://wiki.alfresco.com>) contains a complete reference to the FreeMarker template engine.

FreeMarker template-node model API

These objects, and any child node objects, are called template-node objects, and they provide the following API:

Node method	Description
parseXMLDocument (String virtualPath)	A map of the properties of the node. For example, userhome, properties, and name.
	Properties may return several different types of objects; this depends entirely on the underlying property type in the repository. If the property is multi-valued, then the result will be a sequence, which can be indexed like any other sequence or array.
parseXMLDocuments (String formName, String virtualPath)	A sequence (list) of the child nodes. For example, a list of documents in a space.

FreeMarker directives

Like any programming language, the FreeMarker templating language also supports fundamental directives such as the following:

```
 ${ }  
 #if, #else, #elseif  
 #switch, #case  
 #list  
 #assign  
 #function  
 #include  
 <!-- comment -->  
 <!--comment -->
```

Defining and creating FreeMarker templates

Continuing with our example of the Cignex.com website, we have already created web forms for blogs and a news section. Now we will associate these web forms with the rendition templates. We will first learn to create FreeMarker templates. Let's learn how to create .ftl files.

To create .ftl files, we have to start with the standard declaration of the default name space for the associated web form (consider the blog web form) using the <#ftl> tag (check the target namespace of the blogs.xsd file that has been used in creating the blog web form), as follows:

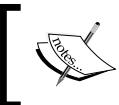
```
<#ftl ns_prefixes={ "D", "http://www.alfrescobook.com/webforms" }>
```

Now we need to display the elements we have used for the blog web form. In the blogs.xsd file, we have defined the root element as "blogs". Assume we want to use the mainTitle, contentSubHeader, and publishedDate elements of the blog.xsd in the .ftl file. The following table shows various uses of FreeMarker directives:

Description	XSD syntax	FTL syntax
Display single element	<pre><x:element name="mainTitle" type="xs:normalizedString" /></pre>	<pre> \${blogs.mainTitle} </pre>
Display recurring elements	<pre><x:element name="subHeader" minOccurs="1" maxOccurs="unbounded"> <x:complexType> <x:sequence> <x:element name="headerLine" type="xs:normalizedString" /></x:sequence> </x:complexType> </x:element></pre>	<pre><#list blogs. subHeader as contentSubHeader > \${ contentSubHeader. headerLine } </#list></pre>
Sort the recurring elements	<pre><x:element name= "subHeader " type="xs:string" minOccurs="1" maxOccurs="unbounded"></pre>	<pre><#list blogs. subHeader?sort_ by("headerLine ")?reverse as contentSubHeader > \${ contentSubHeader. headerLine }</#list></pre>
Conditional checks	<pre><x:element name="publishedDate" minOccurs="0" maxOccurs="1"></pre>	<pre><#if blog.publishedDate != "" > \${blog.publishedDate} </#if></pre>

Let's develop a FreeMarker template for rendition:

1. Navigate to the <installed-alfresco>/extras/wcm/forms folder.
2. Create a file named news.ftl in the above-specified path and populate it with the downloaded code from Packt's website.
3. Create a file named blogs.ftl in the above-specified path and populate it with the downloaded code from Packt's website.



For your reference, a complete guide to FreeMarker directives is available at: <http://FreeMarker.sourceforge.net/docs/>. Download the complete code samples from the Packt website.



Extensible Stylesheet Language

The Extensible Stylesheet Language is divided into two sub-languages: Extensible Stylesheet Language Transformations (XSLT) and Extensible Stylesheet Language-Formatting Objects (XSL-FO).

Using XSLT for renditions

XSLT is a stylesheet language for XML documents. It is designed to transform XML documents into other formats such as **XHTML (Extensible HTML)**. With XSLT you can add/remove elements and attributes to or from the output file. You can also rearrange and sort elements, perform tests, and make decisions about which elements to hide and display. XSLT uses XPath to find information in an XML document. XPath is used to navigate through elements and attributes in XML documents. XSLT uses XPath to define parts of the source document that should match one or more predefined templates. When a match is found, XSLT will transform the matching part of the source document into the result document. XSLT is written in stylesheet language and will have .xsl extension.

Some of the fundamentals used for XSLT are:

Declare namespaces	<?xml version="1.0" encoding="UTF-8"?> <xsl:stylesheet version="1.0" xmlns:xhtml="http://www.w3.org/1999/xhtml" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:newsevents="http://www.cignex.com/newsevents" xmlns:fn="http://www.w3.org/2005/02/xpath-functions" exclude-result-prefixes="xhtml">
Define output format	<xsl:output method="html" encoding="UTF-8" indent="yes" doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN" doctype-system="http://www.w3.org/TR/xhtml1/DTD/ xhtml1-transitional.dtd" media-type="text/html"/>
Define Template	<xsl:template match="/">
Iterate from a set of repeating elements	<xsl:for-each select="/newsevents:news_events">(newsevents is prefix and news_events is complex element defined)
Select the values of single occurring element	<xsl:value-of select="newsevents:title"/>
Provide conditions	<xsl:if test="/newsevents:news_events/ newsevents:training-date"> <xsl:value-of select=" newsevents:news_events/ newsevents:training-date"/> </xsl:if>

Let's develop an XSL template for rendition:

1. Navigate to the <installed-alfresco>/extras/wcm/forms folder.
2. Create a file named `training.xsl` in the above-specified path and populate it with the downloaded code from the Packt website.



Download the complete code samples from Packt Publishing's website. For your reference, a complete guide to XSL template is available at: <http://www.w3schools.com/xsl/>.

Using XSL-FO for renditions

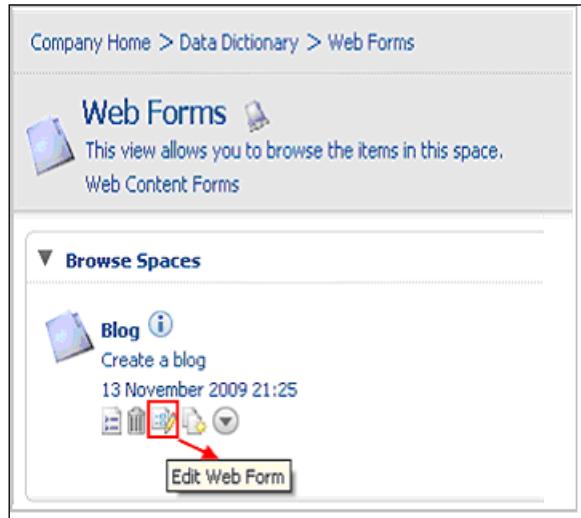
It is designed to provide a mechanism for formatting XML data for print, screen, and other output media. Transforming XML for print is accomplished by transforming an XML document to a Formatting Objects (FO) document, which itself is XML based, via XSLT. The Formatting Objects processor is able to read the FO document and transform it for different types of print output. The most common and best-supported print output is currently Adobe PDF. XSL-FO has the advantage of being able to produce PDFs.

The template files are now ready to use. The next task is to associate rendition templates with the web forms created earlier, for this web form should be created in Alfresco. In our case, we have already created a web form in the earlier section.

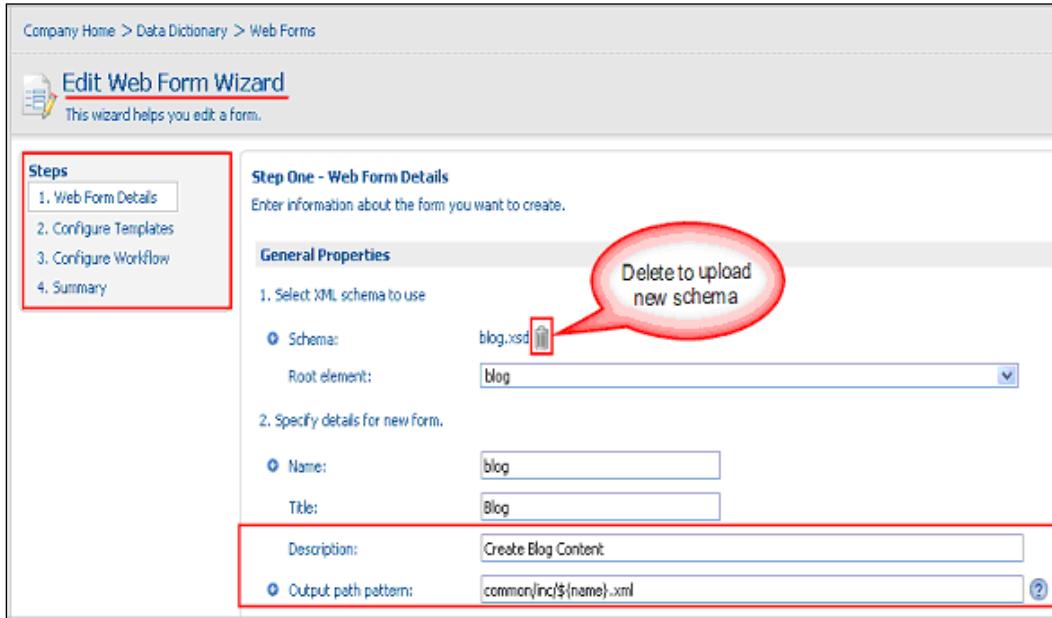
Associating rendition templates to web forms in Alfresco

Templates cannot be associated through the web project after the web form is created. Since the web form is already created, the association of template must be done through **Company Home | Data Dictionary | Web Forms | <web-form space name>**. We will configure our template to the blog's web form. The process is as follows:

1. Ensure that the Alfresco Server is up and running.
2. Go to **Company Home | Data Dictionary | Web Forms | <web-form space name> | Blog:**



3. Clicking on **Edit Web Form** action image under the web form space name (as shown in the previous screenshot) opens the **Edit Web Form Wizard** window. You may notice that the **Edit Web Form Wizard** window has four steps, as shown in the following screenshot. The first step is to delete the previously uploaded XSD (if you want to upload another XSD) file and then upload the XSD, the second step is to configure templates for rendition output, the third step is to define the workflow for form data, and the fourth step is to confirm the web form.



4. In the **Step One** window, you will notice the information provided about the web project if this corresponding web form is associated. You will also see that the rest of the values are prepopulated on the basis of the attached web form. You are given the facility to upload or delete the existing XSD file, if required. You can also change the **Output path pattern**. For now we will not make any change, so click on **Next**.

5. In the **Step Two** window, the **Configure Templates** page is shown. This step allows you to upload a template file in order to generate forms in various outputs. Click on **Browse** to locate and upload the `blog.ftl` file created in the `<installed-alfresco>/extras/wcm/forms` folder. After uploading, it will automatically fill up values, such as **Rendering Engine**, **Name**, **Title**, **Rendition mimetype**, and **Output path pattern**. By default, the **Rendition mimetype** will be selected as **HTML**. You can change the output path pattern and also the extension. This means the file will be created with the specified extension in the specified path. Take a look at the following screenshot and change the output path as specified. We have also changed the extension in order to generate the JSP file. By default, an HTML file will be generated. Click on the **Add to List** button to add the uploaded templates. You can click on the **Add to List** button as many times as you want to upload templates. A list of associated templates will be visible at the bottom. The **Remove** icon is also provided to delete the template in case it's not required. Click on **Next**:

Company Home > Data Dictionary > Web Forms

Edit Web Form Wizard
This wizard helps you edit a form.

Steps

1. Web Form Details
- 2. Configure Templates**
3. Configure Workflow
4. Summary

Step Two - Configure Templates
Enter information about the rendering engine templates you want to use for form blog.

1. Select the rendering engine template to use
Rendering Engine Template File: `blog.ftl`

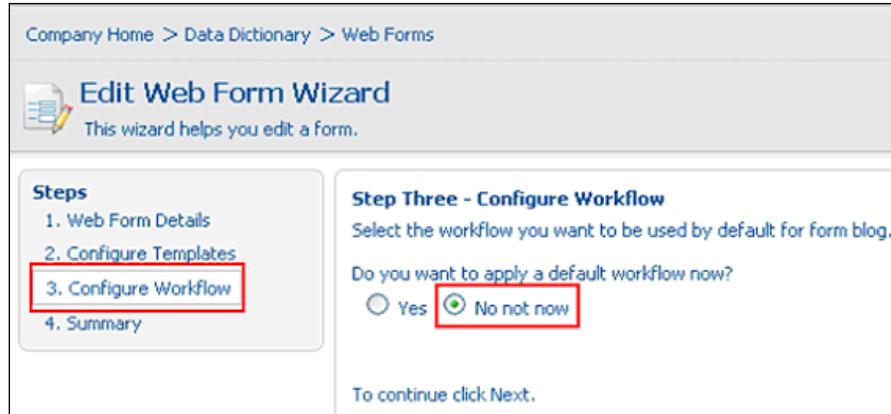
2. Specify details for the new rendering engine template

<input type="radio"/> Rendering Engine:	<input checked="" type="radio"/> FreeMarker	<input type="radio"/> XSLT	<input type="radio"/> XSL-FO
<input type="radio"/> Name:	<input type="text" value="blog.ftl"/>		
Title:	<input type="text" value="blog"/>		
Description:	<input type="text" value="Generate Blogs User Interface"/>		
<input type="radio"/> Rendition mimetype:	<input type="text" value="HTML"/>		
<input type="radio"/> Output path pattern:	<input type="text" value="\${name}.jsp"/>		

Selected Rendering Engines
No selected items.

<input type="button" value="Add to List"/>	<input type="button" value="3. Add to List"/>												
<table border="1"> <tr> <td>Name:</td> <td>blog.ftl</td> </tr> <tr> <td>Type:</td> <td>FreeMarker</td> </tr> <tr> <td>Title:</td> <td>blog</td> </tr> <tr> <td>Description:</td> <td>Generate Blogs User Interface</td> </tr> <tr> <td>Rendition mimetype:</td> <td>text/html</td> </tr> <tr> <td>Output path pattern:</td> <td><code>\$(name).jsp</code></td> </tr> </table>		Name:	blog.ftl	Type:	FreeMarker	Title:	blog	Description:	Generate Blogs User Interface	Rendition mimetype:	text/html	Output path pattern:	<code>\$(name).jsp</code>
Name:	blog.ftl												
Type:	FreeMarker												
Title:	blog												
Description:	Generate Blogs User Interface												
Rendition mimetype:	text/html												
Output path pattern:	<code>\$(name).jsp</code>												
<input type="button" value="Remove"/>													

6. In the **Step Three** window, the **Configure Workflow** page is shown. Select the **No not now** option (this step allows the creation of default workflows for form data; you will learn more about this in the next chapter *WCM Workflow*).



7. Clicking on the **Next** button will take you to the **Step Four** window, which displays a summary of the web form. Clicking on the **Finish** button will create the template. You will notice **blog.ftl** inside the blog space:



8. Continue the previous steps to upload the news.ftl and training.xsl file. Provide the **Output path pattern**, as shown in the following screenshot, for creating the news template when the **Configure Templates** pane is open.

Step Two - Configure Templates
Enter information about the rendering engine templates you want to use for form news.

1. Select the rendering engine template to use
Rendering Engine Template File: news.ftl

2. Specify details for the new rendering engine template

Rendering Engine: FreeMarker XSLT XSL-FO

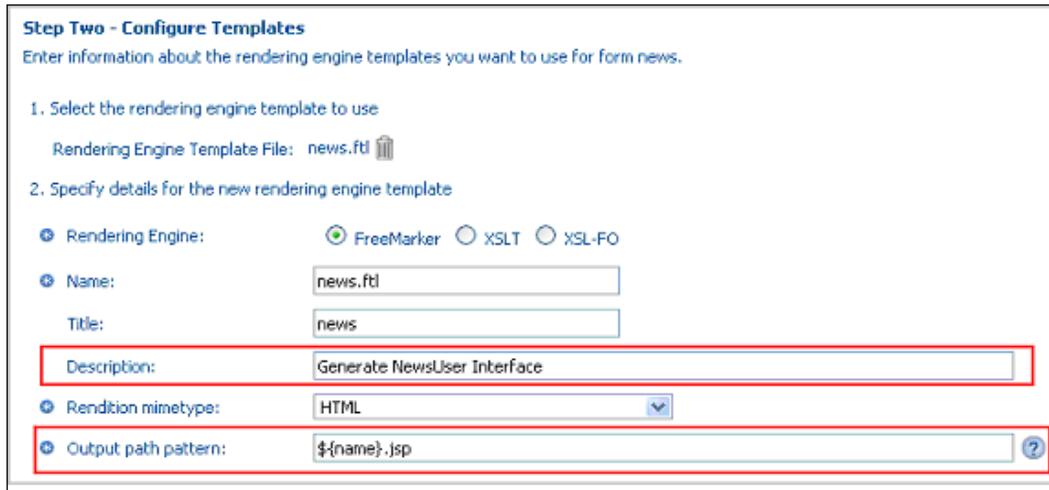
Name: news.ftl

Title: news

Description: Generate NewsUser Interface

Rendition mimetype: HTML

Output path pattern: \${name}.jsp



9. Provide the **Output path pattern** as shown in the following screenshot for creating the training template when the **Configure Templates** pane is open.

Step Two - Configure Templates
Enter information about the rendering engine templates you want to use for form training.

1. Select the rendering engine template to use
Rendering Engine Template File: training.xsl

2. Specify details for the new rendering engine template

Rendering Engine: FreeMarker XSLT XSL-FO

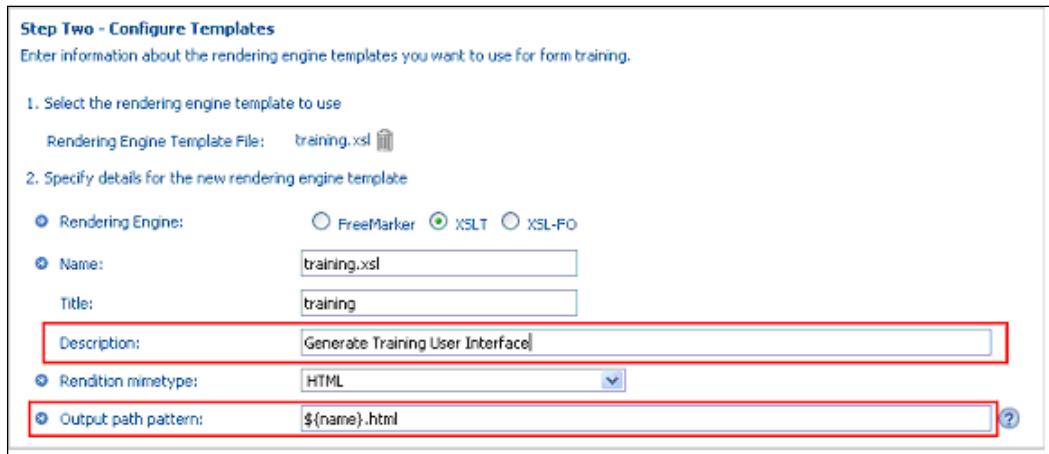
Name: training.xsl

Title: training

Description: Generate Training User Interface

Rendition mimetype: HTML

Output path pattern: \${name}.html





Do not proceed to the subsequent sections without first configuring the templates. The remaining sample solution is based on these web forms and templates. Remember to specify the output path pattern as shown in the previous screenshots for news and blogs. The Alfresco Wiki website (http://wiki.alfresco.com/wiki/WCM_Forms_Rendering) contains information about forms and widgets.

Associating web forms and renditions for specific/multiple project(s)

Now we have created templates for our web forms. We have also associated our template with the web forms and we are ready to use these forms to create and publish content. We will subscribe to both (news and blog) web forms in this web project and configure each for our web project website's unique content generation. Once configured for a given web project, a Content Contributor can create new web content using web forms via the **Create Content** wizard.

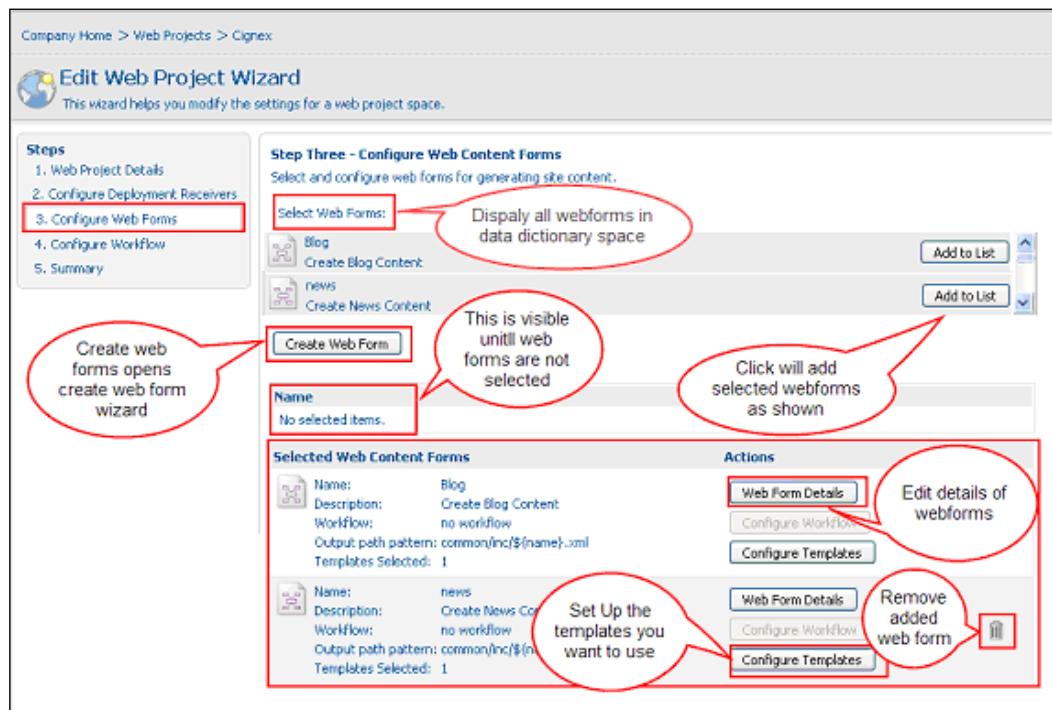
Follow these steps to configure web forms and templates to the web project:

1. Go to **Company Home | Web Projects | Cignex**.
2. Select **Edit Web Project Settings** from the action menu.

The screenshot shows the 'Company Home > Web Projects > Cignex' page. On the right, there is a context menu with several options: 'Actions' (highlighted with a red box), 'View Details', 'Create Webapp Folder', 'Edit Web Project Settings' (underlined in red), 'Invite Web Project Users', 'Delete All Deploy Reports', and 'Delete'. Below the menu, there is a 'Staging Sandbox' section with details: 'Created On: 22 November 2009', 'Created By: admin', and a note that 'There is one user working on this web project.' There are also links for 'Recent Snapshots' and 'Content Awaiting Launch'.

3. Click on **Next**.
4. On the next screen, again click on **Next**.

5. In the **Step Three** window, you will notice that the **Select Web Forms** drop down will display all the available forms in **Company Home | Data Dictionary | Web Forms**. If you want to select from the drop down, then you have to click on the **Add to List** button.
6. If you want to create a new web form, you can select the **Create Web Form** button, which will open the **Create Web Form** wizard. Once the web form is created you can click on the **Add to List** button to add the newly-created web form. You can see the added web forms in the panel as shown in the following screenshot. You will also see the space of the newly created web form inside **Company Home | Data Dictionary | Web Forms**.
7. You can also set up the templates if more than one template is associated with each web form selected. You can also select the web form details to change the **Output path pattern**, **Title**, **Description**, and also configure workflow. You will learn in detail about configuring workflows in the next chapter *WCM Workflow*:



8. Click on **Next** and then click on **Finish**.

A few points to be considered are as follows:

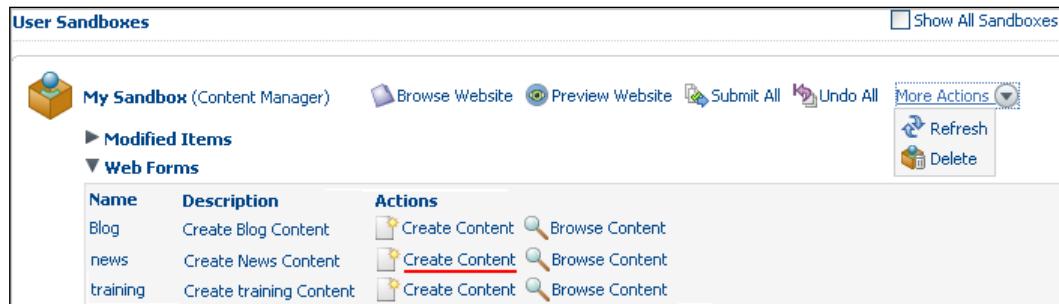
- As we have learned, we can create a web form using two ways.
- Once a web form is created, association of the rendering template is available only with the first option (**Create Web Form**).
- Output paths for web forms and templates can be different for each web project for the same web form. Using the first option (**Create Web Form**), you can create a generic output path and then this can be customized for various projects using the second option (**Edit Web Project setting**).

Creating dynamic content

To support creation and editing of web content, Alfresco provides support for a sandbox development model as you have read in *Chapter 3, Getting Started with Alfresco WCM*.

There are multiple methods of adding and creating content for a web project. In addition to creating web content within a project, you can also upload individual files from your computer and perform a bulk import. For more details refer to *Chapter 3, Getting Started with Alfresco WCM*. Let's assume that we have all the folders in place inside the Cignex web project. We just want to create content for the **news** and **Blog** web forms. The link for the news, blog, and training web form is specified inside `index.jsp`, which is placed under the root folder of the project. Follow these steps:

1. Go to **Company Home | Web Projects | Cignex**.
2. Expand the web form panel in the user sandbox. You will see a list of web forms, which are configured for the web project. Select **Create Content** next to the web form:



The screenshot shows the 'User Sandboxes' interface for the 'My Sandbox (Content Manager)' project. At the top, there are buttons for 'Browse Website', 'Preview Website', 'Submit All', 'Undo All', 'More Actions' (with options for 'Refresh' and 'Delete'), and 'Show All Sandboxes'. Below this, the 'Modified Items' and 'Web Forms' sections are visible. Under 'Web Forms', there is a table with three rows:

Name	Description	Actions
Blog	Create Blog Content	Create Content Browse Content
news	Create News Content	Create Content (highlighted with a red box) Browse Content
training	Create training Content	Create Content Browse Content

3. Put the values and click on **Next**:

Company Home > Web Projects > Cignex

Create Web Content Wizard
This wizard helps you to create a new content item for a website.

Steps

1. Web Content Details (highlighted with a red box)

2. Step Two - Author Web Content

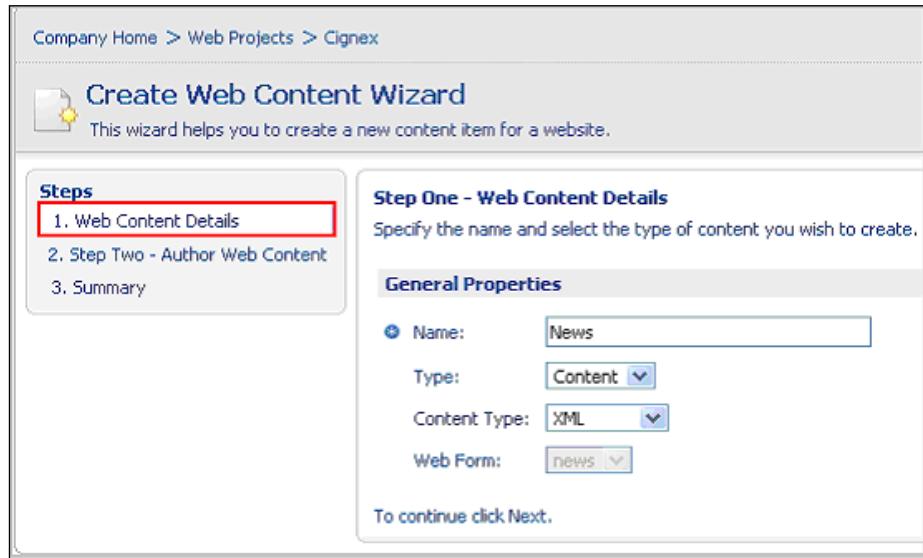
3. Summary

Step One - Web Content Details
Specify the name and select the type of content you wish to create.

General Properties

Name: News
Type: Content
Content Type: XML
Web Form: news

To continue click **Next**.



4. Fill in the details and click on **Next**:

Create Web Content Wizard
This wizard helps you to create a new content item for a website.

Steps

1. Web Content Details (highlighted with a red box)

2. Step Two - Author Web Content (highlighted with a red box)

3. Summary

Step Two - Author Web Content
Enter your document content into the repository.

News News (circled with a red oval, labeled 'Put Content for News')

Brief Title: Packt and Cignex realeases a new book on Alfresco3

News Page Headline: Packt and Cignex realeases a new book on Alfresco3

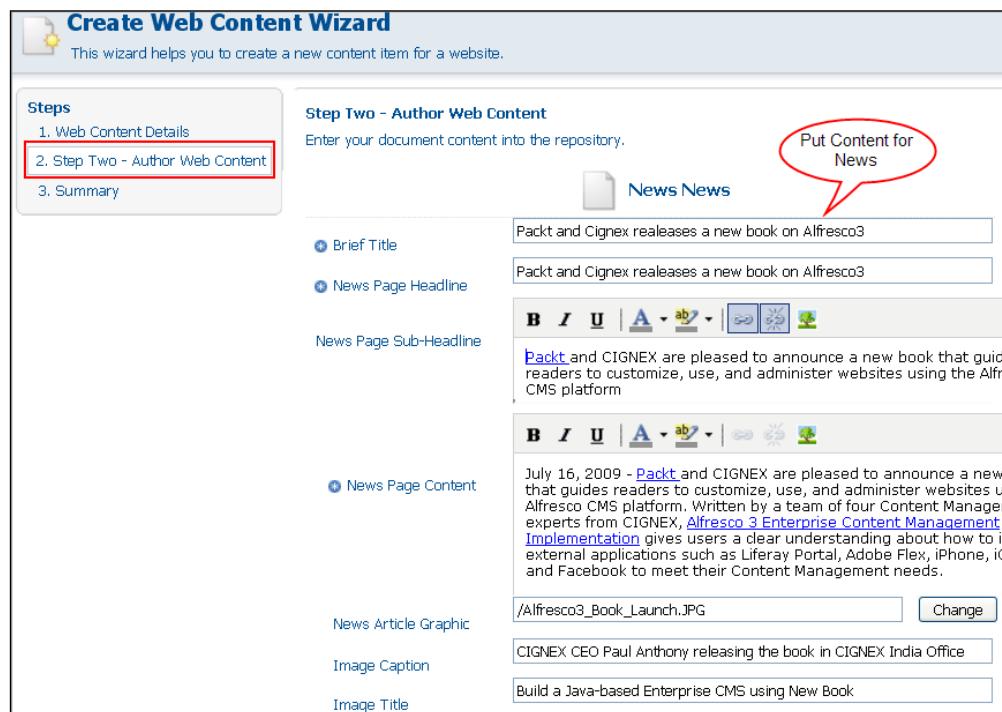
News Page Sub-Headline: 
Packt and CIGNEX are pleased to announce a new book that guid readers to customize, use, and administer websites using the Alfr CMS platform

News Page Content: 
July 16, 2009 - Packt and CIGNEX are pleased to announce a new that guides readers to customize, use, and administer websites us Alfresco CMS platform. Written by a team of four Content Manag experts from CIGNEX, [Alfresco 3 Enterprise Content Management Implementation](#) gives users a clear understanding about how to ir external applications such as Liferay Portal, Adobe Flex, iPhone, iG and Facebook to meet their Content Management needs.

News Article Graphic: /Alfresco3_Book_Launch.JPG

Image Caption: CIGNEX CEO Paul Anthony releasing the book in CIGNEX India Office

Image Title: Build a Java-based Enterprise CMS using New Book



5. In the **Summary** section, you can preview the content created by you:

Summary
The wizard has successfully created the content and all renditions.

Content Details

	EditNews.xml	
Form:	Blog	
Location:	/ROOT/common/inc/news.xml	

Rendition Details

	EditNews.jsp	
Rendered by news into /ROOT/common/inc/News.jsp		

Submit these 2 files when wizard finishes.

6. You can see that the content gets created in the folder specified in the output path. You can preview it from there also:

Website 'Cignex' sandbox 'admin'
Use this view to browse the files and folders within the sandbox for a web project.

Preview Website Create

ROOT > common > inc

Browse Folders

Name	Creator	Created Date	Modifier	Modified Date	Type	Actions
News.jsp	admin	22 December 2009 15:20	admin	22 December 2009 15:20		
News.xml	admin	22 December 2009 15:20	admin	22 December 2009 15:20		

Browse Files

Name	Size	Creator	Created Date	Modifier	Modified Date	Type	Actions
News.jsp	2.55 KB	admin	22 December 2009 15:20	admin	22 December 2009 15:20		
News.xml	0.8 KB	admin	22 December 2009 15:20	admin	22 December 2009 15:20		

7. You can also see the content and folder under the modified section (if it is newly-created content) and can preview it from there.

8. The following screenshot shows only the folders because this was the first news article created and required folders to hold the content (/common/inc/News.xml) and the rendered JSP page (/common/inc/News.jsp). Submitting these will clear the **Modified Items** box, and the next time this news is edited in the Modified box, it will show just the .xml and .jsp files:

My Sandbox (Content Manager)

Browse Website Preview Website Submit All Undo All More Actions

Modified Items

Selected: [Submit Selected](#) [Undo Selected](#)

<input type="checkbox"/>	Name	Created Date	Modified Date	Size	Actions
<input type="checkbox"/>	common	22 November 2009 19:58	22 November 2009 21:56		

Web Forms

Name	Description	Actions
Blog	Create Blog Content	
news	Create News Content	
training	Create training Content	

9. Continue the previous steps to create the content for blogs and training.
 10. As we have created content for news, training, and blog web forms, we can now preview the website Cignex.com from the **Preview Website** action:

User Sandboxes

Show All Sandboxes

My Sandbox (Content Manager)

Browse Website Preview Website Submit All Undo All More Actions

Modified Items

Selected: Submit Selected Undo Selected

	Name	Created Date	Modified Date	Size	Actions
<input type="checkbox"/>	common	22 November 2009 19:58	22 November 2009 21:56		

Web Forms

Name	Description	Actions
Blog	Create Blog Content	
news	Create News Content	

11. You will see the Cignex home page in the next screen. You will see the news, training, and blog content created by you in the website. The following screenshot shows the dynamic content created on our website:

The screenshot displays the Cignex website homepage with several sections highlighted by red circles and arrows:

- Blog section:** A speech bubble points to a large image of a jet engine, indicating where user-created blog content might appear.
- News Section:** A red circle highlights the "News & Press" section under the "Solutions" heading.
- Training Section:** A red circle highlights the "LATEST TRAININGS" section.
- Check Link for News Content created by you:** An arrow points from this text to the "News & Press" section.
- Check Link for training Web Content:** An arrow points from this text to the "LATEST TRAININGS" section.
- Click Link to check your created content for blogs:** An arrow points from this text to the "Blog section".

The website features a header with the Cignex logo and navigation links for SOLUTIONS, SERVICES, TECHNOLOGY, CUSTOMERS, RESOURCES, and COMPANY. Below the header, there's a sidebar with icons for OPEN UP, UP GRADE, and DEPLOY, followed by a main content area with sections like "CIGNEX is your answer", "What's & New", and "WATCH CMS SCREENCAST".

12. The following screenshot shows the content created by you:



13. On clicking the links in the news section, you will find the news section you created, as shown in the following screenshot:



14. Similarly, you can see the training content.

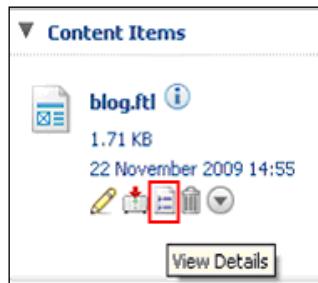
Edit web forms for renditions

You have added associate web forms and template to a web project. Now you want to update the templates and preview it. Follow these steps:

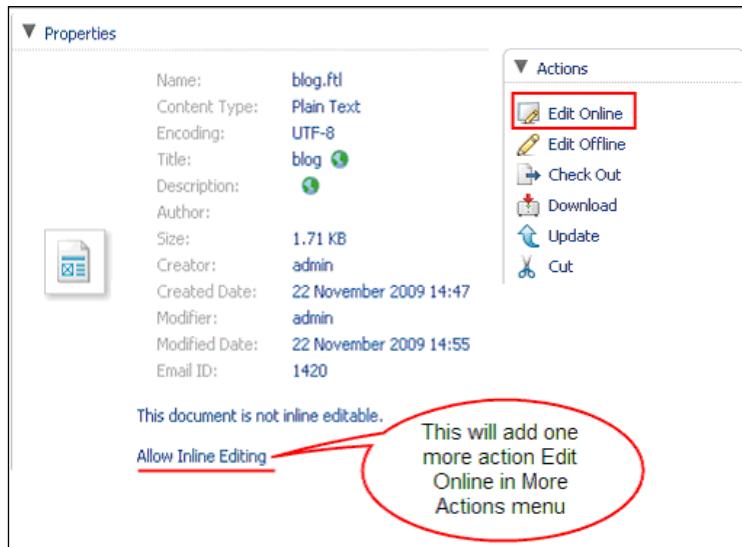
1. Navigate to Company Home | Data Dictionary | Web Forms | blog <Web Form Name> space.
2. You can update/edit the template using two options.

First option: Using edit

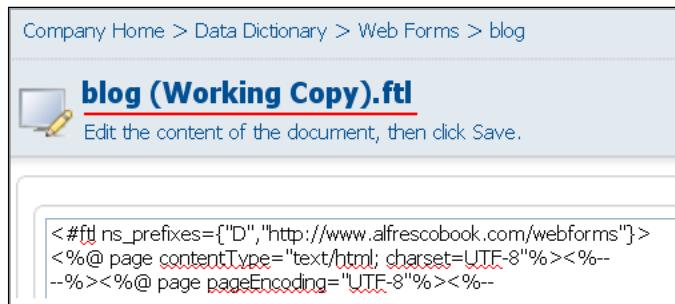
1. Click on the **View Details** action of the blog.ftl file as shown in the following screenshot:



2. Click on the **Allow Inline Editing** option. It will add one more action, **Edit Online**, in the right-hand side action pane as shown in the following screenshot:



3. Clicking on **Edit Online** will open **Edit window** in which you can edit your template file. After editing, click on **Save**.

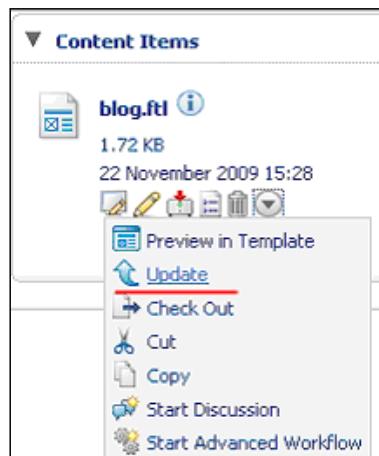


4. You can click on the **Done Editing** action to save the changes.

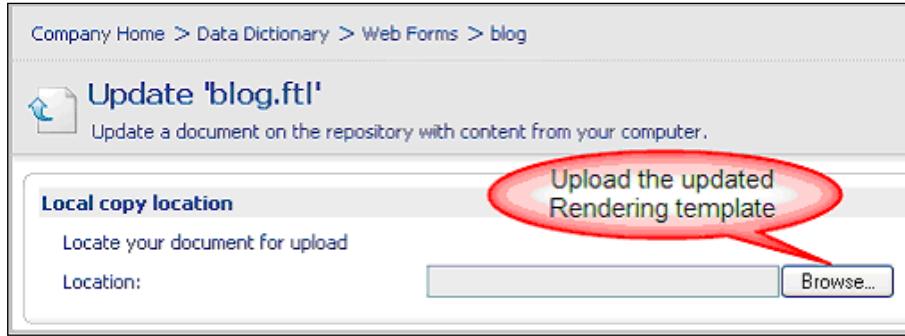


Second option: Using update

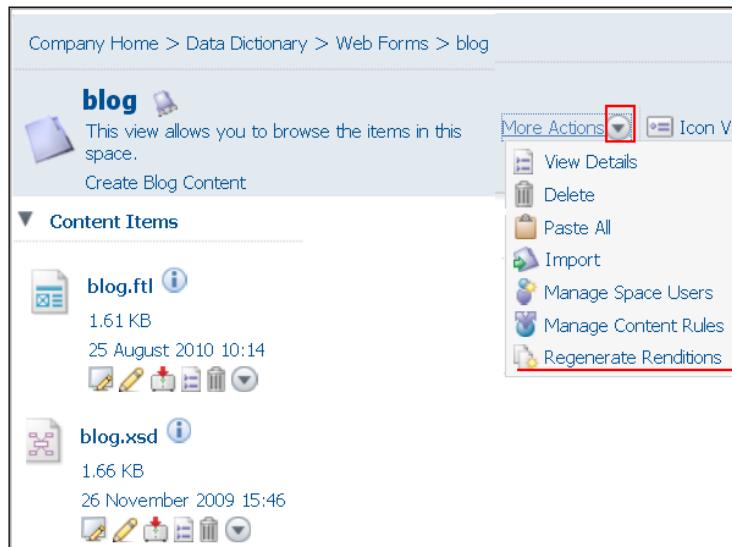
1. Click on the **Update** action under the **More Actions** menu of the blog.ftl file:



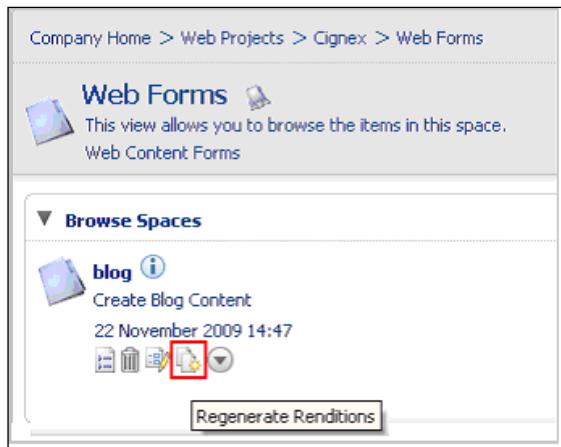
2. Clicking on **Update** will open a window to browse and upload the updated rendering template:



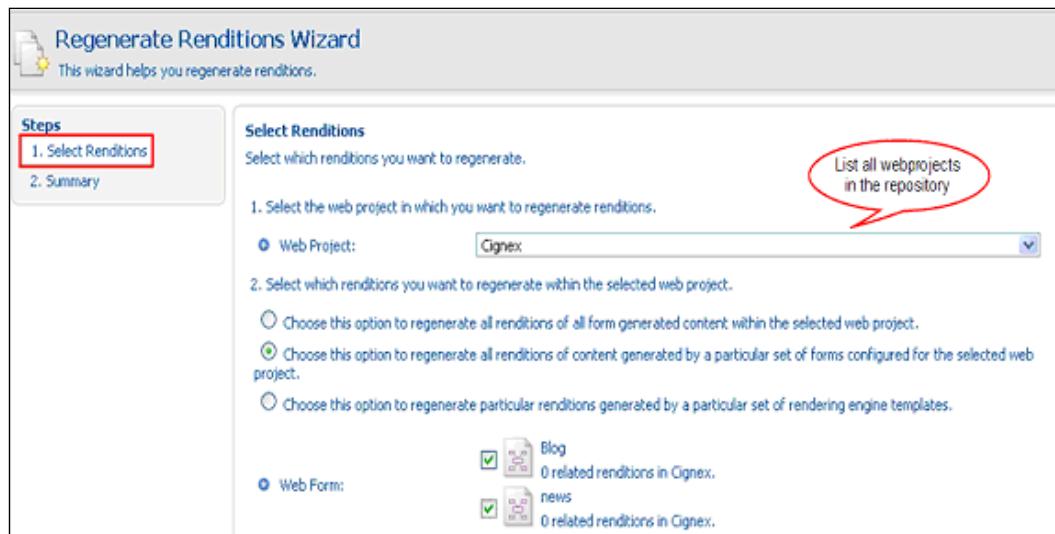
3. Click on **Update**.
4. Now the template is updated. In the next screen, you can select the **Regenerate Renditions** action under the **More Action** menu:



5. Also, you can select **Regenerate Renditions** from the blog space itself as seen in the following screenshot:



6. In the next screen, you can select for which web project you want to update the renditions. See the following screenshot:



7. If the content is deployed on the staging server, you can preview the content from there.

Associating a .xml file to the web form

Suppose you want to do data migration of content from other systems to Alfresco WCM. There is a manageable way by which you can easily move content as XML in WCM and link it to a web form. You can do this through the API or web client extension. Follow these steps to perform the procedure using the web client extension.

1. Upload an .xml file to the web project. This XML file should have the structure similar to the web forms. Go to the Cignex web project. In **My Sandbox**, click on the **Browse Website** link to browse to the folders created in the web project.

My Sandbox (Content Manager)

► Modified Items
▼ Web Forms

Name	Description	Actions
Blog	Create Blog Content	
news	Create News Content	

Browse Website

Browse to the folders created in the web project

2. Go to the **common | inc** folder. Click on **Add Content** to upload an .xml file of the same structure as the news or blog .xsd.

Company Home > Data Dictionary > Web Forms > Web Projects > Cignex

Website 'Cignex' sandbox 'admin'
Use this view to browse the files and folders within the sandbox for a web project.
Cignex

ROOT > common > inc

Browse Folders

Name	Creator	Created Date

Create

- Add Content
- Create Web Content
- Create Folder
- Bulk Import

Click Add Content to Upload .xml file

3. The uploaded file will be shown under the **Browse Files** folder. Click on the **Edit** action for the `EditNews.xml` file as shown in the following screenshot:

Browse Files						Items Per Page <input type="text" value="25"/>
Name	Size	Creator	Created Date	Modifier	Modified Date	Actions
 EditNews.xml	0.8 KB	admin	22 November 2009 21:08	admin	22 November 2009 21:08	           
 News.jsp	2.55 KB	admin	22 December 2009 15:20	admin	22 December 2009 15:20	     
 News.xml	0.8 KB	admin	22 December 2009 15:20	admin	22 December 2009 15:20	     

4. Opening this will show a link: **To edit this file using a Web Form, click here.**

Company Home > Web Projects > Cignex

EditNews.xml
Edit the content of the file.

To edit the file 'EditNews.xml', click the link below and if asked select Save.

 **EditNews.xml**

When the download is complete, click Close.
[To edit this file using a Web Form, click here](#)

5. Clicking on this link will show a next screen in which you can select a web form to be associated with the uploaded XML file. Click on **OK**.

Company Home > Web Projects > Cignex

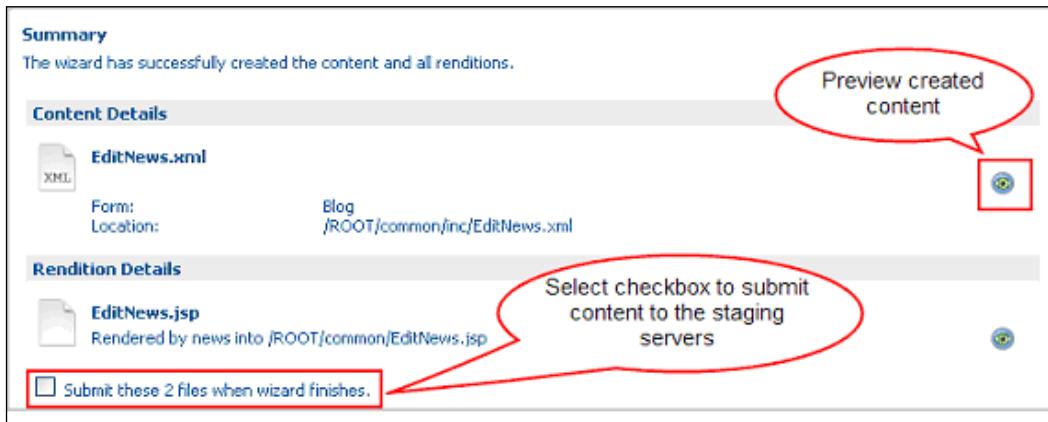
Select Web Form
Select the Web Form to use for editing this file

Select the Web Form to use for `EditNews.xml`.

Web Form: **Select proper web form**

To edit `EditNews.xml` using the selected Web Form, click ok. If `EditNews.xml` is not a Web Form generated asset and should be treated as regular content, click cancel.

6. You will see the news web form. You can fill the content for the news web form and select **Next**. The summary page will be open, which will display details about the web form and content. You can preview content (to preview content, note that the virtual server should be running). Also, you can submit the content to the staging server (you will learn in detail about the staging server in later chapters). Click on **Finish**.



Static and dynamic include of content

This section is in reference to the **Advance schema attributes** section of the web form. In the previous section, we discussed only the syntax to be used for include. Consider a scenario where you want to populate a dynamic combo box.

Let's create a web form to input values for populating the combo box. Create the `newstype.xsd` and `newsType.ftl` files and populate these with the downloaded code from the Packt website.

[

 Download the complete code samples
from the Packt website.
]

1. To create and configure web forms to the Cignex web project, see the section *Creating web forms*. Create web content using the newsType web form. Click on the **Create Content** action.

The screenshot shows the 'My Sandbox (Content Manager)' interface. In the top left, there's a folder icon labeled 'My Sandbox (Content Manager)'. Below it, under 'Modified Items', it says 'No modified items'. Under 'Web Forms', there's a table:

Name	Description	Actions
Blog	Create Blog Content	
news	Create News Content	
NewsType	Create Type Of News	

In the top right corner, there are 'Browse Website' and 'Preview Website' buttons.

2. On clicking **Create Content**, add values as shown in the following screenshot.

The screenshot shows the 'Step One - Web Content Details' dialog. It has a header 'Step One - Web Content Details' and a sub-instruction 'Specify the name and select the type of content you wish to create.' Below this is a 'General Properties' section with the following fields:

<input checked="" type="radio"/> Name:	NewsType
Type:	Content
Content Type:	XML
Web Form:	NewsType

At the bottom, it says 'To continue click Next.'

3. Click on **Next**. Add values as shown in the next screenshot:

Step Two - Author Web Content
Enter your document content into the repository.

The screenshot shows the Alfresco Content Authoring interface. At the top, there's a header with a file icon and the text "NewsType NewsType". Below the header, there are two sections for "ContentType". The first section has an "Id" field containing "entertainment" and a "Display Name" field containing "Entertainment". The second section has an "Id" field containing "business" and a "Display Name" field containing "Business". Each section has a toolbar below it with icons for add, edit, delete, and other operations.

 Do not proceed to the subsequent sections without first creating web content using the news Type web form. Please specify the output path pattern as shown in the earlier screenshots. Provide common/inc/\${name}.xml as the output path pattern for XSD and \${name}.jsp for template. Also, keep your virtual server running.

4. Edit news.xsd and add the following line of code before defining a root element. We are including one JSP into the other XSD.

```
<xs:include schemaLocation="/common/inc/NewsType.jsp" />
```

5. Add the following block of code anywhere after the root element is defined:

```
<xs:element name="ContentType" type="trn:newsType" >
<xs:annotation>
<xs:appinfo>
<alf:appearance>minimal</alf:appearance>
</xs:appinfo>
</xs:annotation>
</xs:element>
```

6. Now create content using the news web form. You will see the combo box populated with the values you added while creating content for the NewsType web form.

The screenshot shows a web form titled "Step One - Author Web Content". It has a sub-instruction "Enter your document content into the repository.". There are three input fields with radio buttons:

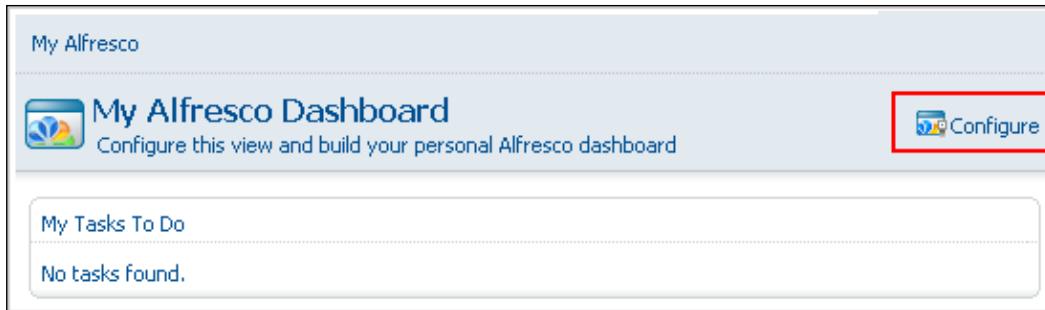
- Brief Title: Packt and CIGNEX release a new book on Alfresco3
- News Page Headline: Packt and CIGNEX release a new book on Alfresco3
- ContentType: A dropdown menu is open, showing "Entertainment" as the selected value. A red oval surrounds this dropdown, and a callout bubble points to it with the text "Values populating from another web form". Other options in the dropdown are "Business" and "Sport".

Below these fields is a rich text editor toolbar with buttons for bold, italic, underline, and other styles. At the bottom of the form is a preview area containing the text: "Packt and CIGNEX are pleased to announce a new book that guides readers to customize, use, and administer websites using the Alfresco CMS platform."

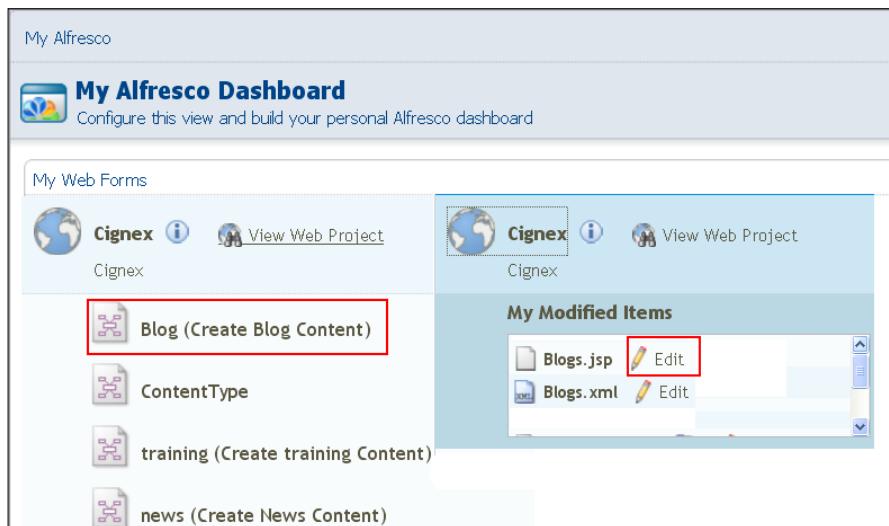
Web publishing dashlets

Alfresco provides two new dashboard components, *My Web Forms* and *My Web Files*, which put the needed set of publishing operations at the fingertips of any knowledge worker, helping them to access and speed through the publishing process while masking the underlying process and structure of the Web Content Management system. Follow the next steps to configure dashlets:

1. In the Alfresco Explorer user interface, the **My Alfresco** area is known as the **Dashboard**.
2. Click on the **My Alfresco** menu link in the toolbar to view your personal dashboard.
3. To start configuring your dashboard, click on the **Configure** icon given in the **My Alfresco** dashboard. The **Configure Dashboard Wizard** will open up a screen, allowing you to select the dashboard layout and dashlets. Select the style of layout you wish to have for your dashboard. Click on the **Next** button to move to the next step, selecting the dashboard components.



4. Select the following components:
 - **My Tasks To Do:** Lists all the tasks assigned to you that are pending
 - **My Web Forms:** Lists all the web forms available to the web project and can create content directly from here
 - **My Web Files:** Lists all the web forms available to the web project and can edit content directly from here
5. Click on the **Next** or **Finish** button to save your selection. The selection is effective immediately, and you can see the dashboard with your selections, as shown in the next screenshot:



Summary

In this chapter, we have learned how Alfresco WCM makes it simpler to manage content through a user-friendly and technology-neutral (as much as possible) interface:

- Alfresco web forms enable users to create XML content from a simple browser-based form.
- There are three types of renditions available: FTL, XSLT, and XSL-FO. With renditions, you can manage the content in various formats.
- A dynamic website can be developed easily with the web forms.
- Web forms can associate with specific/multiple projects.
- Web content can be searched using a query.
- Dashlets are provided to access easily and speed through the publishing process.

In the next chapter, we are going to learn about WCM Workflows. Workflows can be configured for any web form. Through workflow, web content can be submitted to the Staging Sandbox. From the Staging Sandbox it can deployed to servers.

5

WCM Workflows

Workflow is an automation of a business process, during which documents are passed from one participant to another for action, according to a set of procedural rules. Every content management system implementation will have workflow requirements. Workflow provides ownership and control on the content and processes. Alfresco web project uses workflows to support any set of changes, either automated or user-driven steps, in a business process before final commit to the Staging Sandbox. WCM Workflows can be configured for each form or for any arbitrary set of non-form assets. In this chapter, you will understand the basic out-of-the-box workflow capabilities of the Alfresco WCM and the ways to extend it as per your business requirements.

By the end of this chapter you will have seen the:

- Advantages of workflow
- Introduction to the workflow process
- Out-of-the-box workflows
- Associating workflows to web forms
- Associating workflows to web projects
- Dynamically changing workflow for each snapshot submission
- Process of defining and using your custom workflow
- Use of expiration date

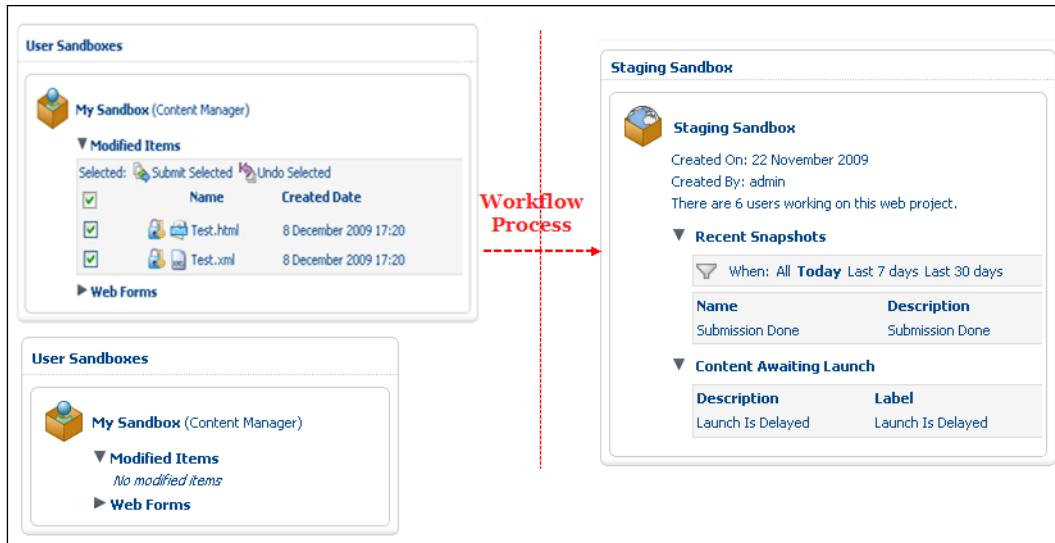
Why workflows are required

Consider that we want to develop any website; so our first step will be to create web content, and the next step will be to get the approval of the reviewers in order to publish content onto live servers. We would also like to set up a test server so that one can preview the changes and finally the content can be pushed to the staging servers and then live servers. At each stage it seems we require a set of processes to be followed to get our approval completed and go to the next stage. This can be achieved using WCM Workflows.

WCM Workflows are required to follow the user approval process for submitted content, to publish them on the Staging Sandbox. You must deal with the approval of content items that are routed through workflow and the submission of the items to staging servers and then the live server. Submitted changes are routed via workflow (assume workflow is configured for the web project) for one or multiple users to review and approve, either serially or in parallel, using Alfresco's out-of-the-box workflow (or any custom-built WCM Workflow). On approval of the item, content is submitted to the Staging Sandbox. Even if we don't configure workflow for any item, the submission takes place in the background using workflow and each content item remains in the **Modified Items** list until its submission is complete. To verify this you can see a workflow task in the **My Task To Do** dashlet of the content reviewer for a fraction of a second. Refresh the page as necessary, to see if the workflow task has been removed from the **My Task To Do** dashlet and that the **Modified Items** list of the user's sandbox is empty.

The additional advantage you get from workflow is that you can configure the launch date and expiration date when promoting content to staging. Prior to this launch date, users can preview the changes and cancel the pending launch if required. In the Staging box, it will be reflected in **Content Awaiting Launch**. Upon expiration, users can see that item as a task in their **My Tasks To Do** dashlet to decide whether the content should be updated or removed from the site. This content expiration also works in conjunction with Alfresco's new Links Validation service, ensuring that the site's link integrity is incrementally tested against the new pending update. In this manner, content cannot only be regularly refreshed through an automated, task-driven review process, but can also be tested upon change so that no errors are accidentally introduced in the form of broken links.

Another advantage is that once the content is approved, a snapshot is taken to the Staging box. For each submission a snapshot is maintained, which gives the advantage to roll back to any submission at any point of time.



In the previous chapter you learned how to create content. At this point, your web project is populated with content items, some imported and some created. Now you want to submit the content to the Staging box. You must configure a workflow process to get the approval, so the content is now routed through workflow to content reviewers for approval. Once you have approved all of your content, modified items are deployed to the staging server.

Introduction to the workflow

The WCM Workflow uses the **JBoss Business Process Management (jBPM)** engine in its core. jBPM is an open source, standalone workflow engine. It can run in any servlet container – it doesn't require the JBoss Application Server. The jBPM engine is responsible for managing deployed processes, instantiating and executing processes, persisting process state and metadata to a relational database (via Hibernate), and tracking task assignment and task lists. JBoss jBPM is a flexible, extensible workflow management system with an intuitive process language to express business processes graphically in terms of tasks, wait states for asynchronous communication, timers, and automated actions. With jBPM, the Alfresco platform is extended to support complex, task-oriented processes. jBPM is built on the idea that any process can be described as a graph or a set of connected nodes. jBPM maintains a list of tasks assigned to each participant. How users interact with the task list is up to each application. In Alfresco, a dashlet displays a to-do list for the currently logged in user. As users complete their tasks the tasks are removed from the to-do list.

Workflows are described with "process definitions" using an XML-based language called **jBPM Process Definition Language (jPDL)**. jPDL is one example of a graph-based execution language.

jPDL is one process language that is build on top of that common framework. It is an intuitive process language to express business processes graphically in terms of tasks, wait states for asynchronous communication, timers, and automated actions. To bind these operations together, jPDL has the most powerful and extensible control flow mechanism. jPDL has minimal dependencies and can be used as easily as using a Java library.

jPDL includes a graphical designer tool. The designer is a graphical tool for authoring business processes. It's an eclipse plugin. Workflows can be performed either by handcoding XML or by using the JBoss jBPM Process Designer tool. In the later sections, we will see both the approaches.

Workflow process

Whenever content is created or modified for a web project, it is added under **Modified Items** of a user's sandbox. A list of modified items in a user's sandbox can be promoted from a sandbox to a Staging Sandbox. Content is promoted from a user's sandbox to the Staging Sandbox by initiating the Submit Wizard. This wizard is initiated by clicking on either the **Submit Selected** or **Submit All** actions in the Alfresco web client. Submitted changes are routed via workflow for one or multiple users to review and approve, either serially or in parallel using Alfresco's out-of-the-box **Web Site Submission** workflow (or any custom-built WCM Workflow). Once approved, a snapshot is automatically taken of the Staging box.

The workflow process can be divided into three steps.

1. The first step is to define and deploy a workflow in WCM.
2. The second step is to associate the workflow to a web form or web project.
3. The third step is to submit modified content to the staging server using a workflow.

Out-of-the-box workflow

Alfresco provides an inbuilt **Web Site Submission** workflow to be associated with the web project.

Workflow can be configured for both form-based and non-form-based assets.

There are two out-of-the-box flows available for the **Web Site Submission** workflow:

- Serial
 - 1. Reviewers approve one at a time and in order.
 - 2. If any reviewer rejects, the workflow is sent back to author.
 - 3. Upon resubmit, all reviewers are reassigned a task, one at a time.
- Parallel
 - 1. Reviewers review in parallel to one another.
 - 2. If any reviewer rejects, the workflow is sent back to the initiator once all the reviewers either approve or reject the workflow.
 - 3. Upon resubmit, all reviewers are reassigned a task in parallel.

When the workflow is complete, it promotes the change set to the Staging Sandbox.

Consider a case where a user has to conduct training. In order to arrange the training, the concerned user needs to take approval from alternative authorities so that the event can be published on the site and participants can register themselves for the training. If any of the authorities rejects the training for whatever reason, it goes to the trainer who was seeking approval. Now the trainer can either resubmit to reviewers or cancel it. If all the authorities approve the training, the event can be published on the site.

For this create **Mark Steven**, **Keenan Hall**, and **Crawford Caton** as users. Invite **Mark Steven** as Content Manger on the **Cignex** web project and **Keenan Hall** and **Crawford Caton** as Content Reviewers on the **Cignex** web project. For more information about creating users and inviting users refer to *Chapter 3, Getting Started with Alfresco WCM*.

Configuring workflows

Workflows can be configured for both web forms and non-generated web content. In order to submit content to the Staging Sandbox, workflows need to be configured. We will discuss shortly how form-based and file-based workflow can be configured, and also how content can be submitted to the Staging box.

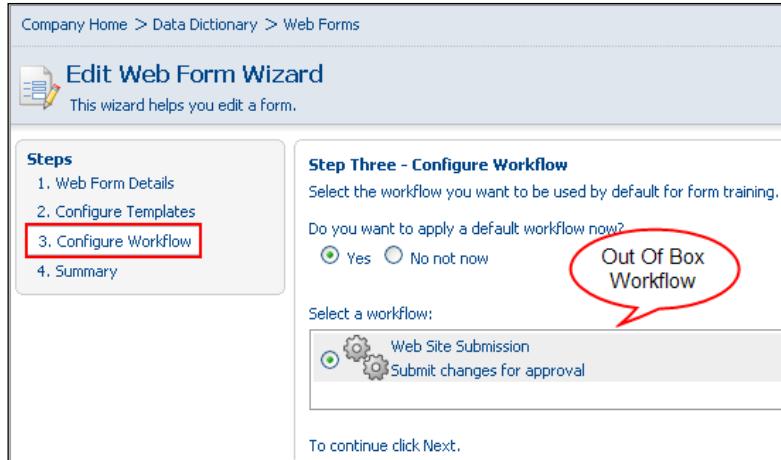
Associating workflows to web forms

Workflow for web forms can be configured using the **Create Web Form Wizard** or the **Edit Web Form** wizard. Using these approaches, workflows can be configured for all web projects and for a specific web project. As we have already created a web form in the last chapter, we will use the same web form to drive a workflow. In this example, we have to use the **Edit Web Form Wizard**. Follow these steps to assign a workflow for the web form:

1. Ensure that the Alfresco server is up and running.
2. Go to **Company Home | Data Dictionary | Web Forms**.
3. Click on the **Edit Web Form** action under the **training** (web form name) space.



4. Clicking on this will open **Edit Web Form Wizard**. Click on **Next** twice to reach the **Configure Workflow** window.
5. Select the **Yes** radio button and click on **Next**.

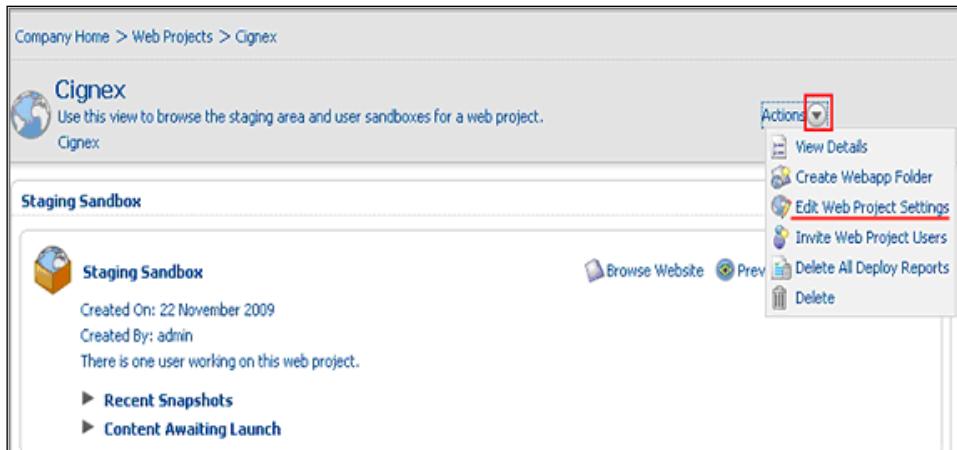


6. Click on **Finish**.

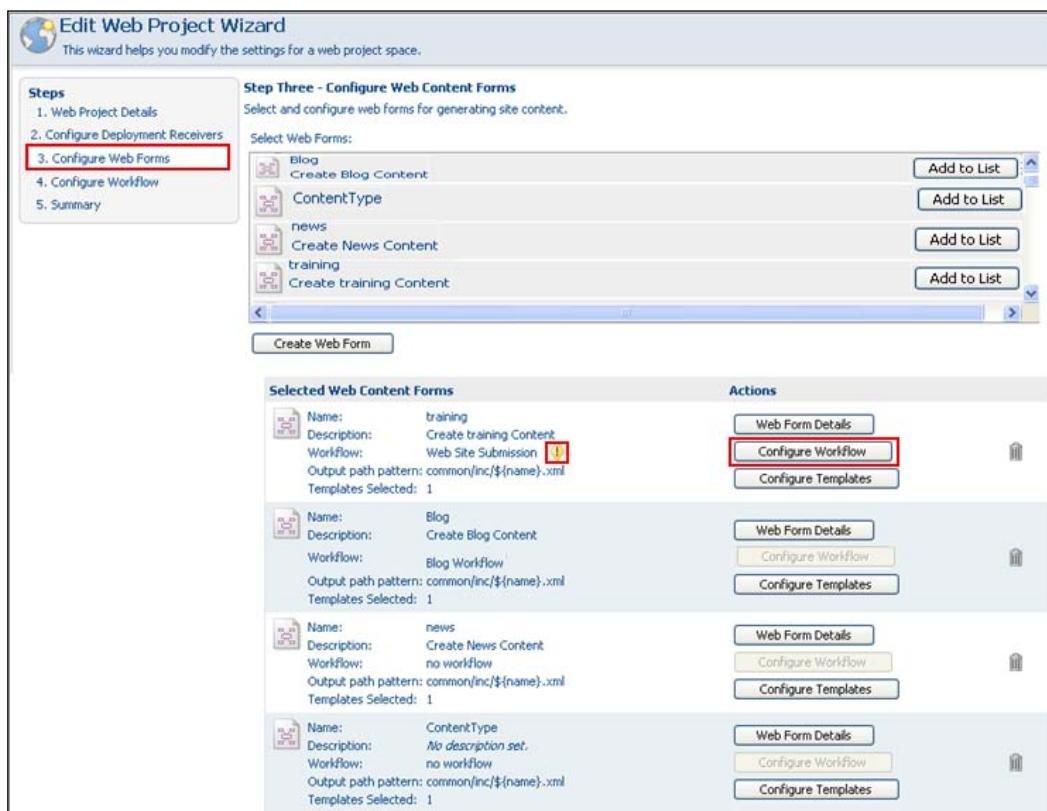
Associating workflows to web projects

To assign a workflow for a specific project, you can use two approaches. The first approach is by using the **Create Web Project Wizard** that will be used if you are creating a new web project. The second approach is by using the **Edit Web Project Wizard**. We will be using the second approach since we have already created the Cignex web project. Follow these steps to associate workflow for a web form:

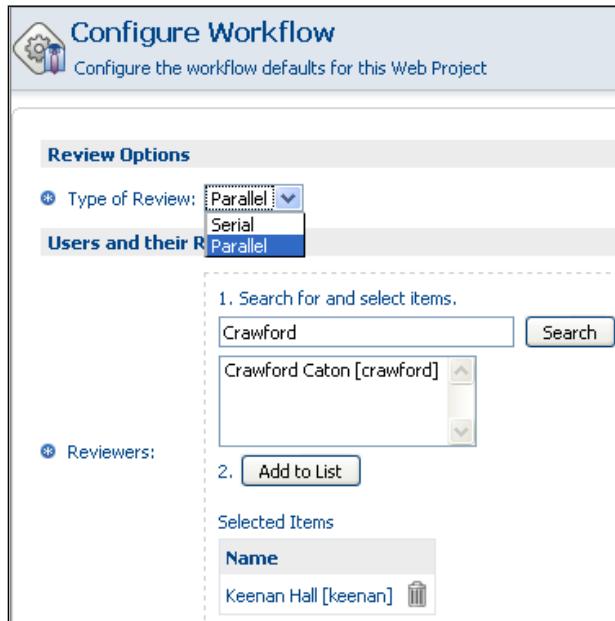
1. Ensure that the Alfresco server is up and running.
2. Go to **Company Home | Web Projects | Cignex**.
3. Select **Edit Web Project Settings** from the action menu.



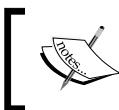
4. Click on **Next**.
5. On the next screen, click on **Next**.
6. In the **Step Three** window, you will notice the added web forms in the panel as shown in the following screenshot. You can see the **Configure Workflow** button available for each web form. This button is enabled only for those web forms for which we have configured workflows. Notice the attention icon next to the workflow. This indicates a workflow has been selected but not configured:



7. On clicking the **Configure Workflow** button, the **Configure Workflow** window is opened. This window is used to configure *form-based workflows* (web form generated content). This is specific for web forms only. Fill out the details as shown in the following screenshot:



8. Click on **OK**. Now, if you notice, the attention icon disappears.
9. Click on **Next**. You will again see the **Configure Workflow** window. Click on the **Add to List** button to add the workflow for the web project. Once the workflow is added in the panel, you can configure file-based workflows (non-web form generated content). They are configured based on filename pattern matching. In **Workflow Settings**, note the default regex pattern matches `.*`. This default means that any asset web form generated, and non-web form generated, will go through this review process. Also, you can add the **Web Site Submission** workflow multiple times in this wizard. For each instance you can configure a different chain of reviewers for different sections of the websites or types of assets by modifying the regex pattern match in **Workflow Settings**, for example, `.css`, `.html`.



Please note that the **Configure Workflow** window can be used for both file and non-web form generated assets. But this window is mainly used for non-web form generated assets.

The screenshot shows the 'Edit Web Project Wizard' interface. The left sidebar lists steps: 1. Web Project Details, 2. Configure Deployment Receivers, 3. Configure Web Forms, 4. Configure Workflow (highlighted with a red box), and 5. Summary. The main panel is titled 'Step Four - Configure Workflow' with the sub-instruction 'Select and configure workflow for non-form generated assets.' Below this, a section titled 'Select Workflows:' shows a list with 'Web Site Submission' and a 'Submit changes for approval' link. A red callout bubble points to the 'Add to List' button with the text 'Will add workflow in below panel'. At the bottom, a table shows 'Selected Workflows' with one entry: 'Web Site Submission' (with a yellow warning icon). A red callout bubble points to this entry with the text 'workflow not configured'. The table includes columns for 'Name', 'Configure', 'Filename pattern match: .*', and 'Configure Workflow' (which is also highlighted with a red box). A red arrow points from the 'Configure Workflow' button in the table to the same button in the main configuration area. At the bottom of the panel, a message says 'To continue click Next.'

10. Click on **Finish**.

Submitting content to the Staging box

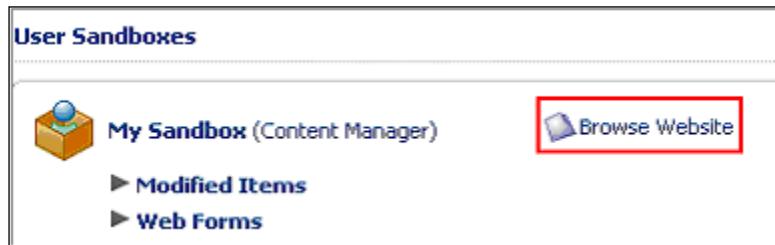
Content can be submitted in two ways.

- The first option is by using the **Edit Web Content Wizard** or the **Create Web Content Wizard**. With this option you can submit only *form-based content*.
- The second option is by using the **Submit Selected** or **Submit All** option provided under **Modified Items**. With this option you can submit both *form-based assets* and *non-form based assets*.

Using the Edit Web Content wizard

We assume that the web content is already created as it was done in the previous chapter. So we are going to use the **Edit Web Content** wizard. Log in as **Mark Steven**, who is the Content Manger of the Cignex project. Refer to *Chapter 3, Getting Started with Alfresco WCM* for how to invite users to a web project and assign roles. Follow these steps to directly submit content to the workflow:

1. Go to **Company Home | Web Projects | Cignex**.
2. Select the **Browse Website** option to browse all files and folders.



3. Go to the common/inc folder.
4. Click on the **Edit** action placed next to the training.html file.
5. Click on **Next**.
6. The **Summary** window is opened. It has the functionality to directly submit content to the workflow (this functionality is also available in the **Create Web Content Wizard**).
7. Select those checkboxes to submit directly. This saves you from initiating a separate submission process.

The screenshot shows the 'Edit Web Content Wizard' summary page. It displays the following information:

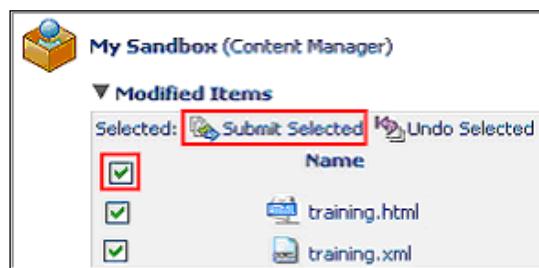
- Steps:** 1. Author Web Content, 2. Summary
- Summary:** The wizard has successfully created the content and all renditions.
- Content Details:** training.xml (XML file), Form: training, Location: /ROOT/common/inc/training.xml
- Rendition Details:** training.html (HTML file), Rendered by training into /ROOT/common/inc/training.html
- Checkboxes:** A checked checkbox labeled 'Submit these 2 files when wizard finishes.'
- Text at the bottom:** To add the content to this space click **Finish**. To review or change your selections click **Back**.

8. Click on **Finish**.

Using Submit Items Wizard

Follow these steps to submit all or selected modified content to the workflow:

1. Go to **Company Home | Web Projects | Cignex**.
2. The **Submit Selected** option is provided under **Modified Items of My Sandbox**. You can use this option in two cases. The first being, if you want to submit some of the items. For this, you can select the checkboxes placed next to the item to submit the selected item. Second, select the top-most checkbox to select all the files for submission.



3. The **Submit All** action can also be used if you want to submit all the items listed under **Modified Items**.



4. The **Submit** option is also provided at the right-hand side of the item to submit only that item.

	Name	Created Date	Modified Date	Size	Actions
<input type="checkbox"/>	Blogs.jsp	23 November 2009 13:07	12 December 2009 18:43	3.35 KB	
<input type="checkbox"/>	Blogs.xml	23 November 2009 13:07	12 December 2009 18:43	3.75 KB	

5. When the **Submit Items** wizard is opened, you will notice most of the information is filled. If you want you can modify this. You are required to put information about the workflow name and the description that will display the name and description of the corresponding snapshot in staging. Configure the workflow if you want to modify some details. By doing this the default file workflow is overridden for this specific workflow instance. You can also configure the launch and expiration dates. When a launch date is set, approved content changes are maintained in their own separate workflow sandbox until the specified time and date. Prior to that date, the Content Publisher or Reviewers can preview the future state of the website and immediately promote or cancel the pending launch. Specific expiration dates can also be set on a global or asset-by-asset basis. Upon expiration, users are automatically assigned the asset as a task, so that they can make a determination whether the asset should be updated or removed from the site. Put the details as shown in the following screenshot:

Name	Description	Path	Modified Date	Expiration Date	Actions
training.html	Rendered by training into /ROOT/common /ROOT/common/inc/training.html /inc/training.html		10 December 2009 22:15		
training.xml		/ROOT/common/inc/training.xml	10 December 2009 22:15		

6. Click on **OK** to start the workflow process. The content will start submitting to the Content Reviewer. Once in a workflow, assets cannot be submitted again until the workflow completes. Files in a current workflow will not have the Submit, Update, and Revert commands available.

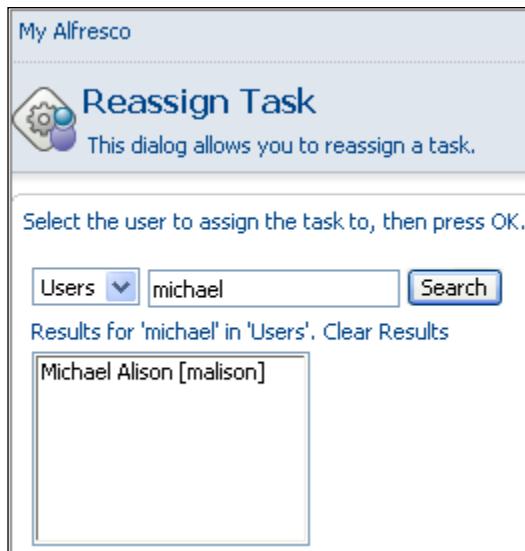
7. Log in as **Keenan Hall** and **Crawford Caton** who are the Content Reviewers. Click on the **My Alfresco** menu link in the toolbar to view your personal dashboard. The **My Tasks To Do** dashboard lists all your tasks. You can choose to **Manage** or **Reassign** the task. Assume you have logged in as **Crawford Caton**.

[ As we have selected **Parallel** as a review type, both the users will be able to see the task in their **My Tasks To Do** dashboard. Any one of the users can approve or reject the task. If one of the users rejects the task, it will go back to the user who has initiated the content submission. The user can decide whether to resubmit or cancel the task.]

8. For various business reasons you can reassign the task by clicking on the **Reassign** button.



9. Once you click on the **Reassign** button, you will see the **Reassign Task** window as shown in the following screenshot. You can search for the users and reassign the task to an appropriate user.



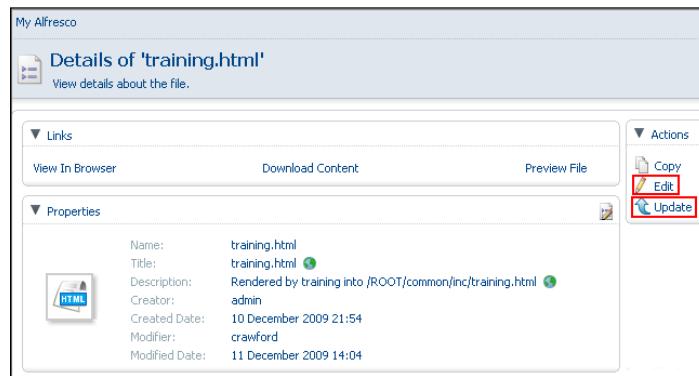
10. You can manage the task by clicking on the **Manage** button.

The screenshot shows a table titled 'My Tasks To Do'. It has columns for Description, Type, Id, Created, Due Date, Status, Priority, and Actions. A single row is visible: 'Training' (Type: Review, Id: 165, Created: 10 December 2009 22:42, Status: Not Yet Started, Priority: 3). The 'Actions' column contains three icons: a magnifying glass, a pencil, and a trash can. The 'trash can' icon is highlighted with a red box. Below the table, there is a page navigation bar showing 'Page 1 of 1' and several small navigation icons.

11. Once you click on the **Manage** button you will see the **Manage Task** window as shown in the following screenshot. You can Update, Edit, Preview, and Revert the items. You can see the workflow history, input comments, and see the change of set attached to this task:

The screenshot shows the 'Manage Task: Review' interface. It includes tabs for 'Task Properties', 'Reviewers', 'Resources', and 'Workflow History'. The 'General' tab is active, showing fields for Identifier (166), Description (Training), Submission Label (Training), Status (Not Yet Started), and a Comment about conducting a training. The 'Reviewers' tab shows a type of review (Parallel) and reviewers (Keenan Hall [keenan], Crawford Caton [crawford]). The 'Resources' tab lists attachments: 'training.xml' and 'training.html'. The 'Workflow History' tab shows a single entry: 'Training' (Description: Web Site Submission, Task Type: Web Site Submission, Id: 164, Created: 10 December 2009 22:42, Assigned to: mark, Completed on: 10 December 2009 22:42, Outcome: Task Done). On the right side, there are buttons for 'Save Changes', 'Approve', 'Reject', 'Cancel', 'Revert', 'Preview', and 'View Details'. Red arrows and boxes highlight the 'Approve' button, the 'Revert' button, the 'Preview' button, and the 'View Details' button.

12. On clicking the **View Detail** icon you will see the following screen, using which you can **Edit** and **Update** the content if required. Click on **Close** to go back to the **Manage Task** window.



13. To complete the task, click on the **Approve** or the **Reject** button as shown in the previous screenshot.
14. Now log in as **Keenan Hall** and continue the same steps we have just seen for **Crawford Caton**. In this case we are going to reject the training:

Description	Task Type	Id	Created	Assignee	Comment	Completed on	Outcome
Training	Review	166	10 December 2009 22:42	crawford	Permitted for conducting a training	12 December 2009 11:32	Approved
Training	Web Site Submission	164	10 December 2009 22:42	mark		10 December 2009 22:42	Task Done

15. Log in as **Mark** who has submitted this content. Note the task on Mark's task list.

Description	Type	Id	Created	Due Date	Status	Priority	Actions
Training	Rejected	183	12 December 2009 11:59		Not Yet Started	3	

16. Open the task and you will have two options to select; one is **Resubmit For Review** and the second is **Abort Review**. If you click on **Abort Review**, the content will not be submitted to the Staging box and you have to start the procedure of submitting content from scratch (this means the workflow sandbox is deleted, and in the future when you submit the same content, it will create new workflow with a new **Advanced Versioning Manager (AVM)** store). If you click on **Resubmit For Review**, you will find again, task for both the users – **Keenan Hall** and **Crawford Caton** (this indicates the same AVM store will be in use).

Description	Path	Created	Modified	Actions
training.xml	/ROOT/common/inc	10 December 2009 21:54	12 December 2009 11:03	
training.html	Rendered by training into /ROOT/common/inc/training.html	/ROOT/common/inc 10 December 2009 21:54	12 December 2009 11:10	

Comments and status from both Crawford and Keenan

Description	Task Type	Id	Created	Assignee Comment	Completed on	Outcome
Training	Review	166	10 December 2009 22:42	crawford Permitted for conducting a training	12 December 2009 11:32	Approve
Training	Review	165	10 December 2009 22:42	keenan Training cannot be conducted due to logistic reasons	12 December 2009 11:59	Reject
Training	Web Site Submission	164	10 December 2009 22:42	mark	10 December 2009 22:42	Task Done

17. Now we will seek the approval again. Click on **Resubmit For Review** and you will find the task in both the users' dashlets. Assume that you have logged in as **Keenan Hall**.

Description	Type	Id	Created	Due Date	Status	Priority	Actions
Training (2)	Review	184	12 December 2009 12:25		Not Yet Started	3	

18. Open the **Manage Task** dialog and **Approve** the task. Also, log in as **Crawford Canon** and **Approve** the task.

Description	Path	Created	Modified	Expiration Date	Actions
training.xml	/ROOT/common/inc	10 December 2009 21:54	12 December 2009 11:03		
training.html	Rendered by training into /ROOT/common/inc/training.html	10 December 2009 21:54	12 December 2009 11:10		

Description	Task Type	Id	Created	Assignee	Comment	Completed on	Outcome
Training	Rejected	183	12 December 2009 11:59	mark	Please review once again	12 December 2009 12:25	Resubmit for Review
Training	Review	166	10 December 2009 22:42	crawford	Permitted for conducting a training	12 December 2009 11:32	Approve
Training	Review	165	10 December 2009 22:42	keenan	Training cannot be conducted due to logistic reasons	12 December 2009 11:59	Reject
Training	Web Site Submission	164	10 December 2009 22:42	mark		10 December 2009 22:42	Task Done

19. Now log in as **Mark**. Note the task on the **My Task To Do** dashlet only for a fraction of seconds (5 seconds). Refresh it after some time; the content is already submitted to the Staging box.

Description	Type	Id	Created	Due Date	Status	Priority	Actions
Training	Submitted	186	12 December 2009 12:50		Not Yet Started	3	

Page 1 of 1

20. In your user sandbox, expand the **Modified Items** list. Each content item remains in the **Modified Items** list until its submission is complete. Refresh the page after some time. In the **Staging Sandbox**, expand the **Recent Snapshots** list to view the snapshot you have created.

A few things to be considered when an item is attached to a workflow:

- Assets attached to a workflow cannot be submitted to another workflow while the first one is running.
- If you have to rename, delete, or modify any asset items, then those items are also going to submit.
- XML and generated renditions are always placed in workflow as a unit. If you submit any XML item, then all the items related to that XML will also be submitted.
- Each workflow creates "AVM" (it creates a branch till it is closed). Hence, having more and more active instances will degrade the performance. In AVM store, it keeps the entire content of user's sandbox. One should not keep pending workflows for a long time. Whenever a Content Reviewer receives the notification of a successful approval, the task appears in the task list. Close the task as soon as possible. After clicking, the AVM store of that workflow instance will be removed.
- Any change set associated with a workflow is isolated in its own workflow sandbox. Workflow sandboxes are visible via CIFS, but difficult to use due to autogenerated folder names (GUIDs).

- Content Contributors can continue to work in their sandbox without breaking the Reviewer's context.
- Content Reviewers can see an in-context preview of the change set as if it had been applied to the Staging Sandbox.

Dynamically changing workflow for each snapshot submission

Whenever you submit content to the Staging Sandbox, once approved, a snapshot is automatically taken of the Staging to provide an archive of the current version of the site. This snapshot is maintained over time to provide an audit trail and rollback point for previous versions of the site. This gives an advantage of recovering any content at any point of time. It also keeps record of all deleted, renamed, and moved items. Once a snapshot is taken, all committed changes are immediately reflected and available to each user in their own sandbox, enabling all users to consistently check their changes against the latest and greatest version of the website.

The screenshot shows the 'Staging Sandbox' interface. At the top, there is a logo of a blue cube with a white 'S' on it, followed by the text 'Staging Sandbox'. To the right are three buttons: 'Browse Website' (with a document icon), 'Preview Website' (with a globe icon), and 'Refresh' (with a circular arrow icon). Below these buttons, the text 'Created On: 22 November 2009' and 'Created By: admin' is displayed. A message states 'There are 9 users working on this web project.' Under the heading 'Recent Snapshots', there is a table with the following data:

Name	Description	Date	Submitted By	Version	Status	Actions
Training	Training	12 December 2009 12:50	mark	45		

Below the table, under the heading 'Content Awaiting Launch', the message 'No content awaiting launch' is shown.

Creating a custom WCM Workflow for a group

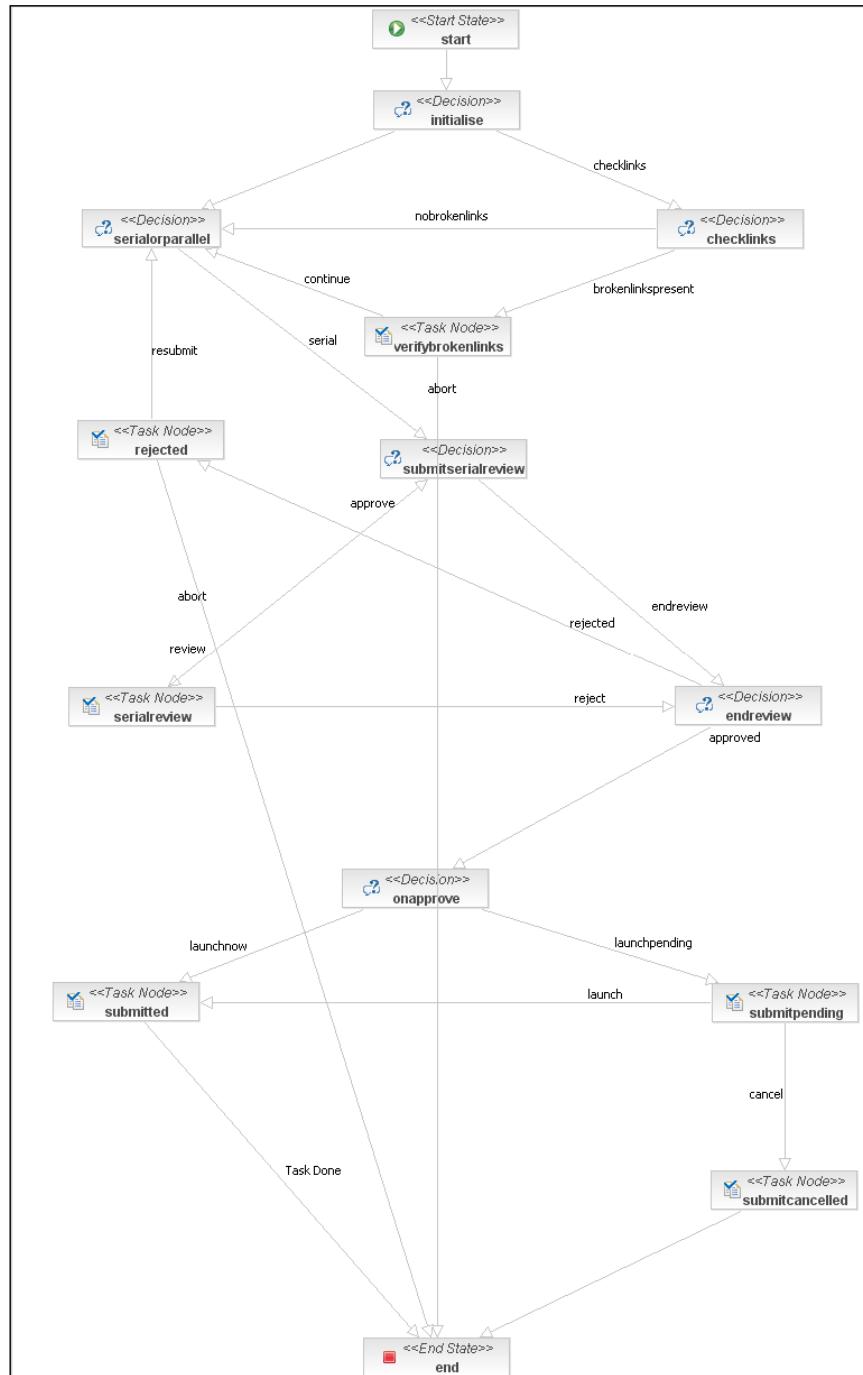
You can define and deploy your own task-oriented workflows in the Alfresco repository. However, you need to follow a particular format to define your workflow and a particular process to deploy it in Alfresco. Workflows can be deployed manually (which requires a restart of the server) and dynamically (without starting the server). For now we will deploy the workflow manually. These customizations are typically deployed via the `alfresco/extension` folder and require the Alfresco server to be restarted to take effect. In the later examples, we will deploy using the dynamic approach.

As an example, we will configure one workflow. The use case scenario is as follows.

There is a section of Blogs and News on the Cignex website, which needs to be updated monthly. The blog has to be published regularly. In order to publish, one needs to follow some process that can be defined in a workflow. The blog has to be reviewed by three different groups. Each group has different roles. Groups approve the blog one at a time and in order. When the blog is submitted, it will go to the first group. All the users belonging to that group will receive a notification via a task in the **My Pooled Tasks** dashlet. Any one of the users can take ownership and approve or reject the task. If rejected, it will go to the initiator. On approval it will go to next group and the process will continue for all three groups. Once the process is complete, a notification will be sent to the initiator. Also the blog would be submitted to the Staging box.

For this, create **Jennifer Bruce**, **Kristie Dawid**, **LeRoy Fuess**, **Michael Alison**, and **Jessica Tucker** as users. Create three groups: **Technical Reviewer**, **Editorial**, and **Publisher**. Add **Jennifer Bruce** and **Kristie Dawid** to **Technical Reviewer**, add **LeRoy Fuess** to **Editorial**, and add **Michael Alison** and **Jessica Tucker** to **Publisher**. Invite **Technical Reviewer**, **Editorial**, and **Publisher** as **Reviewer** on the Cignex web project. For more information about creating a group and users refer to *Chapter 3, Getting Started with Alfresco WCM*.

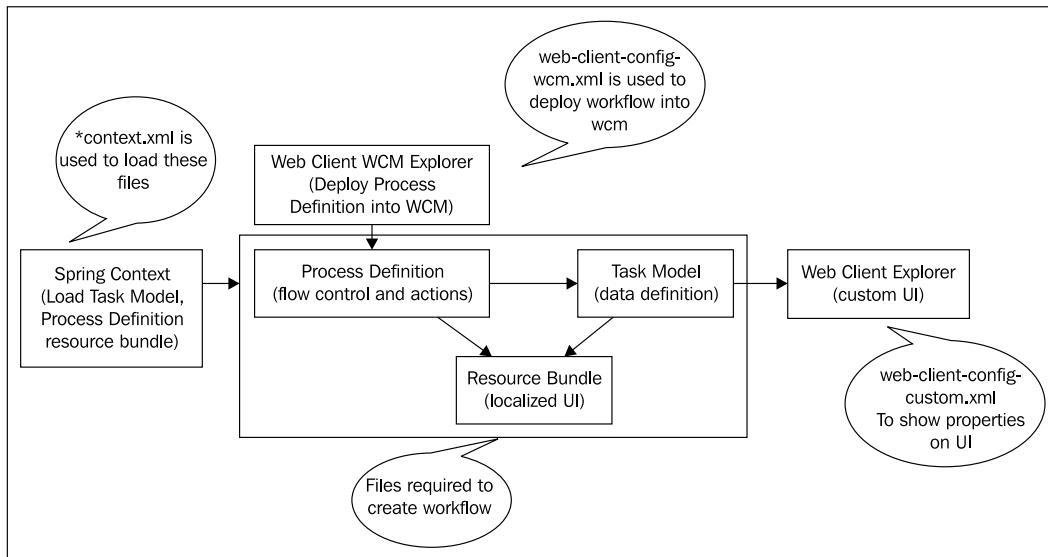
The custom workflow process is shown in the following diagram:



Defining the workflow process

For any workflow to be deployed you should have the following files:

1. **Task Model:** The Task Model provides a description for each of the tasks in the workflow. Each task description consists of Name, Title, Properties, Mandatory Aspects, and Association.
2. **Process Definition:** The Process Definition describes the states (steps) and transitions (choices) of a workflow.
3. **Resource Bundle (optional):** A workflow Resource Bundle provides all the human-readable messages displayed in the user interface for managing the workflow. Messages include task titles, task property names, task choices, and so on.
4. **web-client-config-custom.xml:** Web Client configuration specifies the presentation of Tasks and properties to the user in the Alfresco Explorer.
5. **custom-model-context.xml:** The custom model Spring Context file instructs Spring on how to bootstrap or load the Task Model definition file, Process Definition file, and Resource Bundle.
6. **web-client-config-wcm.xml:** Web Client configuration specifies the availability of workflow to the web project in the Alfresco Explorer.



Follow these steps to create a custom workflow.

Step 1: Create a Task Model

For each task in the Process Definition (as defined by <task> elements), it is possible to associate a task description. The description specifies information that may be attached to a task, that is properties (name and data type) associations (name and type of associated object), and mandatory aspects. A user may view and edit this information in the **Task** dialog within the Alfresco Explorer.

The Task Model is expressed as a Content Model, as supported by the Data Dictionary. To create a Task Model, create a new Content Model file for Process Definition with the .xml extension.

- Define a Content Model name
- Create a Type for each task
- Define Properties
- Define and add Aspects to a Type

Define a Content Model name

Create a new Content Model for the Process Definition. Define the namespace of the model. XML namespaces provide a method for avoiding element name conflicts. If you want to use any other model's task, aspect, or association, then you can use it by importing their namespace. Reusability of Task Model is possible.

```
<?xml version="1.0" encoding="UTF-8"?>
<model name="bookwcmwf:workflowmodel"
  xmlns="http://www.alfresco.org/model/dictionary/1.0">
  <imports>
    <import uri="http://www.alfresco.org/model/wcmworkflow/1.0"
      prefix="wcmwf" />
    <import uri="http://www.alfresco.org/model/bpm/1.0" prefix="bpm">
  </imports>
  <namespaces>
    <namespace uri="http://book.com" prefix="bookwcmwf" />
  </namespaces>
</model>
```

Create a Type for each task

For each task we have to define a Content Type. The Type can also be extended as follows:

```
<types>
  <type name="bookwcmwf:submitReviewTask">
    <parent>wcmwf:startTask</parent>
  </type>
</types>
```

Define Properties

Within each Type, describe the Properties and Associations (information) required for that task. Properties can also be inherited from other task definitions. Using the previous example all the properties of wcmwf: startTask will be added to this Type.

```
<type name="bookwcmwf:submitReviewTask">
    <parent>wcmwf:startTask</parent>
    <properties>
        <property name="wcmwf:submitReviewType">
            <title>Serial or Parallel Review</title>
            <type>d:text</type>
        </property>
    </properties>
    <associations>
        <association name="wcmwf:webproject">
            <source>
                <mandatory>false</mandatory>
                <many>false</many>
            </source>
            <target>
                <class>wca:webfolder</class>
                <mandatory>true</mandatory>
                <many>false</many>
            </target>
        </association>
    </associations>
</type>
```

Define Aspect

You can also introduce custom properties by defining an Aspect. An Aspect can be applied to any Content Type. Once applied, the properties are added to that Content Type.

You cannot define a dependency on other Aspects. They cannot be extended.

```
<type name="bookwcmwf:verifyBrokenLinksTask">
    <parent>wcmwf:workflowTask</parent>
    <mandatory-aspects>
        <aspect>bookwcmwf:reviewInfo</aspect>
        <aspect>bpm:assignee</aspect>
    </mandatory-aspects>
</type>
<aspects>
    <aspect name="bookwcmwf:reviewInfo">
```

```
<properties>
  <property name=" bookwcmwf:reviewerCnt">
    <title>Reviewer Count</title>
    <type>d:int</type>
    <mandatory>true</mandatory>
  </property>
</properties>
</aspect>
</aspects>
```

The following are the advantages of having custom Aspect over custom content:

- **Flexibility:** You will have more flexibility. Having a custom Aspect will give you the flexibility to add an additional set of properties to the documents in specific spaces.
- **Efficiency:** Since these properties are applied selectively to certain documents only in certain spaces, you will use limited storage in a relational database for these properties.

The following are the disadvantages of having custom Aspect over custom content:

- **High Maintenance:** If the custom Aspect (additional properties) is added to documents based on business rules, you need to define it at every space, wherever required.
- **Dependency:** You cannot define the dependency with other Aspects. For example, if you need the `effectivity` aspect to always be associated with the `custom` aspect, you need to make sure you attach both the Aspects to the documents.

Now that we are familiar with the code, let's develop a complete model file to deploy our case study in action.

For any customization of files you have to develop the files in the extension folder of `<install-alfresco>`. Create a file `book-serial-group-workflow-wcmModel.xml` in the specified location `<install-alfresco>/tomcat/shared/classes/alfresco/extension`. Copy the downloaded content into the file.



For reference, go to http://wiki.alfresco.com/wiki/Data_Dictionary_Guide#Content_Types.



Step 2: Create the Process Definition

A Process Definition represents a formal specification of a business process and is based on a directed graph. The graph is composed of nodes and transitions. Every node in the graph is of a specific Type. The Type of the node defines the runtime behavior. A Process Definition has exactly one Start-state and End-state.

The following table describes some of the key terms used in a Process Definition:

Key term	Description
Swimlane	Swimlane is used to define a role for a user.
Transition	Transitions have a source node and a destination node. The source node is represented by the property <code>from</code> and the destination node is represented by the property <code>to</code> . It is used to connect nodes. A Transition can optionally have a name. The name is represented in the UI with a button.
Task	Tasks are associated with a Swimlane. These tasks are defined in the Workflow model files. On the basis of these tasks, the Properties are displayed.
Actions	Actions are pieces of Java code that are executed upon events in the process execution. These actions are performed on the basis of these tasks, as defined in the Process Definition.
Events	The jBPM engine will fire Events during the graph execution. Events specify moments in the execution of the process. An Event can be task-create, node-enter, task-end, process-end, and so on. When the jBPM engine fires an event, the list of Actions is executed.
Scripts	Script is executed within Action. Some of the variables that can be available in Script are node, task, execution context, and so on.
Nodes	Each Node has a specific type. The Node Type determines what will happen when an execution arrives in the Node at runtime.

The following table summarizes the Node Types available in jBPM out of the box.

Node types	Description
Task Node	A Task Node represents one or more tasks that have to be performed by users.
Start-state	There can be only one Start-state in the Process Definition, which logs the start of the workflow.
decision	The distinction between multiple paths. When the decision between multiples path has to be taken, a decision node is used.
fork	A fork splits one path of execution into multiple concurrent paths of execution.

Node types	Description
join	Joins multiple paths into single path. A join will end every token that enters the join.
node	The node serves the situation where you want to write your own code in a node.
End-state	There can be only one End-state in the Process Definition, which logs the end of the workflow.

There are two ways of building the Process Definition. One is by hand, that is create a jPDL XML document. The second option is by designer, that is use a tool to generate the jPDL XML document. To create a Process Definition, create a new Process Definition file with the extension .xml.

Define a Process Definition name

The Process Definition name is important.

```
<?xml version="1.0" encoding="UTF-8"?>
<process-definition xmlns="urn:jbpm.org:jpd1-3.1"
    name="bookwcmwf:bookworkflow">
```

In the previous code we have used bookwcmwf :bookworkflow where bookwcmwf is the namespace of the workflow model file defined earlier, which we are going to use in this Process Definition, and bookworkflow can be any name.

Define a Swimlane

Swimlanes are used to declare workflow "roles". Tasks are associated with a Swimlane. Here initiator is the user who is starting the workflow. Likewise, we have some other roles also defined. For example, bpm_assignee (one user to whom the workflow is assigned), bpm_assignees (one or more user), bpm_groupAssignee (single group), and bpm_groupAssignees (one or more groups).

```
<swimlane name="initiator"/>

<swimlane name="approver">
    <assignment
        class="org.alfresco.repo.workflow.jbpm.AlfrescoAssignment">
            <pooledactors>#{bpm_groupAssignee}</pooledactors>
        </assignment>
    </swimlane>

<swimlane name="assignee">
    <assignment
        class="org.alfresco.repo.workflow.jbpm.AlfrescoAssignment">
            <actor>#{bpm_assignee}</actor>
        </assignment>
    </swimlane>
```

Associate a task

We have already defined task in the Content Model files. On the basis of these tasks the properties are displayed. Next step is to add these tasks to the workflow process. To start with, add a task to the start node. The Start Task is assigned to the initiator of the workflow. It's used to collect the information (that is the workflow parameters) required for the workflow to proceed.

```

<start-state name="start">
    <task name="bookwcmwf:submitReviewTask" swimlane="initiator"/>
        <transition name="" to="initialise"/>
</start-state>

<swimlane name="assignee">
    <assignment
        class="org.alfresco.repo.workflow.jbpm.AlfrescoAssignment">
        <actor>#{bpm_assignee}</actor>
    </assignment>
</swimlane>

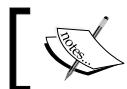
    <task-node name="initialise ">
        <task name="bookwcmwf:verifyBrokenLinksTask"
            swimlane="assignee" />
        <transition name="abort" to="end">
            <action
                class="org.alfresco.repo.workflow.jbpm.AlfrescoJavaScript">
                <script>
                    var mail = actions.create("mail");
                    mail.parameters.to =
                        initiator.properties["cm:email"];
                    mail.parameters.subject = "Adhoc Task " +
                        bpm_workflowDescription;
                    mail.parameters.from =
                        bpm_assignee.properties["cm:email"];
                    mail.parameters.text = "It's done";
                    mail.execute(bpm_package);
                </script>
            </action>
        </task-node>
    <end-state name="end"/>

```

During runtime, all the properties of the task bookwcmwf:submitReviewTask are visible to the user who is initiating a workflow. Once the properties are filled, the initiator assigns a task to another user or group. In this case, it is assigned to user. Now the task appears in dashlets of assigned user. The Assignee fills the properties of the task bookwcmwf:verifyBrokenLinksTask and clicks on the **abort** button. The abort transition would call Alfresco JavaScript that sends an e-mail. And an end-state event will log the end of the workflow.

We are now ready to create a Process Definition file and use the workflow model we developed earlier for our case study.

Create a file book-serial-group-processdefinition.xml in the specified location <install-alfresco>/tomcat/shared/classes/alfresco/extension. Copy the downloaded content into the file.



For reference go to <http://wiki.alfresco.com/wiki/WorkflowAdministration>.



Step 3: Create the workflow Resource Bundles

For localized workflow interaction it is necessary to provide Resource Bundles containing UI labels for each piece of text that is exposed to the user. With the appropriate Resource Bundles, a single workflow instance may spawn tasks where the user interface for each task is rendered in a different language, based on the locale of the user. Specific structure has to be followed in order to define labels for UI in Resource Bundle.

```
<model_prefix>_<model_name>. [title|description]
<model_prefix>_<model_name>. <model_element>. <element_prefix>_
    <element_name>. [title|description]
```

Add all the properties that relate to this Process Definition and model.

```
bookwcmwf_bookworkflow.workflow.title=Book Workflow
bookwcmwf_bookworkflow.node.verifybrokenlinks.transition.abort.
    title=Abort Submission
bookwcmwf_workflowmodel.type.bookwcmwf_reviewTask.description=
    Review Documents to approve or reject them
```

Create a file book-serial-group-messages.properties in the specified location, <install-alfresco>/tomcat/shared/classes/alfresco/extension. Copy the downloaded content into the file.

Step 4: Create the Alfresco Explorer Task dialogs

The custom web client configuration file contains information on how to display these custom Content Types, Aspects, and Associations. You need to make sure that the web client program recognizes this new custom aspect and displays it in the web-based interface. In order to make this happen, you need to configure the web client file, `web-client-config-custom.xml`, in the extension folder.

Open the `web-client-config-custom.xml` file from the specified location `<install-alfresco>/tomcat/shared/classes/alfresco/extension`. Copy the downloaded content into the file.

Step 5: Create a custom model Spring Context file

The custom model context file defines the Spring bean that will be used to bootstrap the definition of your custom Model, Workflow, and Resource Bundle. It lists one or more custom Model files, Workflow files, and Resource Bundles. When Spring starts up, it will instantiate this bean and will load your files from the disk.

Create a custom model context file and name the file as `<your-custom-model-name>-context.xml`, for example, `bookWorkflowModel-context.xml`. Create the file in the specified location `<install-alfresco>/tomcat/shared/classes/alfresco/extension`. Copy the downloaded content in the file.



Download the complete code samples from the Packt website. It is very important for you to note that the Alfresco server recognizes context files.



Step 6: Deploy into WCM project

In order to identify this workflow for WCM, open the `web-client-config-wcm.xml` file from the specified location `<install-alfresco>/tomcat/webapps/alfresco/WEB-INF/classes/alfresco` and insert the highlighted XML code within the `workflows` tag, as follows:

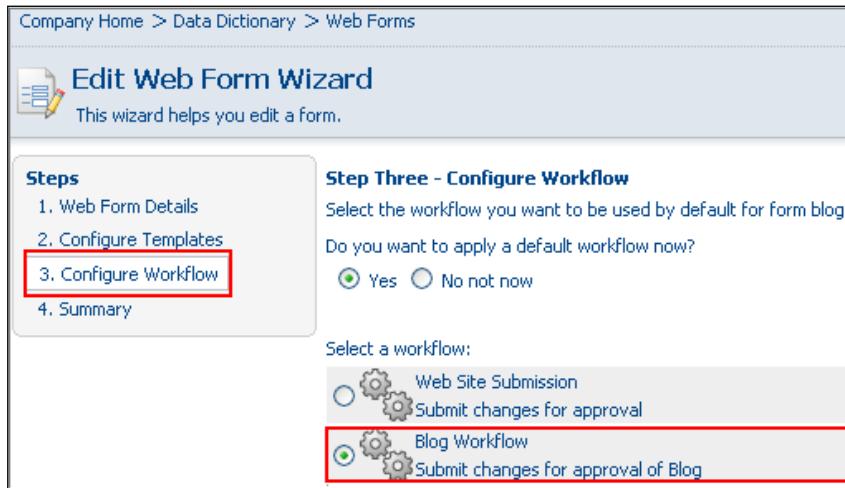
```
<workflows>
    wcmwf:submit ,bookwcmwf:bookworkflow
</workflows>
```

Test the workflow

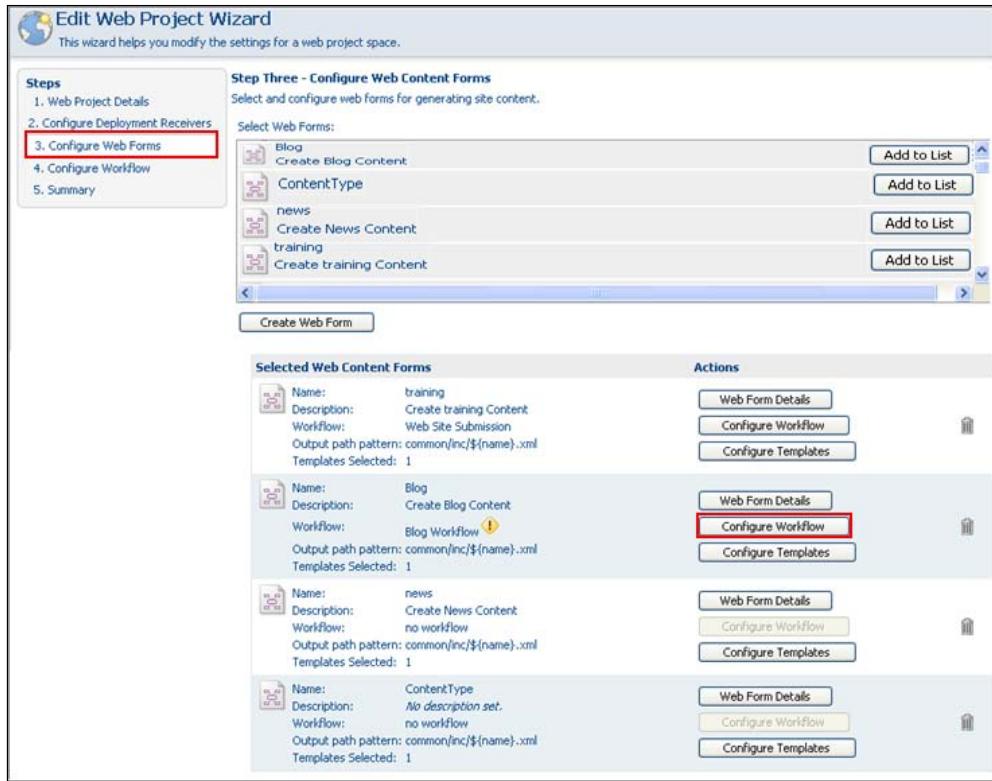
Now that we have completed workflow implementation, let's test the workflow.

Follow the steps below to test workflow.

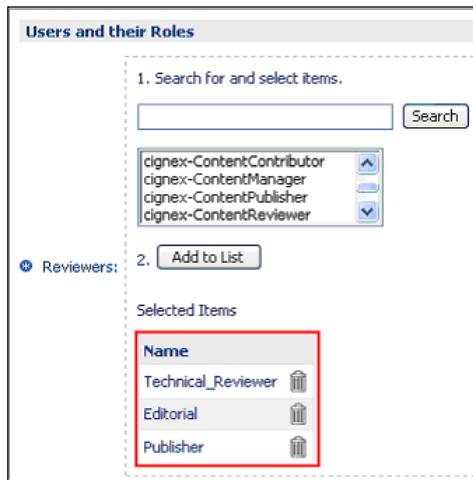
1. Refer to the *Associating workflows to web forms* section for how to configure a workflow for Blog web form. You will notice one more workflow is added. Follow the steps as mentioned in the specified section.



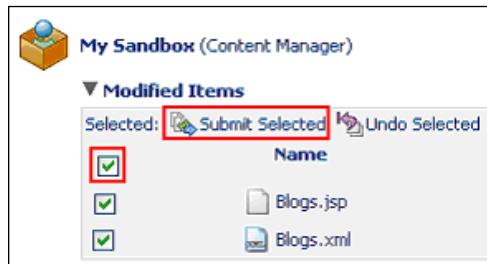
2. Go to **Company Home | Web Projects | Cignex**.
3. Select **Edit Web Project Settings** from the action menu.
4. Click on **Next**.
5. On the next screen, click on **Next** again.
6. In the **Step Three** window, you will notice the added web forms in the panel as shown in the following screenshot. You can see the **Configure Workflow** button available for the web form. This button is enabled only for those web forms for which we have configured workflows. Notice the attention icon next to the workflow. This indicates a workflow has been selected but not configured.



7. Configure the workflow and assign it to three groups as shown in the next screenshot. This workflow is a serial one. So it will go to the groups, one by one, only in the series **Technical Reviewer**, **Editorial**, and **Publisher**.



8. Click on **Finish**.
9. Log in as user **Mark**. Submit the Blog to the Staging box. We will also see how the launch date functions. Click on **OK**.



10. On submit, the **Submit Items** wizard will open as shown in the following screenshot. You can select the launch date as shown in the screenshot. If you want to make changes in the configuration of a workflow you can click on **Configure Workflow** as well. Click on **OK**:

Submission Info

Label: Publishing Blog
Description: Publish Blog

Auto Deploy (This will deploy the changes from this submission upon approval.)

Workflow

Use the following workflow to submit all modified items

Blog Workflow(Submit changes for approval of Blog) Configure Workflow

Content Launch

Launch Date: 12 December 2009 15 : 54 Today None

Content Expiration

Set expiration date for all modified items.
Note: To change individual expiration dates click the 'Change Expiration Date' action icon.

Modified Items

The following items will be submitted

Name	Description	Path	Modified Date	Expiration Date	Actions
Blogs.jsp	Rendered by blog into /ROOT/common/inc/Blogs.jsp	/ROOT/common/inc/Blogs.jsp	12 December 2009 14:45		
Blogs.xml				/ROOT/common/inc/Blogs.xml 12 December 2009 14:45	

11. Log in as a user of group **Technical Reviewer**. Consider we have logged in as **Jennifer Bruce**. Configure the **My Pooled Tasks** dashlet. This dashlet is required for group workflows. Repeat the same steps for **Kristie Dawid**. For now, both the users will be able to see the task in their dashlet.

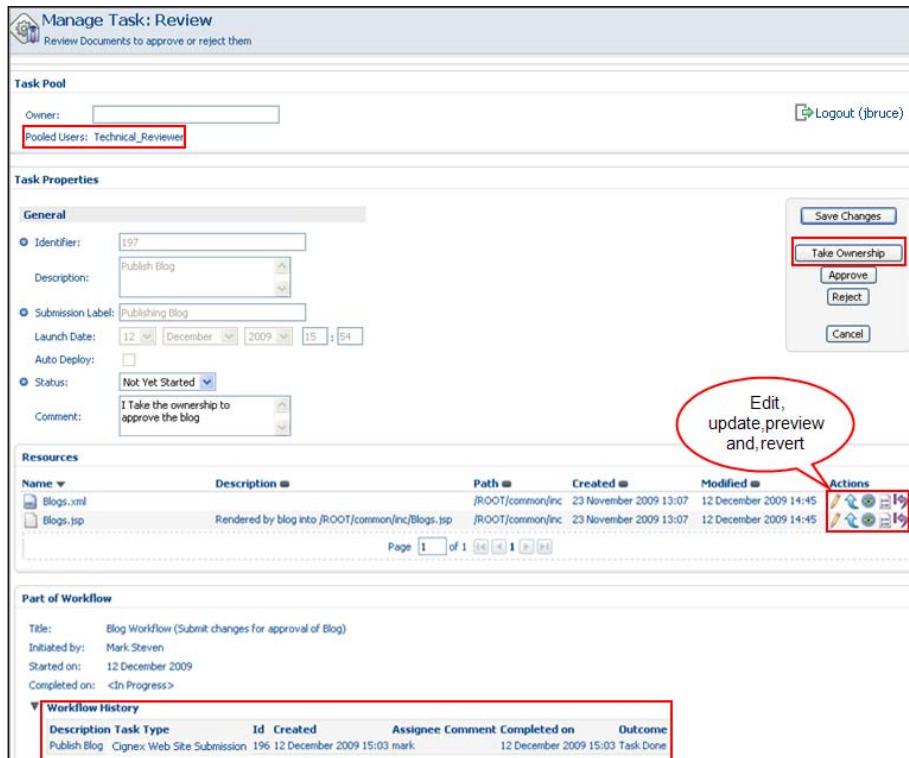
[ No other group can see the task, as this workflow is for serial flow. A group will see the task in the order they are assigned.]



Description	Type	Id	Created	Due Date	Status	Priority	Actions
Publish Blog	Review	197	12 December 2009 15:03		Not Yet Started	3	

Page 1 of 1

12. Any one of the users of the group has to take ownership. Click on **Take Ownership**.



Manage Task: Review
Review Documents to approve or reject them

Task Pool
Owner:
Pooled Users: **Technical_Reviewers**

Task Properties
General
 Identifier: 197
 Description: Publish Blog
 Submission Label: Publishing Blog
 Launch Date: 12 December 2009 15:54
 Auto Deploy:
 Status: Not Yet Started
 Comment: I Take the ownership to approve the blog

Take Ownership

Resources
Name Description Path Created Modified Actions
 Blogs.xml
 Blogs.jsp
Rendered by blog into /ROOT/common/inc/blogs.jsp /ROOT/common/inc 23 November 2009 13:07 12 December 2009 14:45

Part of Workflow
Title: Blog Workflow (Submit changes for approval of Blog)
Initiated by: Mark Steven
Started on: 12 December 2009
Completed on: <In Progress>

Workflow History
Description Task Type Id Created Assignee Comment Completed on Outcome
Publish Blog Cignex Web Site Submission 196 12 December 2009 15:03 mark 12 December 2009 15:03 Task Done

13. You will notice that the task now appears in the **My Task To Do** dashlet.

The screenshot shows the 'My Alfresco Dashboard' interface. In the center, there is a table titled 'My Tasks To Do'. The first row of the table has a red box around the 'Type' column, which contains the value 'Review'. The table includes columns for Description, Type, Id, Created, Due Date, Status, Priority, and Actions. Below the table is a page navigation bar showing 'Page 1 of 1'.

14. Open the **Manage Task** dialog. You are also given the facility to leave the ownership. Click on **Return To Pool**.

The screenshot shows the 'Manage Task: Review' dialog. The 'General' tab is active, displaying the following information:

- Identifier:** 197
- Description:** Publish Blog
- Submission Label:** Publishing Blog
- Launch Date:** 12 December 2009 15:03
- Status:** Not Yet Started
- Comment:** I dont want to take the ownership

On the right side of the dialog, there is an 'Actions' panel with buttons for Save Changes, Return to Pool (highlighted with a red box), Approve, Reject, and Cancel.

Below the General tab, there are sections for Resources and Part of Workflow, each containing tables with data. At the bottom, there is a 'Workflow History' table.

Description	Task Type	Id	Created	Assignee	Comment	Completed on	Outcome
Publish Blog	Cignex : Web Site Submission	196	12 December 2009 15:03	mark			Task Done

15. You will find again that the task appears in **My Pooled Task**. This gives the advantage for both the users to take the ownership and approve the task.
16. Consider that **Jeniffer Bruce** has taken the ownership and approved the task. As soon as ownership is taken, the task disappears from the other two users.
17. Once approved by users of the **Technical Reviewer** group, log in as user **LeRoy Fuess** of the **Editorial** group and approve the items.

Manage Task: Review
Review Documents to approve or reject them

Task Pool

Owner: []
Pooled Users: Editorial

Task Properties

General

- Identifier: 198
Description: Publish Blog
- Submission Label: Publishing Blog
Launch Date: 12 December 2009 15:54
Auto Deploy:
- Status: Not Yet Started
Comment: I too approve for the blog
- Reviewers: Technical_Reviewer
Editorial
Publisher

Resources

Name	Description	Path	Created	Modified	Actions
Blogs.xml		/ROOT/common/inc	23 November 2009 13:07	12 December 2009 15:03	
Blogs.jsp	Rendered by blog into /ROOT/common/inc/Blogs.jsp	/ROOT/common/inc	23 November 2009 13:07	12 December 2009 15:03	

Page 1 of 1

Part of Workflow

Title: Blog Workflow (Submit changes for approval of Blog)
Initiated by: Mark Steven
Started on: 12 December 2009
Completed on: <In Progress>

Workflow History

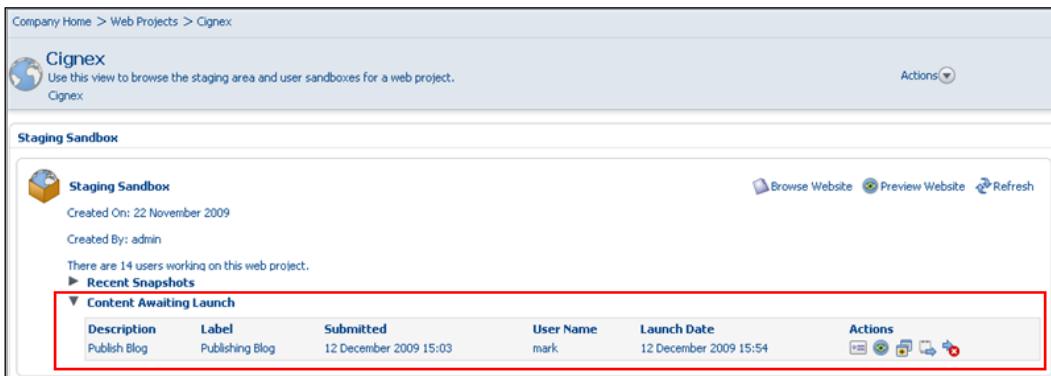
Description	Task Type	Id	Created	Assignee	Comment	Completed on	Outcome
Publish Blog	Review	203	12 December 2009 16:21	jbruce		12 December 2009 15:03	Approve
Publish Blog	Cignex Web Site Submission	202	12 December 2009 16:21	mark		12 December 2009 15:38	Task Done

18. Once approved by **LeRoy Fuess** of the **Editorial** group, log in as **Michael Alison** of the **Publisher** group and approve the items (you can log in with any user of the group, take ownership, and approve the task).
19. After being approved by all the groups, the task appears in the **My Task To Do** dashlet of **Mark**, who has initiated the process for submitting the content. The content is not submitted yet, as the Launch date is **12 December 2009 15:54**.

My Tasks To Do								
Description	Type	Id	Created	Due Date	Status	Priority	Actions	
Publish Blog	Submission Pending	200	12 December 2009 15:42	12 December 2009	Not Yet Started	3		

Page 1 of 1

20. You will find the content in the **Staging Sandbox** in **Content Awaiting Launch** item. The content will be automatically submitted on **12 December 2009 15:54**.



Company Home > Web Projects > Cignex

Cignex
Use this view to browse the staging area and user sandboxes for a web project.
Cignex Actions

Staging Sandbox

Staging Sandbox																	
	Created On: 22 November 2009																
Created By: admin																	
There are 14 users working on this web project.																	
Content Awaiting Launch <table border="1"> <thead> <tr> <th>Description</th> <th>Label</th> <th>Submitted</th> <th>User Name</th> <th>Launch Date</th> <th>Actions</th> </tr> </thead> <tbody> <tr> <td>Publish Blog</td> <td>Publishing Blog</td> <td>12 December 2009 15:03</td> <td>mark</td> <td>12 December 2009 15:54</td> <td> </td> </tr> </tbody> </table>						Description	Label	Submitted	User Name	Launch Date	Actions	Publish Blog	Publishing Blog	12 December 2009 15:03	mark	12 December 2009 15:54	
Description	Label	Submitted	User Name	Launch Date	Actions												
Publish Blog	Publishing Blog	12 December 2009 15:03	mark	12 December 2009 15:54													

21. Open the **Manage Task** dialog. You can either **Abort Submission** or **Submit Now**. If you click on **Submit Now**, it will submit the content to the Staging box. On clicking **Abort Submission**, the content will not be submitted.

My Alfresco

Manage Task: Submission Pending

Submission Pending

Task Properties

General

- Identifier: 200
- Description: Publish Blog
- Submission Label: Publishing Blog
- Launch Date: 12 December 2009 15:54
- Auto Deploy:
- Comment:

Review Status

- Type of Review: Serial
- Reviewers: Mark Steven [mark]
- Total Approved: 0

Review Status

- Type of Review: Serial
- Reviewers: Mark Steven [mark]
- Total Approved: 0

Resources

Name	Description	Path	Created	Modified	Actions
Blogs.xml		/ROOT/common/inc	23 November 2009 13:07	12 December 2009 15:24	
Blogs.jsp	Rendered by blog into /ROOT/common/inc/Blogs.jsp	/ROOT/common/inc	23 November 2009 13:07	12 December 2009 15:24	

Page 1 of 1

Part of Workflow

Title: Blog Workflow (Submit changes for approval of Blog)
 Initiated by: Mark Steven
 Started on: 12 December 2009
 Completed on: <In Progress>

Workflow History

Description	Task Type	Id	Created	Assignee	Comment	Completed on	Outcome
Publish Blog	Review	199	12 December 2009 15:42	malison		12 December 2009 15:42	Approve
Publish Blog	Review	198	12 December 2009 15:38	fuss		12 December 2009 15:42	Approve
Publish Blog	Review	197	12 December 2009 15:03	jbruce		12 December 2009 15:38	Approve
Publish Blog	Cignex Web Site Submission	196	12 December 2009 15:03	mark		12 December 2009 15:03	Task Done

22. On clicking Abort Submission, you will notice the task in My Task To Do.

My Tasks To Do

Description	Type	Id	Created	Due Date	Status	Priority	Actions
Publish Blog	Submission Aborted	216	12 December 2009 18:54	12 December 2009	Not Yet Started	3	

Page 1 of 1

23. Open the task and click on **Task Done**. After that you can see the files again in the **Modified Items** section.



The screenshot shows the 'My Sandbox (Content Manager)' interface. At the top, there are navigation links: 'Browse Website', 'Preview Website', 'Submit All', 'Undo All', and 'More Actions'. Below this is a section titled 'Modified Items' with a dropdown arrow. Underneath are buttons for 'Selected': 'Submit Selected' and 'Undo Selected'. A table lists three items:

	Name	Created Date	Modified Date	Size	Actions
<input type="checkbox"/>	Blogs.jsp	23 November 2009 13:07	12 December 2009 18:43	3.35 KB	
<input type="checkbox"/>	Blogs.xml	23 November 2009 13:07	12 December 2009 18:43	3.75 KB	

24. If you have to submit a Blog, start the process again.

Epiring content in WCM

For any changes promoted to the site, specific expiration dates can be set on a global or asset-by-asset basis. Expiration is the only indication that the content is expired. The content is not disabled or hidden in the Staging Sandbox. Upon expiration, end users are automatically assigned the asset as a task so that they can determine whether the asset should be updated or removed from the site.

The repository checks periodically for expired items. When any expired items are found they are added to a workflow and sent to the user that last modified the asset. If multiple items are destined for the same user, they are batched up into a workflow for each website the asset belongs to.

Clicking on the task will launch the **Manage Task** dialog that lists all of the expired items, from here the assigned user can perform the relevant action on each item. These changes occur in isolation from the user's sandbox, thus not affecting any work that they may currently be doing in their sandbox. The workflow, however, is layered over the user's sandbox so any changes that they have made in their sandbox will be visible.

Configuration

Configuration for content expiration is in one file—`scheduled-jobs-context.xml`. You'll find the file in the Alfresco package, that is `tomcat-home/webapps/alfresco/WEB-INF/classes/alfresco /scheduled-jobs-context.xml`. It contains the configuration for the frequency of expired item checks. By default it is set to check once a day at 3:30 a.m. as can be seen in the following cron expression:

```

<bean id="avmExpiredContentTrigger" class="org.alfresco.util.
CronTriggerBean">
    <property name="jobDetail">
        <bean id="avmExpiredContentJobDetail"
            class="org.springframework.scheduling.quartz.JobDetailBean">
            <property name="jobClass">
                <value>org.alfresco.repo.avm.AVMExpiredContentJob</value>
            </property>
            <property name="jobDataAsMap">
                <map>
                    <entry key="expiredContentProcessor">
                        <ref bean="avmExpiredContentProcessor" />
                    </entry>
                </map>
            </property>
        </bean>
    </property>
    <property name="scheduler">
        <ref bean="schedulerFactory" />
    </property>
    <!-- trigger at 3:30am each day -->
    <property name="cronExpression">
        <value>0 30 3 * * ?</value>
    </property>
</bean>

```

You can make the changes for cron expression and check the expiration date functionality.

Changes can be also made directly to these files, but better practice would suggest to make changes in `content-expiration-debug-context.xml.sample` located at `<install-alfresco>/tomcat/shared/classes/alfresco/extension/`.

Rename the file to `content-expiration-debug-context.xml`.

Paste the previous code with changes in cron expression as suggested in the following:

```

<property name="cronExpression">
    <value>0 3 * * ?</value>
</property>

```

This will run the scheduler in every three minutes and will check for the expired item.

WCM Workflows

Submit any item and apply expiration date as mentioned in the following screenshot:

My Alfresco > Cignex

Submit Items

This page helps you to submit modified items for publishing on the website.

Submission Info

* Label: Test Expiry Date

* Description: Test Expiry Date

Auto Deploy (This will deploy the changes from this submission upon approval.)

Content Expiration

14 December 2009 15 : 45 Today None Apply To All

Note: To change individual expiration dates click the 'Change Expiration Date' action icon.

Modified Items

The following items will be submitted

Name	Description	Path	Modified Date	Expiration Date	Actions
index.jsp	/ROOT/index.jsp	14 December 2009 15:40	14 December 2009 15:45		

Page 1 of 1

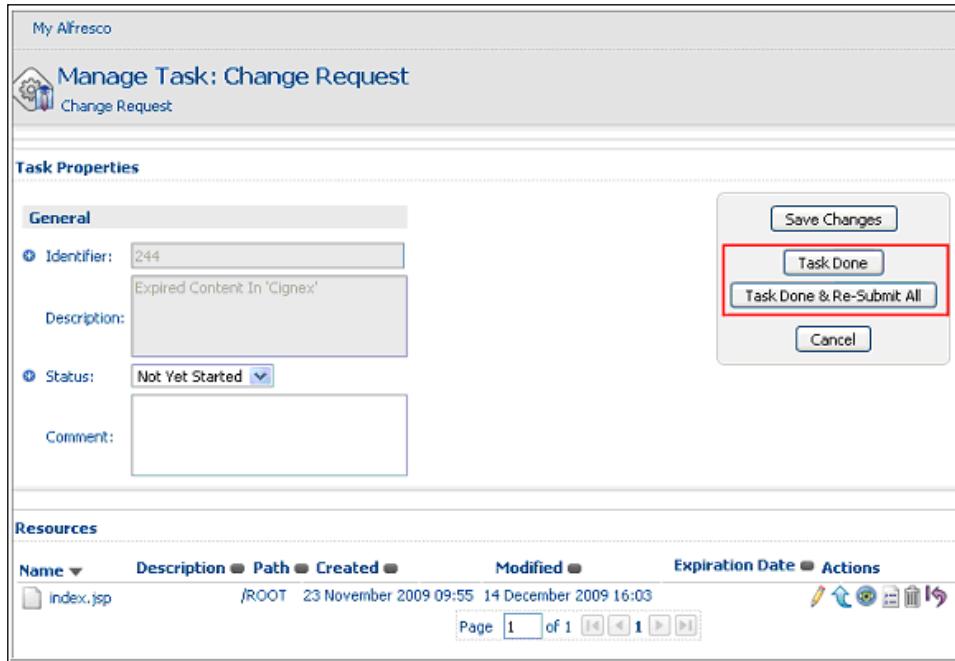
Process the workflow as explained in the previous sections.

After **14 December 2009 15:45**, you will find a task in the **My Task To Do** dashlet as shown in the following screenshot:

My Tasks To Do

Description	Type	Id	Created	Due Date	Status	Priority	Actions
Expired Content In 'Cignex'	Change Request	244	14 December 2009 16:03		Not Yet Started	3	

Open the **Manage Task** dialog. The dialog displays two buttons as shown in the following screenshot. Both signal that all changes have been made and will therefore apply the changes to the user's sandbox. The bottom button, **Task Done & Re-Submit All**, will immediately launch the **Submit** dialog so that the users can submit their changes. Whereas the **Task Done** button will just apply the changes and users can then pick and choose when and which items to submit.



Summary

Workflows are an important aspect in Alfresco WCM. Alfresco web project uses workflows to support any set of changes, either automated or user-driven steps, in a business process before final commit to the Staging Sandbox. In this chapter we have learned the following points:

- Alfresco includes two flows of workflows out of the box. One is serial oriented and the other one is the parallel workflow.
- Auto review of date for submission can be made using launch and expiry date.
- Workflow can be configured for a web form and web project.
- Whenever content is submitted, a snapshot is automatically taken of staging to provide an archive of the current version of the site.

6

Dynamic Deployment and Customizations

The dynamic model feature enables dynamic customization of models without requiring a restart of the Alfresco server, and is also applicable to a multi-tenant environment. This also includes dynamic reloading of the web client UI customizations. Customizations of workflows can be done easily.

By the end of this chapter you will have learned:

- The advantages of workflow
- How to enable dynamic customization of workflow without requiring a restart of the Alfresco server
- The process of customizing an existing workflow
- How to remove workflow for a specific staging submission
- How to configure a ZERO workflow
- How to implement workflow viewer to see the pending list of workflows

Dynamic deployment

You are able to dynamically deploy workflow in Alfresco WCM by using workflow types, workflow definitions, web client configurations, and property files. Since version 3.0, Alfresco supports dynamic deploy of models, workflows, messages, and web-client configuration changes to facilitate dynamic customization without requiring a restart of the Alfresco server. Let's discuss how to dynamically deploy models, workflows, messages, and web-client changes.

The following are the advantages of dynamic deployment:

- There is no need to restart the Alfresco server every time you make changes to the files.
- Both the content as well as the files are stored in the repository. It is easier to maintain and move content along with the files.
- You can activate and deactivate the dynamic models by keeping the model XML file in the repository.
- You can deploy and undeploy the workflow definitions as and when required.
- In a multi-tenant setup, the files that are defined in the <extension> folder are available to all tenants. If you would like to customize workflow for a specific tenant only, then a dynamic deployment is the best choice.

Dynamic models

Dynamic models are nothing but XML-based model files. Creating a dynamic custom model is the same as creating a regular custom model. For this example, create a custom content type that has two properties. Create a file named `dynamicWCMWorkflowModel` on your personal computer with the downloaded content.



Download the complete code sample
from Packt's website.



Deploying a model file

The Task Model can also be dynamically deployed without restarting the server.

This can be achieved by two ways:

- Uploading or creating the model file in the new 'Models' space (**Company Home | Data Dictionary | Models**).
- Using pre-registered URLs provided by the Alfresco Repository Admin Console.

First approach

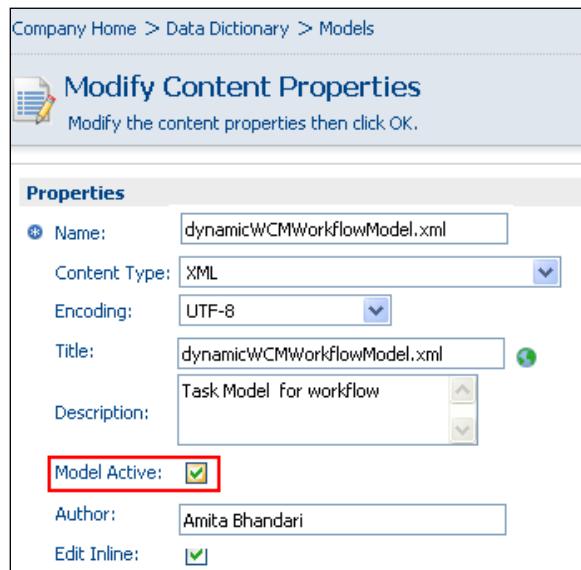
Dynamic workflow models are stored in the new 'Models' space (**Company Home | Data Dictionary | Models**).

Upload a custom XML model file, `dynamicWCMWorkflowModel`, to the 'Models' space. By default, the model will not be active unless the **Model Active** checkbox is selected during the upload. To activate a previously inactive model, select **View Details** and then select the **Modify** properties icon. In the **Modify Content Properties** page, select the **Model Active** checkbox.

To deactivate a model, select **View Details** and then select the **Modify** properties icon. In the **Modify Content Properties** page, unselect the **Model Active** checkbox.

Follow these steps to deploy a model dynamically:

1. Go to **Company Home | Data Dictionary | Models**.
2. In the header click on **Add Content**.
3. The **Add Content Wizard** is displayed. Upload the custom XML file.
4. In the **Name** textbox type `dynamicWCMWorkflowModel`.
5. Click on **OK**.
6. The **Modify Content Properties** dialog is displayed. Check the **Model Active** property, as shown in the following screenshot:



7. Click on **OK**.

To verify the changes, log out and log in if required.

Updating a custom model

You can directly edit or update the XML model file. If the model is active, then it will be reloaded. If the file is checked out, then the working copy will be ignored until such a time as the file is checked in.

Second approach

Deployment of model files can be achieved as an administrator using the URL
`http://<server-name>:<port>/alfresco/faces/jsp/admin/reloadadmin-console.jsp`

You can use the following commands to activate or deploy the model.

`activate model dynamicWCMWorkflowModel.xml`: This command is used to set the repository model to active and load into runtime data dictionary.

`deploy model alfresco/extension/dynamicWCMWorkflowModel.xml`: This command uploads the model to the repository and loads it into runtime data dictionary. This will also set the repository model as active.

Dynamic Resource Bundles

Creating a Dynamic Resource Bundle is the same as creating a regular property file. For this example, create labels for two properties that are defined in the model file. Create a file named `dynamicWCMWorkflowMessages` on your personal computer with the following content:

```
dynamic_processTask.workflow.title=Dynamic Workflow
dynamic_processTask.workflow.description=Dynamic Workflow for approval
```

Deploying a Resource Bundle

The Resource Bundle can also be dynamically deployed without restarting the server.

This can be achieved by two ways:

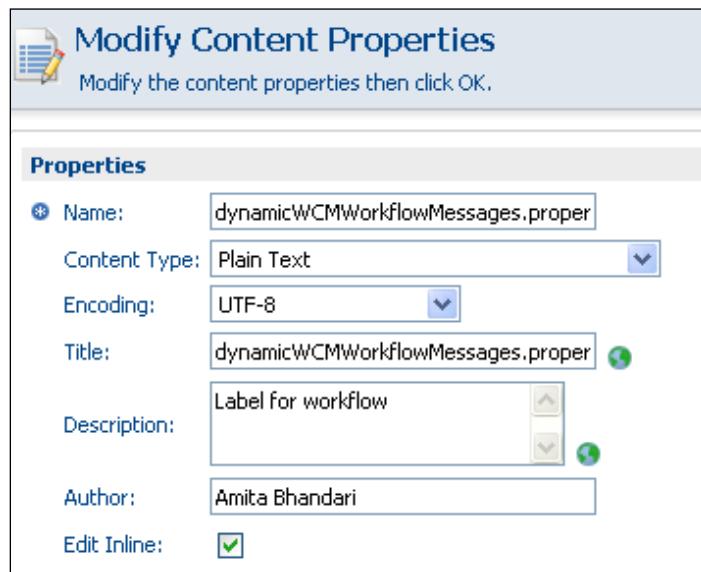
- Uploading or creating the message file in the new **Messages** space (**Company Home | Data Dictionary | Messages**).
- Using the Repository Admin Console provided by Alfresco.

First approach

The associated message Resource Bundles are stored in the new **Messages** space (**Company Home | Data Dictionary | Messages**). Upload the custom Resource Bundle by uploading each of the message property files (for all locales) to the **Messages** space. The messages will not be applied until they are explicitly reloaded or when the server is restarted.

Follow these steps to create a message:

1. Go to **Company Home | Data Dictionary | Messages**.
2. In the header, click on **Add Content**.
3. The **Add Content Wizard** is displayed. Upload the custom property file.
4. In the **Name** textbox type **dynamicWCMWorkflowMessages.properties**.
5. Click on **OK**.



The custom configuration will not be applied until it is explicitly reloaded (refer to the following section) or the server is restarted.

Reloading the Resource Bundle

If the Resource Bundle file has been added, edited, or updated, it can be dynamically reloaded by using the Alfresco Repository Admin Console via:

```
http://<server-name>:<port>/alfresco/faces/jsp/admin/reloadadmin-console.jsp
```

This has a single command, `reload`, which will cause the Resource Bundle to be reloaded.

```
reload messages dynamicWCMWorkflowMessages
```

Updating a Resource Bundle

You can directly edit or update the Resource Bundle file. Use the `reload` command to reflect the changes. If the file is checked out, then the working copy will be ignored until such a time as the file is checked in.

Second approach

They can be dynamically reloaded by using the Alfresco Repository Admin Console via: `http://<server-name>:<port>/alfresco/faces/jsp/admin/reloadadmin-console.jsp`.

The command `deploy messages <resource bundle base name>` will cause the message resource to be re-registered.

```
deploy messages alfresco/extension/dynamicWCMWorkflowMessages
```

Dynamic workflows

Dynamic workflows are nothing but XML-based Process Definition files. Creating a dynamic Process Definition is the same as creating a regular Process Definition file. For this example, create a custom Process Definition. Create a file named `dynamicProcessDefinition` on your personal computer with the downloaded content.



Download the code samples from
Packt publisher's book website.



Deploying a Process Definition file

The Process Definition can also be dynamically deployed without restarting the server.

This can be achieved by three ways:

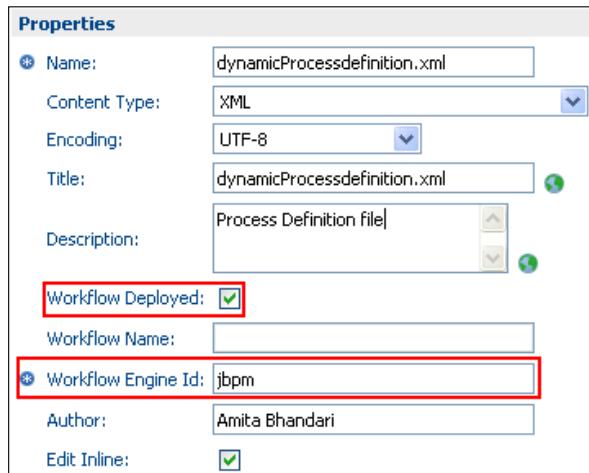
- Uploading or creating the Process Definition file in the new Workflow Definitions space (**Company Home | Data Dictionary | Workflow Definitions**).
- Using JBoss jBPM Process Designer tool.
- Using pre-registered URLs provided by Alfresco workflow console.

First approach

The Process Definitions are stored in the new **Workflow Definitions** space (**Company Home | Data Dictionary | Workflow Definitions**). Upload a custom XML Process Definition file to the **Workflow Definitions** space. By default, the Process Definition will not be deployed unless the **Workflow Deployed** checkbox is selected during the upload.

Follow these steps to create a Process Definition:

1. Go to **Company Home | Data Dictionary | Workflow Definitions**.
2. In the header click on **Add Content**.
3. The **Add Content Wizard** is displayed. Upload the custom XML file.
4. In the **Name** textbox type **dynamicProcessdefinition.xml**. Click on **OK**.
5. Select the **Workflow Deployed** checkbox. By default, the Process Definition will not be deployed unless the **Workflow Deployed** checkbox is selected during the upload.
6. In the **Workflow Engine Id** type **jBPM**. The completed definition is shown in the following screenshot:



7. Click on **OK**.

To verify the changes, log out and log in.

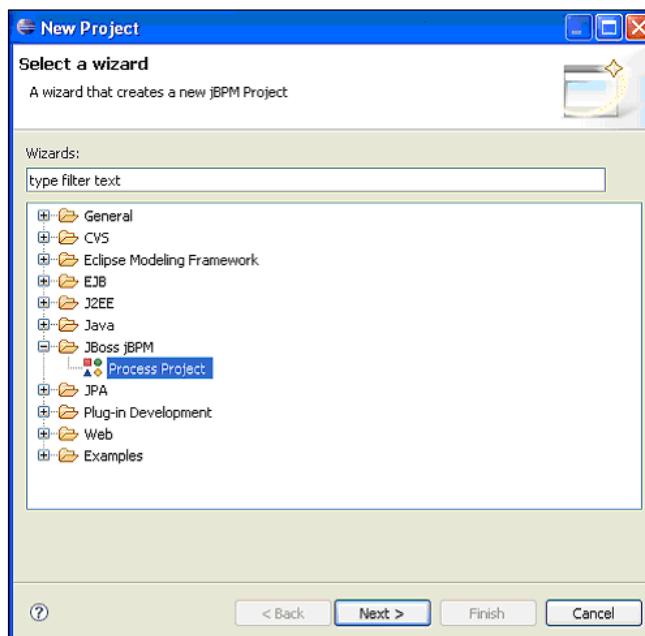
To undeploy a Process Definition, select **View Details** and then select the **Modify properties** icon. In the **Modify Content Properties** page, unselect the **Workflow Deployed** checkbox.

Second approach

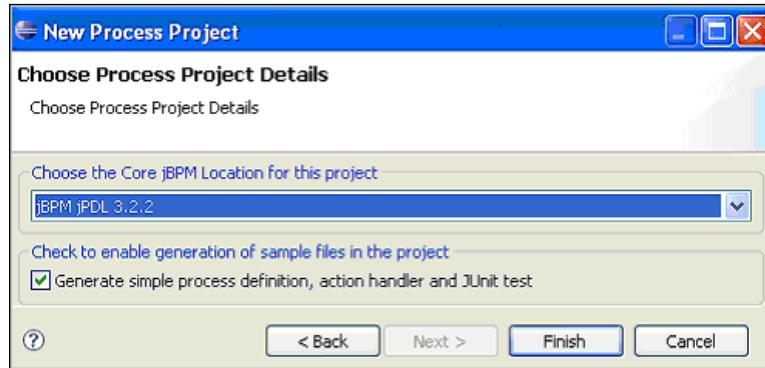
JBoss jBPM also includes a graphical designer tool for authoring business processes. The most important feature of the graphical designer tool is that it includes support for both the tasks—that of the business analyst as well as the technical developer. This enables a smooth transition from business process modeling to the practical implementation.

Follow these steps for configuring the jBPM:

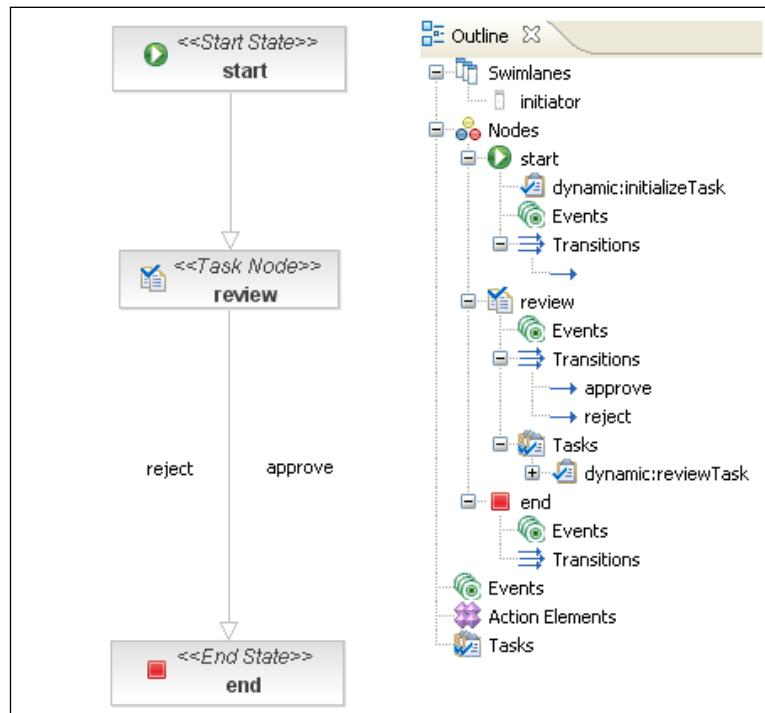
1. To implement this you have to download jBPM plugins from:
<http://sourceforge.net/projects/jbpm/files/jBPM%20Process%20Designer/jbpm-jpd1-designer-3.1.7>
2. Once the installation is over, restart Eclipse.
3. Go to **File | New | Project**. Expand **jBoss JBPM** and select **Process Project** as shown in the following screenshot:



4. Enter the **Project Name** and click on **Next**.
5. Select the jBPM location as mentioned in the following screenshot and click on **Finish**:



6. Copy the Process Definition code and paste it in the `src/main/jpd` package.
7. The process should look like the next screenshot:

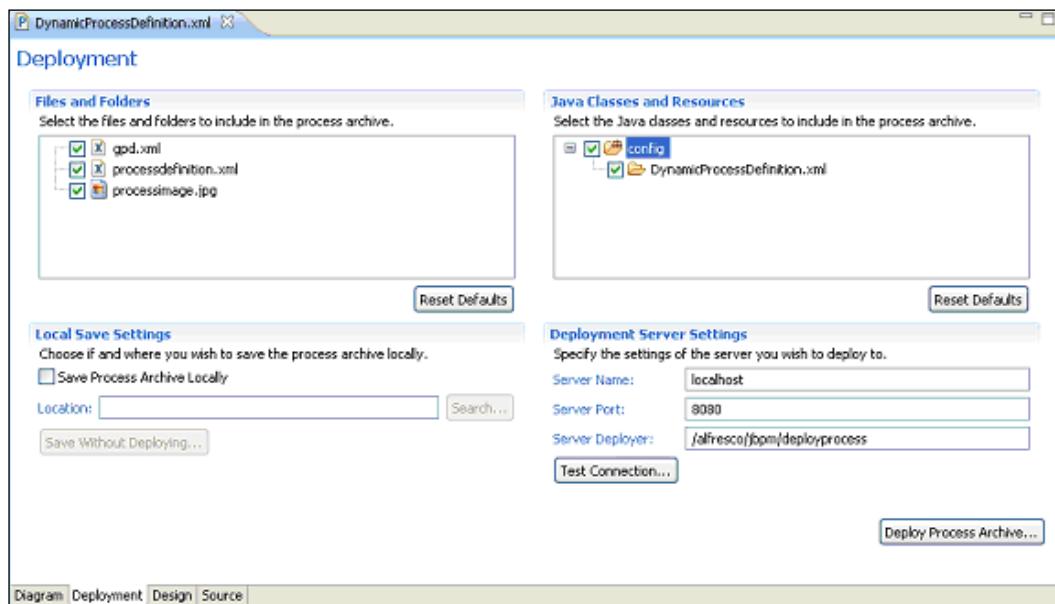


8. Deploy it as mentioned ahead.

Following are the steps to be followed to deploy a workflow via JBoss jBPM Process Designer. This will deploy workflow without restart of the server.

1. Ensure the Alfresco server is up and running.
2. Click on the **source** tab and copy the content of the file **dynamicProcessdefinition**.
3. Click on the **Deployment** tab and enter the following information:
 - **Server Name** = machine name where Alfresco is installed
 - **Server Port** = port number assigned to Alfresco (default: 8080)
 - **Server Deployer** = /alfresco/jbpm/deployprocess
4. Click on **Test Connection....**
5. If all is **OK**, click on **Deploy Process Archive....**

The following screenshot shows the details provided earlier:



Display of workflow images

If Hot Deployment is done, then we can see the workflow images that show the current status of workflow in Alfresco Explorer under the workflow outline section. The screenshots can be seen in the following sections when you test the workflows that are deployed through jPDL.

Modify these files:

- /jsp/workflow/start-workflow-wizard/workflow-options.jsp
- /jsp/workflow/manage-task-dialog.jsp

Change the rendered and expanded value to true as highlighted.

The modified code is as follows:

```
<a:panel rendered="true" id="workflow-outline"
label="#{msg.workflow_outline}" progressive="true" expanded="true"
border="white" bgcolor="white" titleBorder="lbgrey"
expandedTitleBorder="dotted" titleBgcolor="white"
styleClass="mainSubTitle">
```

Third approach

The workflow console can be used (as an alternative to the Alfresco Explorer) to deploy and undeploy Process Definitions. Its primary use is to test newly-developed workflow definitions. However, it also supports the debugging/diagnosis of current "in-flight" workflows. This can be deployed via `http://<server-name>:<port>/alfresco/faces/jsp/admin/workflow-console.jsp`.

The command `deploy <workflow-name>` will cause the workflow to be deployed.

```
deploy alfresco/extension/dynamicProcessdefinition.xml
```

Dynamic Alfresco Explorer

A dynamic web client configuration file will have the same name as a custom web client configuration file. For example, create a `web-client-config-custom.xml` file, with the following content, to support the dynamic content type that you created earlier:

```
<config evaluator="node-type" condition="dynamic:initializeTask"
    replace="true">
    <property-sheet>
        <separator name="sep2" display-label-id="users_and_roles"
            component-generator="HeaderSeparatorGenerator" />
        <show-property name="dynamic:property" />
    </property-sheet>
</config>
```

```
<show-association name="bpm:assignee"
    display-label-id="wf_reviewers" />
</property-sheet>
</config>
<config evaluator="node-type" condition="dynamic:reviewTask"
    replace="true">
    <property-sheet>
        <separator name="sep1" display-label-id="general"
            component-generator="HeaderSeparatorGenerator" />
        <show-property name="bpm:taskId" />
        <show-property name="bpm:description"
            component-generator="TextAreaGenerator" read-only="true"/>
        <show-property name="bpm:comment"
            component-generator="TextAreaGenerator" />
        <separator name="sep2" display-label-id="wf_reviewers"
            component-generator="HeaderSeparatorGenerator" />
        <show-association name="bpm:assignee"
            display-label-id="wf_reviewers"
            read-only="true"/>
    </property-sheet>
</config>
```

Deploying Alfresco Explorer customizations

The web client configurations can also be dynamically deployed without restarting the server.

This can be achieved by two ways:

- Upload or create the Process Definition file in the new Workflow Definitions space (**Company Home | Data Dictionary | Workflow Definitions**).
- Using the pre-registered URLs provided.

Dynamic Alfresco Explorer customizations are stored in the new **Web Client Extension** space (**Company Home | Data Dictionary | Web Client Extension**).

Upload a custom `web-client-config-custom.xml` file to the **Web Client Extension** space.

Company Home > Data Dictionary > Web Client Extension

Modify Content Properties

Modify the content properties then click OK.

Properties	
Name:	<input type="text" value="web-client-config-custom.xml"/>
Content Type:	<input type="text" value="XML"/> <input type="button" value="▼"/>
Encoding:	<input type="text" value="UTF-8"/> <input type="button" value="▼"/>
Title:	<input type="text" value="web-client-config-custom.xml"/> <input type="button" value=""/>
Description:	<input type="text" value="Display properties"/> <input type="button" value=""/> <input type="button" value=""/>
Author:	<input type="text" value="Amita Bhandari"/>
<input type="checkbox"/> Edit Inline: <input checked="" type="checkbox"/>	

The custom configuration will not be applied until it is explicitly reloaded or when the server is restarted.

Reloading web client customizations

If the `web-client-config-custom.xml` file has been added, edited, or updated, it can be dynamically reloaded by using the Alfresco Explorer configuration console via: `http://<server-name>:<port>/alfresco/faces/jsp/admin/webclientconfig-console.jsp`.

This has a single command `reload`, which will cause the Alfresco Explorer configuration to be reloaded.

Using the dynamic approach you can deploy workflow in DM. In order to deploy it in WCM you have to make the change manually in `web-client-config-wcm.xml`, as discussed earlier (see the *Deploying into WCM Project* section) in Chapter 5, *WCM Workflows* and then restart the server once. The dynamic approach can also be used if some changes are required after deployments so, no need to restart the server.

Testing the workflow

To assign workflow for a specific project, refer to the *Associating workflows to web project* section in *Chapter 5*. While submitting, the content task appears in user's dashlet. On clicking on the task, you find the screen shown in the following screenshot. The highlighted section, **Workflow Outline**, displays the diagram of the deployed Process Definition. This is visible since we have made hot deployment of the workflow. Now you can approve or reject the workflow:

Manage Task: dynamic:reviewTask
dynamic:reviewTask

Task Properties

General

Identifier: 242
Description: Dynamic Deployment
Comment: Approved

Reviewers

Reviewers: Administrator [admin]

Resources

Name	Description	Path	Created	Modified	Actions
Expand-snap.JPG		/ROOT	23 November 2009 09:55	14 December 2009 15:45	

Part of Workflow

Title: dynamic:processTask (dynamic:processTask)
Initiated by: Administrator
Started on: 14 December 2009
Completed on: <In Progress>

Workflow History

Description	Task Type	Id	Created	Assignee	Comment	Completed on	Outcome
Dynamic Deployment	dynamic:initializeTask	241	14 December 2009 15:47	admin		14 December 2009 15:47	Task Done

Workflow Outline

```
graph TD; start((<<Start State>>)) --> review[<<Task Node>> review]; review -- reject --> end((<<End State>> end)); review -- approve --> end;
```

Customization of existing workflow to use e-mail notifications

Alfresco provides an inbuilt Web Site Submission workflow to be associated with the web project. If you want to make change in the existing workflow, following are the location of the files to make the changes.

1. The Process Definition file is present in the specified location:

```
<install-alfresco>/tomcat/webapps/alfresco/WEB-INF/classes/
alfresco/workflow/submit_processdefinition.xml.
```

2. The Task Model file is also present at same location:

```
<install-alfresco>/tomcat/webapps/alfresco/WEB-INF/classes/
alfresco/workflow/wcmWorkflowModel.xml.
```

3. The Resource Bundle is also present at same location:

```
<install-alfresco>/tomcat/webapps/alfresco/WEB-INF/classes/
alfresco/workflow/wcm-workflow-messages.properties.
```

4. The Spring Context file, which is used to load model, Process Definition, and Resource Bundle is present at the location:

```
<install-alfresco>/tomcat/webapps/alfresco/WEB-INF/classes/
alfresco/bootstrap-context.xml.
```

5. The Alfresco Explorer Dialog is located at:

```
<install-alfresco>/tomcat/webapps/alfresco/WEB-INF/classes/
alfresco/web-client-config-properties.xml.
```

Consider a case where you want to add an e-mail notification at the submission of content. It is easy to customize the workflow by adding the following code snippet that is highlighted:

```
<task name="wcmwf:submittedTask" swimlane="initiator">
  <timer dueDate="5 seconds" transition="onsubmit">
    <action class=
      "org.alfresco.repo.workflow.jbpm.AlfrescoJavaScript">
      <script>
        logger.log("WCM Submit Process: Triggering submit for "
          + bpm_workflowDescription);
      </script>
    </action>
  </timer>

  <event type="task-end">
    <script>
      <variable name="submitfailed" access="write"/>
    </script>
  </event>
</task>
```

```
<expression>submitfailed = false;</expression>
</script>

<action class=
"org.alfresco.repo.workflow.jbpm.AlfrescoJavaScript">
<script>
    logger.log("WCM Submit Process: Start submit for " +
    bpm_workflowDescription + " (by " +
    person.properties.userName + ") ");
</script>
</action>
<action class=
"org.alfresco.repo.workflow.jbpm.AlfrescoJavaScript">
<script>
    var mail = actions.create("mail");
    mail.parameters.to = initiator.properties.email;
    mail.parameters.subject = "Submit Task " +
        bpm_workflowDescription;
    mail.parameters.from = person.properties.email;
    mail.parameters.text = "It's done";
    mail.execute(bpm_package);
</script>
</action>

<action class=
"org.alfresco.repo.avm.wf.AVMSubmitPackageHandler"/>
<action class="org.alfresco.repo.avm.wf.AVMDeployHandler"/>
<action class=
"org.alfresco.repo.workflow.jbpm.AlfrescoJavaScript">
<script>
    logger.log("WCM Submit Process: End submit for " +
    bpm_workflowDescription + " (by " +
    person.properties.userName + ") ");
</script>
</action>
</event>
</task>
```

Deploy the Process Definition using a dynamic approach and test the workflow.

If you want to deploy changes manually, you have to make changes in the `bootstrap-context.xml` file as mentioned previously in the fourth point.

```
<props>
    <!-- WCM workflow definition -->
    <prop key="engineId">jbpm</prop>
    <prop key="location">
        alfresco/workflow/submit_processdefinition.xml</prop>
    <prop key="mimetype">text/xml</prop>
    <prop key="redeploy">true</prop>
</props>
```

You should receive an e-mail notification at the submission of content to the Staging box.

Remove workflow for specific staging submission

We have already configured workflows for blogs and training web content. There can be various reasons why you don't want to follow a complete process of workflow and have to submit the content directly. If we think of the out-of-the-box feature, then we have to remove the workflow for those web forms so that web content can be submitted directly. This is not a good practice sometimes when you are configuring a workflow or sometimes when you are removing a workflow for a particular web content.

In order to make the task simpler we have to customize the submit dialog using APIs.

Follow these steps to create two Java files:

1. Create a package `com.book.web.bean.wcm`, in Eclipse, in your project. For more details refer *Chapter 2, Installation and Configuration*.
2. Create a Java file `ExtendedSubmitDialog` and copy the downloaded content from the Packt website.



Please download the complete code from the Packt website.



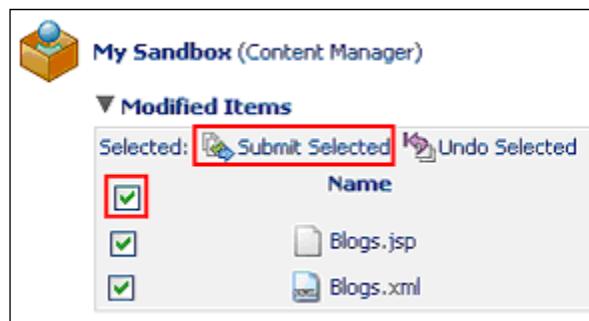
3. Create a Java file `ExtendedSubmitConfigureWorkflowDialog` and copy the downloaded content from the Packt website.

4. Open the file `faces-config-custom.xml`, located at `<install-alfresco>/tomcat/webapps/alfresco/WEB-INF` and copy the downloaded content from the Packt website.
5. Run Ant to compile the code and create a JAR file in `<install-alfresco>/tomcat/webapps/alfresco/WEB-INF/lib`. Please refer to *Chapter 2, Installation and Configuration* for compiling and deploying code to the installed Alfresco.
6. Start the Alfresco server.

Test using Submit Items wizard

Follow the steps mentioned ahead to submit all or selected modified content to the workflow.

1. Go to **Company Home | Web Projects | Cignex**.
2. **Submit Selected** option is provided under **Modified Items of My sandbox**.



3. On click of **Submit Selected**, the **Submit Items** wizard is opened and you will notice that one more workflow is configured.

Company Home > Web Projects > Cignex

Submit Items

This page helps you to submit modified items for publishing on the website.

Submission Info

Label: Directly Deploying content
 Description: Directly Deploying content
 Auto Deploy (This will deploy the changes from this submission upon approval.)

Workflow

Use the following workflow to submit all modified items

Web Site Submission(Submit changes for approval)
 Dynamic Workflow(Dynamic Workflow for approval) [Configure Workflow](#)
 No workflow now

Content Launch

Launch Date: None

Content Expiration

Set expiration date for all modified items.
 Note: To change individual expiration dates click the 'Change Expiration Date' action icon.

Modified Items

Name	Description	Path	Modified Date
Blogs.jsp	Rendered by blog into /ROOT/common /inc/Blogs.jsp	/ROOT/common/inc/Blogs.jsp	12 December 2009 18:43
Blogs.xml		/ROOT/common/inc/Blogs.xml	12 December 2009 18:43

4. Select **No workflow now** and click on **OK**.

5. The content will be submitted directly into the Staging server.

The screenshot shows the 'Staging Sandbox' interface. At the top, there's a header with a 'Staging Sandbox' icon, the text 'Created On: 22 November 2009', 'Created By: admin', and 'There are 14 users working on this web project.' Below this is a navigation bar with 'Recent Snapshots' (selected), 'When: All Today Last 7 days Last 30 days', and buttons for 'Browse Website', 'Preview Website', and 'Refresh'. A table titled 'Recent Snapshots' lists one entry: 'Name: Directly Deploying content', 'Description: Directly Deploying content', 'Date: 14 December 2009 10:39', 'Submitted By: admin', 'Version: 51', 'Status: Pending', and 'Actions' (with icons for edit, preview, and delete). Below the table is a link 'Content Awaiting Launch'.

ZERO Workflow

You are required to select a user or group whenever your workflows are processed to deploy content on the staging server. However, if you customize your workflow, the user group selection becomes optional.

Let's take the same example that we have used for the *Dynamic deployment* section.

- **Step 1:** Add one more task in your model Dynamic Model
dynamicWCMWorkflowModel:

```
<aspect name="dynamic:assignee">
  <associations>
    <association name="bpm:assignee">
      <source>
        <mandatory>false</mandatory>
        <many>false</many>
      </source>
      <target>
        <class>cm:person</class>
        <mandatory>false</mandatory>
        <many>false</many>
      </target>
    </association>
  </associations>
</aspect>
```

Make the highlighted changes. Earlier we have used `bpm:assignee`; now we are using `dynamic:assignee`.

```
<type name="dynamic:initializeTask">
  <parent>dynamic:startTask</parent>
  <properties>
    <property name="dynamic:property">
```

```

<title>Dynamic Property</title>
<type>d:text</type>
</property>
</properties>
<mandatory-aspects>
    <aspect>dynamic:assignee</aspect>
</mandatory-aspects>
</type>
<type name="dynamic:reviewTask">
    <parent>dynamic:workflowTask</parent>
    <overrides>
        <property name="bpm:packageItemActionGroup">
            <default>edit_wcm_package_item_actions</default>
        </property>
    </overrides>
    <mandatory-aspects>
        <aspect>dynamic:assignee</aspect>
    </mandatory-aspects>
</type>

```

Reload the model using a dynamic approach as discussed earlier.

- **Step 2:** Make the highlighted changes in `web-client-config-custom.xml`.

```

<config evaluator="node-type" condition="dynamic:initializeTask"
    replace="true">
    <property-sheet>
        <separator name="sep2" display-label-id="users_and_roles"
            component-generator="HeaderSeparatorGenerator" />
        <show-property name="dynamic:property" />
        <show-association name="dynamic:assignee"
            display-label-id="wf_reviewers" />
    </property-sheet>
</config>
<config evaluator="node-type" condition="dynamic:reviewTask"
    replace="true">
    <property-sheet>
        <separator name="sep1" display-label-id="general"
            component-generator="HeaderSeparatorGenerator"/>
        <show-property name="bpm:taskId" />
        <show-property name="bpm:description"
            component-generator="TextAreaGenerator"
            read-only="true"/>
        <show-property name="bpm:comment"
            component-generator="TextAreaGenerator" />
    <separator name="sep2" display-label-id="wf_reviewers">

```

```
        component-generator="HeaderSeparatorGenerator" />
    <show-association name="dynamic:assignee"
        display-label-id="wf_reviewers"
        read-only="true"/>
</property-sheet>
</config>
```

Reload Alfresco explorer using dynamic approach as discussed earlier.

- **Step 3:** Make the highlighted changes in `dynamicProcessdefinition.xml`.

```
<?xml version="1.0" encoding="UTF-8"?>
<process-definition xmlns="urn:jbpm.org:jpd1-3.1"
    name="dynamic:processTask">
    <swimlane name="initiator"/>
    <start-state name="start">
        <task name="dynamic:initializeTask" swimlane="initiator"/>
        <transition name="" to="submitreview">
        </transition>
    </start-state>
    <decision name="submitreview">
        <transition name="" to="end" >
            <condition>#{bpm_assignee ==null}</condition>
        </transition>
        <transition name="" to="review">
            <condition>#{bpm_assignee!=null}</condition>
        </transition>
    </decision>
    <task-node name="review">
        <task name="dynamic:reviewTask">
            <assignment class=
                "org.alfresco.repo.workflow.jbpm.AlfrescoAssignment">
                <actor>#{bpm_assignee}</actor>
            </assignment>
            <event type="task-end">
                <action class=
                    "org.alfresco.repo.avm.wf.AVMSubmitPackageHandler"/>
                <action class=
                    "org.alfresco.repo.avm.wf.AVMDeployHandler"/>
                <action class=
                    "org.alfresco.repo.workflow.jbpm.AlfrescoJavaScript">
                    <script>
                        logger.log("WCM Submit Process: End submit for " +
                            bpm_workflowDescription + "
                            (by " + person.properties.userName + ")");
                    </script>
            </event>
        </task>
    </task-node>
</process-definition>
```

```

        </action>
    </event>
</task>
<transition name="approve" to="end">
</transition>
<transition name="reject" to="end" />
</task-node>
<end-state name="end"/>

<event type="process-end">
    <action class=
        "org.alfresco.repo.avm.wf.AVMRemoveAllSrcWebappsHandler"/>
    <action class=
        "org.alfresco.repo.avm.wf.AVMSubmitPackageHandler"/>
    <action class="org.alfresco.repo.avm.wf.AVMDeployHandler"/>
    <action class=
        "org.alfresco.repo.avm.wf.AVMReleaseTestServerHandler"/>
    <action class=
        "org.alfresco.repo.avm.wf.AVMRemoveWFStoreHandler"/>
</event>
</process-definition>

```

Update Process Definition using a dynamic approach as discussed earlier.

- **Step 4:** Test using the **Submit Items** Wizard.

Click on **Configure workflow** to remove the user you have configured earlier.

Submit Items

This page helps you to submit modified items for publishing on the website.

Submission Info

* Label: Test Zero Workflow

* Description: Test Zero Workflow

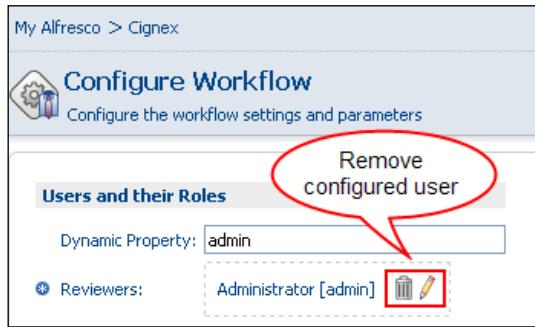
Auto Deploy (This will deploy the changes from this submission upon approval.)

Workflow

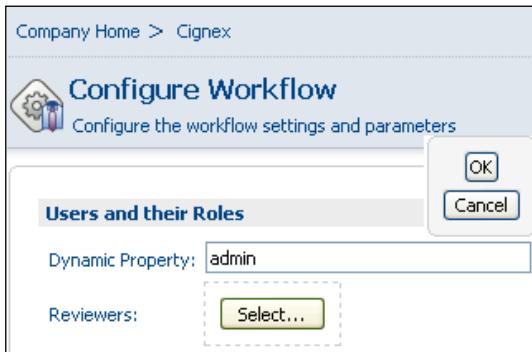
Use the following workflow to submit all modified items

Dynamic Workflow(Dynamic Workflow for approval) Configure Workflow

- **Step 5:** Remove the user as shown in the following screenshot:



- **Step 6:** After removing the user click on **OK**.



Now that the workflow settings have been changed, click on **OK** to submit content to the staging server. It will not go to any user for processing, as we have not selected any user. Content is submitted directly to staging.

Name	Description	Date	Submitted By	Version	Status	Actions
Test Zero Workflow	Test Zero Workflow	18 December 2009 14:52	mark	65	Published	

Workflow Viewer

Suppose you want to see list of pending workflows. We can create a dashlet that will list down all the pending workflows for Groups (the same way you can also create for users). The list is quite useful for administrators.

Follow the steps to create dashlets:

1. Create a Java file `CustomWorkflowStatusBean` in the package `com.book.web.bean.wcm` and paste the downloaded code from the Packt website.
2. Add the following code in the `faces-config-custom.xml` file:

```
<managed-bean>
    <managed-bean-name>CustomWorkflowStatusBean</managed-bean-name>
    <managed-bean-class>
        com.book.web.bean.wcm.CustomWorkflowStatusBean
    </managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>
    <managed-property>
        <property-name>navigationBean</property-name>
        <value>#{NavigationBean}</value>
    </managed-property>
    <managed-property>
        <property-name>serviceRegistry</property-name>
        <value>#{ServiceRegistry}</value>
    </managed-property>
</managed-bean>
```

3. Create a JSP file `workflow-status.jsp` in the specified location `<install-alfresco>/tomcat/webapps/alfresco/JSP/extension/dashlets/` folder. Paste the downloaded code from the Packt website.
4. Add the following code in the `web-client-config-custom.xml` file:

```
<config evaluator="string-compare" condition="Dashboards">
    <dashboards>
        <dashlets>
            <dashlet id="wfstatus-bean"
                label="In-flight Workflow Status"
                description="In-flight Workflow Status of invited project"
                jsp="/jsp/extension/dashlets/workflow-status.jsp" />
        </dashlets>
    </dashboards>
</config>
```

5. Run Ant to compile the code and create a JAR file in `<install-alfresco>/tomcat/webapps/alfresco/WEB-INF/lib`.

6. Start the Alfresco server.
7. Log in as admin.
8. Configure **In-flight Workflow Status** dashlets. Refer to the *Web Publishing dashlets* section in *Chapter 4, Web Content Production with Web Forms*, to configure.
9. You will see the following screen that lists all the pending workflows:

The screenshot shows a web-based dashboard titled "In-flight Workflow Status". A red box highlights the title, and a red arrow points from the text "List of Pending workflows" to the table below. The table has columns for "Submission Label", "Type", "Id", "Start Date", "Launch Date", "Initiated By", and "Pooled Actors". It displays two rows of data: one for a "Training" submission labeled "Review" with ID 249, and another for a "Launch Date" submission labeled "Submission Aborted" with ID 88. The "Start Date" for both is 16 December 2009 17:07, and the "Launch Date" is 7 December 2009 13:25. The "Initiated By" column shows "mark@ignex.com" for the first row and "admin@alfresco.com" for the second. The "Pooled Actors" column is empty. At the bottom, there is a page navigation bar with "Page 1 of 1" and several icons.

Submission Label	Type	Id	Start Date	Launch Date	Initiated By	Pooled Actors
Training	Review	249	16 December 2009 17:07		mark@ignex.com	
Launch Date	Submission Aborted	88	7 December 2009 13:25	7 December 2009	admin@alfresco.com	

Summary

Workflows can be deployed dynamically also. In this chapter we have learned the following points:

- The customization of workflow can be achieved using two approaches, manual and dynamic.
- E-mail notifications can be made to all the concerned people involved in the workflow process.
- Workflows for a specific Staging Sandbox submission can also be removed using some customization.
- Implementing ZERO workflow gives the flexibility to submit content using workflows without selecting any user/group.
- Workflow Viewer can be implemented to see the number of active workflows, which is good for administrators.

7

Content Delivery and Deployment

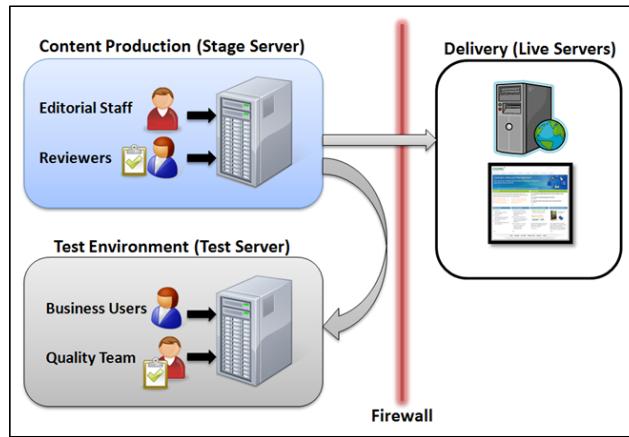
So far you have learned the content production capabilities of Alfresco using web projects, User Sandboxes, web forms, and workflow. This chapter introduces you to the content delivery and deployment features, of Alfresco. You will understand the concepts behind delivering static content as well as dynamic content to the external production servers. This chapter covers deployment to both live servers as well as to the test servers. This chapter also focuses on the auto deployment feature where the content can be scheduled to be delivered to the production servers automatically.

By the end of this chapter you will have learned how to:

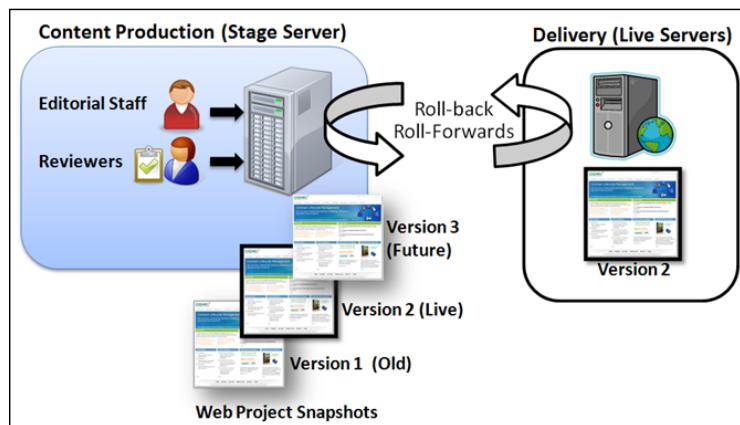
- Install and configure File System Receiver (FSR)
- Use Alfresco Server Receiver (ASR)
- Set up the process for auto deployment
- Deploy to a test server
- Deploy directly from a workflow
- Set up hybrid deployment for both static and dynamic content

Introduction to content delivery

Alfresco provides a framework for pushing content from a stage (or authoring) server to live and test servers, as shown in the following figure:



The Alfresco content production environment produces an approved view of a web project called a **snapshot**. Consider each snapshot as a web project version. Alfresco deployment takes a snapshot and pushes it out to either live or test servers. Consider a sample scenario as shown in the following diagram, where the content from the stage server is deployed to live servers. When snapshot version 2 is deployed to live servers, then the Alfresco deployment engine only copies the files which are either new or modified and removes the files which are deleted when compared to snapshot version 1. The deployment engine is smart, which affects only few files rather than copying all of the files of a web project. Now that the snapshot version 2 is live (deployed to live servers), the editorial staff may work on a future version 3.



Let's say for some reason there is an issue with snapshot version 2, which is live. You have the option of rolling it back to the previous good version of snapshot version 1. You can roll forward or you can roll back to a specific version of a web project snapshot version. This feature is very powerful even from a legal audit point of view, wherein you have an ability to reproduce the website as of a specific date.

Further, the deployment process may be automated so that it happens automatically when content is approved for publishing.

The deployment framework provides a flexible and highly-configurable system to allow you to tailor the system to your requirements. If the Alfresco-supplied components are not suitable, you can plug in your own authenticators, transport implementations, content transformers, and deployment targets.

Live server vs. Test server

Alfresco WCM enables previewing the content within the stage server environment. After content creation, the Editorial staff may preview web pages to verify the content, as well as the look and feel. Similarly the content reviewers and business owners may preview to review the web pages during the workflow process.

Because of this powerful feature, you may not need a separate test server to preview and approve the content. The stage server itself is used for both authoring and testing the content. Hence, the content is authored and approved on the stage server, and then deployed to the live servers directly.

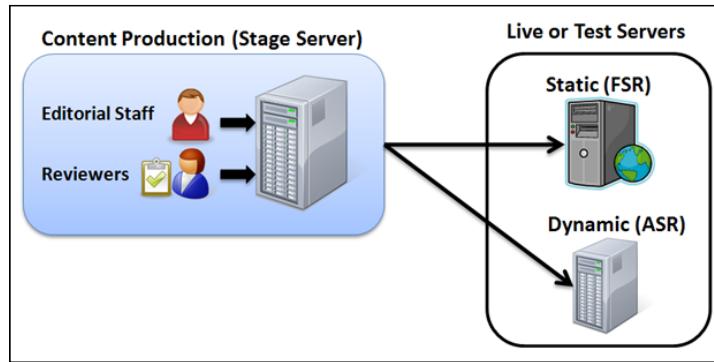
However, there can be a situation where you may need a separate test server. For example, if you are deploying content to another frontend application outside of Alfresco such as a PHP or .NET application, or situations when the virtualization server is not capable of providing the preview. Starting with the version 2.2 release, Alfresco introduced the concept of a Test Server.

You deploy the content from a Staging Sandbox to the live server and you deploy the content from User Sandbox or from a workflow to the test server.

Static vs. Dynamic delivery model

Within the live or test server environment, you can push out content to a flat filesystem to be served up by Apache or IIS, or you can push your content into another runtime instance of Alfresco.

Pushing content to a flat filesystem environment is also known as **Static Deployment** and it is achieved using **Alfresco File System Receiver (FSR)**. Pushing content to another runtime instance of Alfresco is also known as **Dynamic Deployment** and it is achieved using **Alfresco Server Receiver (ASR)**.



In static deployment, the web pages are already rendered (or baked) before deploying. In dynamic deployment, since the content is in the runtime instance of Alfresco, the web pages will be generated (or fried) at runtime. The following is a summary of static and dynamic delivery models:

	Static "Bake" Model	Dynamic "Fry" Model
Delivery Technology	Web Servers	Application Servers
Page Compositing	Submission time	Request time
Content deployed to	Filesystem	Alfresco Runtime
Content Search	Not supported	Supported out of the box
Content Security	Not supported	Supported out of the box
Personalization	Limited	Unlimited
Performance	Ultimate	Less than the "bake" model

You can consider a hybrid deployment (both static and dynamic) for some business applications. You can define certain static content of the web project such as images, videos, and scripts to be deployed to the filesystem and certain dynamic content such as web pages to be deployed to the Alfresco runtime. This approach gives you good performance as well as personalized and dynamically changing content in a production environment.

FSR for static delivery

A File System Receiver (FSR) will need to be installed and configured on each live or test server to receive published static content from the Alfresco Staging Server.

The FSR is a small, standalone server that receives updates from an Alfresco repository running Web Content Management; content is published to a flat filesystem. The published flat files will typically be served by a web server such as Apache, for static content or an application server such as Tomcat, JBoss, or IIS for web applications (WARs, PHP files, and so on).

FSR requires filesystem access and must run as a user with appropriate rights to the target filesystem. The FSR is a standalone Java Daemon (no Tomcat or other app server required) and it has minimal resource requirements. The FSR supports the invocation of custom Java code and/or programs. Therefore, it can be used to perform additional tasks post-deployment such as search engine indexing, pushing content to a **Content Delivery Network (CDN)**, or replicating content to other systems or repositories.

The destination file server receiver has to be running with its RMI registry port and service port (44100 and 44101 by default) opened.

Installing FSR

If you refer to SourceForge at <http://sourceforge.net/projects/alfresco/files/>, you will notice three different downloads of FSR. A Microsoft Windows installer file (`Alfresco-DeploymentCommunity-3.3-Setup.exe`), a Linux installer file (`Alfresco-DeploymentCommunity-3.3-Linux-x86-Install`) for automatic installation, and a ZIP file (`alfresco-community-deployment-3.3.zip`) for manual installation. I would prefer using the ZIP file and manually installing the standalone deployment receiver. Both Windows and Linux installers have certain limitations as they do not provide configuring various deployment targets.

Unzip the deployment ZIP file into a convenient location (it does not make its own directory) on a live or test server. Notice a file named `deployment.properties`, which contains the configuration information. The folder `deployment` includes default target information.

To configure the filesystem receiver, open the `deployment.properties` file in the text editor of your choice. Choose locations for each of the following:

```
; filesystem receiver configuration
deployment.filesystem.datadir=D:/07_MUN_WORK/alfresco_book_wcm_32e/
deployment-data/depdata
```

```
deployment.filesystem.logdir=D:/07_MUN_WORK/alfresco_book_wcm_32e/
deployment-data/deplog
deployment.filesystem.metadatadir=D:/07_MUN_WORK/alfresco_book_
wcm_32e/deployment-data/depmetadata
deployment.filesystem.autofix=true
deployment.filesystem.errorOnOverwrite=false

; Deployment Engine configuration
deployment.rmi.port=44100
deployment.rmi.service.port=44101

; Stand alone deployment server specific properties
deployment.user=admin
deployment.password=admin
```

- **deployment.filesystem.datadir:** This is the location in which the filesystem deployment receiver stores deployed files during a deployment, before committing them to their final locations.
- **deployment.filesystem.logdir:** This is the location in which the filesystem deployment receiver stores deployment time log data.
- **deployment.filesystem.metadatadir:** This is the location in which the filesystem deployment receiver stores metadata about deployed content.
- **deployment.filesystem.autofix:** The file system deployment target can either issue an error upon detecting a problem or automatically fix the problem. The `autofix` parameter controls whether the File System Deployment Target will attempt to fix the metadata itself or just issue a warning. Set the value to true to fix, or false to not fix.
- **deployment.filesystem.errorOnOverWrite:** The file system deployment target can issue an error upon overwriting the files. Set the value to false to overwrite the files, which is needed when updating the existing files.
- **deployment.rmi.port:** The port number to use for the RMI registry. Choose this so as not to conflict with any other services. By default, the standalone deployment receiver uses 44100.
- **deployment.rmi.service.port:** The port number to use for RMI service. Choose this so as not to conflict with any other services. By default this is 44101.

Note that while specifying the directory locations on Microsoft Windows, either use forward slashes or escape the backslashes. For example, C:/dir1/dir2 or C:\\dir1\\\\dir2.

Configuring your deployment targets

You can configure as many target filesystem receivers as you need on a single live or test server. By default, a single filesystem receiver is defined with simple configuration via `deployment.properties`.

Deployment targets are placed in the `deployment` folder with the filename `deployment/*target.xml`. To define more targets, follow the pattern of `deployment/default-target.xml`. There are two steps involved:

1. Definition of your target information in the `deployment.properties` file
2. Registration of your target with the deployment engine using an XML file

Let's create a deployment target for the CIGNEX website and let's name it as `cignex-live1` target. As a first step to configure filesystem receiver, open the `deployment.properties` file in the text editor of your choice and add the `cignex-live1` filesystem target configuration as follows:

```
; cignex-live1 filesystem target configuration
deployment.filesystem.cignex-live1.metadatadir= ${deployment.
filesystem.metadatadir}/cignex-live1
deployment.filesystem.cignex-live1.rootdir=
D:/07_MUN_WORK/alfresco_book_wcm_32e/deployment-data/targets/cignex-
live1
deployment.filesystem.cignex-live1.name=cignex-live1
deployment.filesystem.cignex-live1.user=admin
deployment.filesystem.cignex-live1.password=admin
```

Now to register this new target, you need to create a target XML file in the `deployment` folder. You can refer to an existing target file, `default-target.xml`, in the `deployment` folder for more information.

Copy `deployment/default-target.xml` as the `deployment/cignex-live1-target.xml` file. Open the `deployment/cignex-live1-target.xml` file in your text editor of choice and replace the keyword `default` with the keyword `cignex-live1`.

With these simple two steps, you have configured a new target named `cignex-live1`.

Start and stop deployment receiver

To run the receiver, execute `deploy_start.sh` (or `deploy_start.bat`) as the user on that server. Remember this user will be the owner of the deployed content.

To stop the receiver, execute the `deploy_stop.sh` or `deploy_stop.bat` file.

Using FSR from Alfresco WCM staging

Now that the FSR is configured and running, you can use it from Alfresco staging to deploy the content.

Configuring a web project to use FSR

The following are the steps to configure a Web Project to use an FSR.

1. Navigate to Company Home | Web Projects | <web project name>.
2. Select **Edit Web Project Settings** from the **Action** menu.
3. Click on **Next** to reach the **Configure Deployment Servers** window.
4. Click on the **Add Deployment Receiver** link as shown in the following screenshot. Fill out the form as needed. The minimum required fields to be filled out, assuming default settings, are the **Host** name where the FSR is located and the **Target Name**.

Step Two - Configure Deployment Servers
Configure deployment servers for the web project.

[Add ASR \(Deprecated\)](#) [Add Deployment Receiver](#)

Edit details of the Deployment Receiver

Type:

Display Name:

Display Group:

Transport Name:

Host: *

Port:

URL:

Username:

Password:

Source Path:

Excludes:

Target Name:

Include In Auto Deploy

The following table contains the description of each of the FSR configuration fields.

Field Name	Description
Type	Live Server or Test Server. You deploy the content from Staging Sandbox to the live server. And you deploy the content from User Sandbox or from workflow to the test server.
Display Name	A descriptive label for the server, used by the UI.
Display Group	The deployment receivers configured using the same Display Group name will be treated as one batch during deployment.
Transport Name	Name of the network protocol connection to the remote filesystem receiver. By default it is RMI.
Host	The host name of the destination server, can be a name or IP address.
Port	The RMI port to connect to on the destination server.
URL	The runtime URL of the destination server. Can be used to preview the deployment, upon a successful deployment.
User Name	The username to use to connect to the destination server.
Password	The password to use to connect to the destination server.
Source Path	The path of the folder to deploy, for example /ROOT/site1.
Excludes	A single regular expression (multiple rules can be defined within the expression) of items to exclude from the deployment, for example .*\.\.jpg\\$.*\.gif\\$.
Target Name	The name of a target to deploy to, configured in the FSR.
Include in Auto Deployment	If checked, then this target will be included in auto deployment.

- Click on the **Add** and **Finish** buttons to complete the configuration.

Deploying a snapshot to FSR manually

The following represents the steps required to deploy content to an FSR. Similar steps are used to deploy content to an ASR.

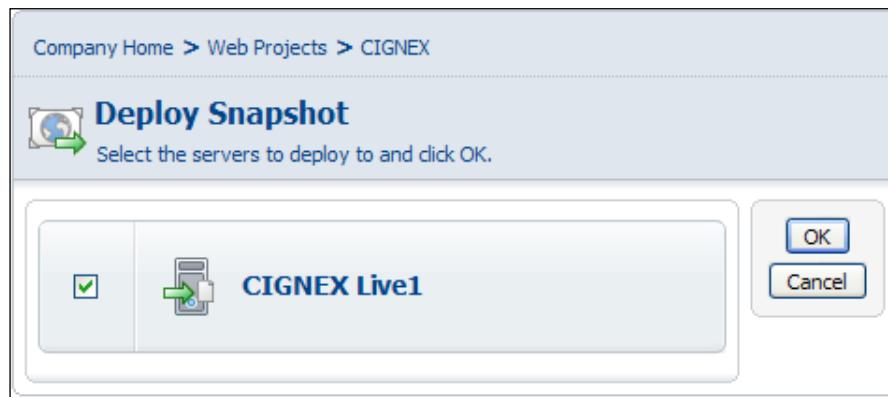
- Navigate to **Company Home | Web Projects | <web project name>**.

Content Delivery and Deployment

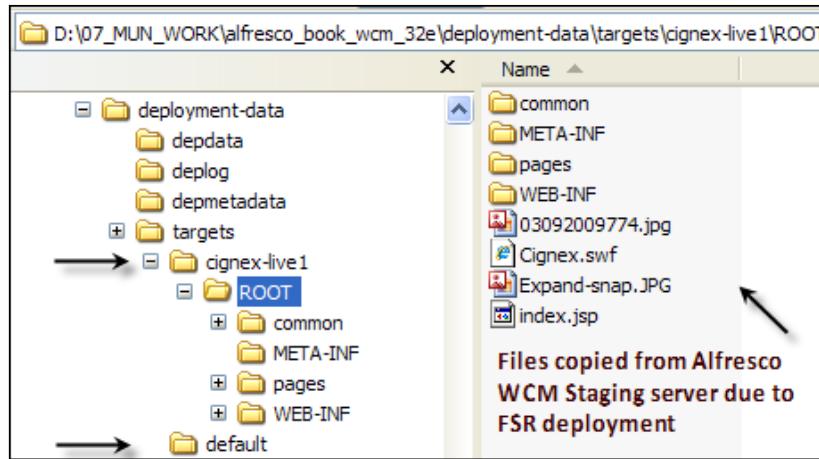
2. Expand **Recent Snapshots**; it may be required to select the appropriate **When** value to find a snapshot to deploy.

The screenshot shows the 'Staging Sandbox' interface. At the top, there are three buttons: 'Browse Website', 'Preview Website', and 'Refresh'. Below these, project details are listed: 'Created On: 21 March 2010', 'Created By: admin', and 'There is one user working on this web project.' A section titled 'Recent Snapshots' is expanded, showing a table with one row. The table columns are Name, Description, Date, Submitted By, Version Status, and Actions. The single entry is 'first version of website' with the description 'created the first version of the website'. The date is '21 March 2010 17:38' and the submitted by field is 'admin'. The version status is '3'. In the 'Actions' column, there are four icons: a purple arrow, a green circle with a plus sign, a blue square with a minus sign, and a yellow 'Deploy' button. The 'Deploy' button is highlighted with a mouse cursor.

3. Deploy content by clicking on the **Deploy** icon to the right-hand side of the desired snapshot.
4. Select the server(s) to deploy the snapshot to, on the **Deploy Snapshot** window. Click on **OK**. This screen will auto refresh to show the success or failure of a deployment.



When the deployment is successful, you will notice that the files from the Alfresco WCM staging environment are copied to the filesystem of the live server where the target location is specified. The following screenshot shows that the files are copied to the folder configured for the target **cignex-live1**:



Viewing deployment report and history

Alfresco WCM staging environment captures all of the deployment reports and maintains the complete deployment history for audit trail purpose.

In order to view the deployment report and the history, navigate to **Company Home | Web Projects | <web project name>** and click on the **View Deployments** link on the right-hand side top corner of the **Staging Sandbox** window. You will notice the report similar to the one shown in the following screenshot.

Content Delivery and Deployment

For a specific deployment, you can list the files deployed by clicking on the **Details** link. You can also see reports for all of the deployments done earlier. Click on any **Attempt Date** listed in the **More Deployment Reports** window to view the detailed deployment report:

Company Home > Web Projects > CIGNEX

Last Deployment Report

View deployment details for each of the servers selected in the last deployment.

 **CIGNEX Live1**
Deployment Successful
Server: \\localhost
Snapshot: 4
Started: 21 March 2010 18:03
Finished: 21 March 2010 18:03
By: admin
Target Name: cignex-live1
► Details

▼ More Deployment Reports

Today Yesterday **Last 7 days** Last 30 days All

Attempt Date	Deployed To	Snapshot
21 March 2010 18:03	CIGNEX Live1	4
21 March 2010 17:48	CIGNEX Live1	3

Reverting or rolling back to an older snapshot

Snapshots can be reverted, compared to the previous snapshot, and compared to any snapshot by selecting the appropriate actions icon as shown in the following screenshot:

The screenshot shows the 'Staging Sandbox' interface. At the top, there are links for 'Browse Website', 'Preview Website', 'Refresh', and 'View Deployments'. Below this, it says 'Created On: 20 February 2010' and 'Created By: admin'. It also indicates 'There are 3 users working on this web project.'

A dropdown menu 'Recent Snapshots' is open, showing three entries:

Name	Description	Date	Submitted By	Version	Status	Actions
deleted file	deleted file	21 March 2010 17:29	admin	5		
news1	news1 blog1 training1 submitted	20 February 2010 21:54	admin	4	LIVE	
first update	brand new	20 February 2010 21:14	admin	3		

Below the table, there is a section titled 'Content Awaiting Launch'.

Annotations in red text and arrows point to specific icons:

- 'Deploy' points to the 'Deploy' icon in the Actions column of the first two rows.
- 'Compare to Previous Snapshot (4)' points to the 'Compare to Previous Snapshot' icon in the Actions column of the first row.
- 'Compare To Current Snapshot (5)' points to the 'Compare to Any Snapshot' icon in the Actions column of the second row.
- 'Compare to Any Snapshot' points to the 'Compare to Any Snapshot' icon in the Actions column of the third row.
- 'Revert or Roll-back' points to the 'Revert' icon in the Actions column of the second row.

Let's say for some reason there is an issue with snapshot version 4, which is live. You can revert to a previous and stable snapshot version 3. In order to revert to a specific snapshot, click on the **Revert** icon and select the targets to revert to.

For the deployment **Status**, there could be four possible values:

- **IN PROGRESS**: Deployment is occurring on one or more servers
- **LIVE**: All servers were deployed to successfully
- **PARTIAL FAILURE**: Deployment failed on one or more servers
- **FAILED**: Deployment failed on all servers

Deploying to multiple servers

On a specific receiving server (FSR), you can configure more than one target. For example, you can deploy images to a doc-root of the Apache web server (say target1) and you can also deploy videos to a streaming server (say target2). Similarly, you can install and configure FSR on multiple servers.



To configure multiple deployment receivers, follow the instructions given in this chapter in the *Configuring a web project to use FSR* section.

In the **Add Deployment Receiver** form, use the **Source Path** field to specify a folder for the web project to be deployed. Also use the **Excludes** field to exclude certain types of files to be deployed.

Advanced topics on FSR

This section covers a few advanced features that are useful to extend the File System Receiver.

Configuring prepare and postCommit callbacks

On the target receiving servers, the FSR can be extended by implementing a `prepare()` or `postCommit()` callback. The `prepare()` callback is called after all files have been successfully deployed to the FSR's temporary storage areas before actually copying them to the target location. The `postCommit()` callback is called after the files are successfully copied to the target location.

These two callbacks allow the developers to implement processing on deployed content through the use of a system command, script, or Java class.

Some examples are:

- Re-indexing of external search engine for newly added content in the filesystem
- Integrating with a third-party system, such as updating a database upon receipt of certain files
- Copying files to an external filesystem via FTP
- Sending e-mail notification upon failure or success of deployment

Refer to the `deployment/file-system-target-sample.xml` file, which includes `prepare` and `postCommit` blocks listed as follows:

```
<!-- Add your prepare callbacks here -->
<property name="prepare">
    <list>
        <bean class="org.alfresco.deployment.SampleRunnable" />
    </list>
</property>

<!-- Add your postCommit callbacks here -->
<property name="postCommit">
    <list>
        <bean class="org.alfresco.deployment.SampleRunnable" />
    </list>
</property>
```

Defining payload transformations

The data that streams out of Alfresco and into the FSR can be transformed by content transformation. On the Alfresco server, content transformers are defined in the configuration file `deployment-service-context.xml`. On the File System Receiver, transformers are defined in the configuration file `application-context.xml`.

Following are a few use cases listed for your reference.

- File compression for slow networks
- Encryption of sensitive data
- Transformation of content as it is deployed

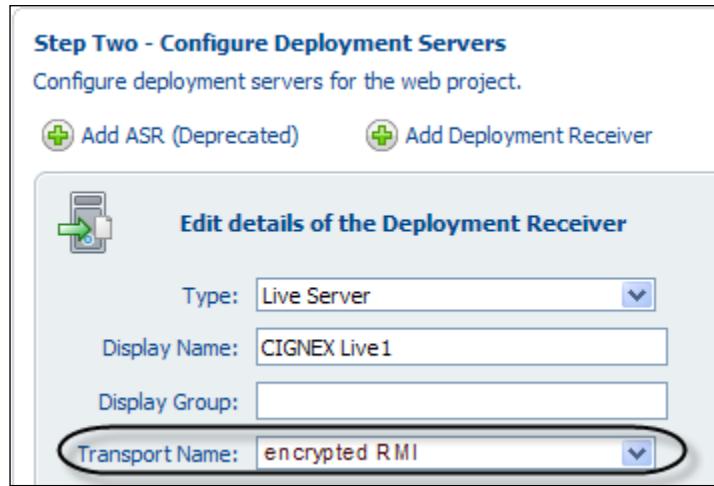
Here is an extract from `deployment-service-context.xml` showing the configuration of the encryption transformer, which takes two parameters:

```
<!-- Payload transformers -->
<bean id="deploymentEncryptor" class="org.alfresco.deployment.
transformers.SampleEncryptionTransformer">
    <property name="password">
        <value>Alfresco</value>
    </property>
    <property name="cipherName">
        <value>PBEWithMD5AndDES</value>
    </property>
</bean>
```

Defining transport adapters

The Alfresco deployment service supports the configuration of multiple transport adapters to enable connection to remote filesystem receivers using different network protocols (that is, encrypted RMI).

Transport adapters are configured on the Alfresco Staging Server in the Spring configuration file `deployment-service-context.xml`. An instance of the Alfresco server may support many different transports. However, each deployment receiver only exposes a single transport as shown in the following screenshot:



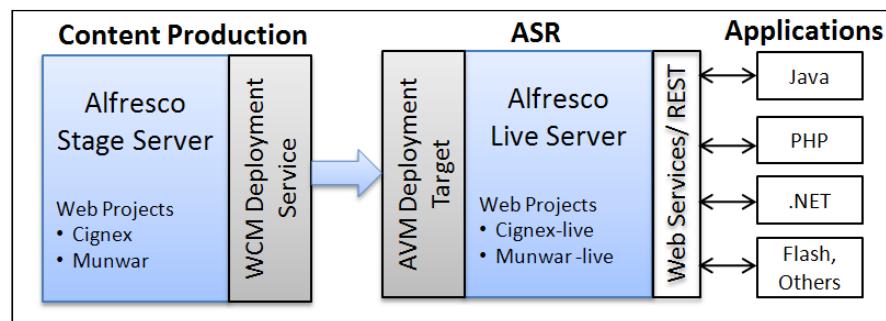
ASR for dynamic delivery

The Alfresco System Receiver (ASR) is just another instance of the Alfresco Server. It is also known as headless Alfresco, as the Web Client UI is not used. Also some features such as CIFS, WebDAV, FTP, and NFS are disabled.

The ASR allows a web project being authored in one Alfresco server instance to be deployed to another separate instance of Alfresco. ASR provides the ability to leverage search, versioning, and dynamic queries (via web scripts) within a live server environment—thereby making the content available for dynamic queries by basically any web technology (PHP, Python, J2EE, ASP .NET, AJAX, Flash, Cold Fusion, and so on). This provides ultimate flexibility in what and how the content is displayed on a page.

 From Alfresco 3.2 version onwards ASR is replaced by the client-side WCM Deployment Service and receiver-side AVM Deployment Target.

The WCM deployment service is the staging (sending) side of WCM deployment. The AVM Deployment Target is a target which is registered with the receiving side of deployment. By default its target name is "avm", although of course this can be changed through configuration. The AVM Deployment Target receives a deployment from an Alfresco WCM authoring environment and puts the content into an AVM store where it can be used to support a dynamic website.



Configuring WCM deployment service

You can configure the WCM deployment service by editing the settings in the `<tomcatHome>/shared/classes/extension/deployment-service-context.xml` file.

Number of send threads

To assist with cases where files are being deployed over a network with high latency, the deployment client is multi-threaded and sends several files at once. By default, five sends are done in parallel.

From 3.2 Enterprise onwards you can set the `deployment.service.numberOfSendingThreads` property in the `alfresco-global.properties` file.

Number of deployments in parallel

Deployment is controlled through the Action Service which controls how many deployments happen in parallel. If you need to deploy to many servers, then you may need to increase the number of deployments in parallel. But if you run out of processing power or memory, then you may need to reduce this setting.

From 3.2 Enterprise onwards you can set the following properties in the `alfresco-global.properties` file.

```
deployment.service.corePoolSize=2  
deployment.service.maximumPoolSize=3
```

AVM Deployment Target

The AVM Deployment Target is a target that is registered with the receiving side of deployment. By default its target name is "avm".

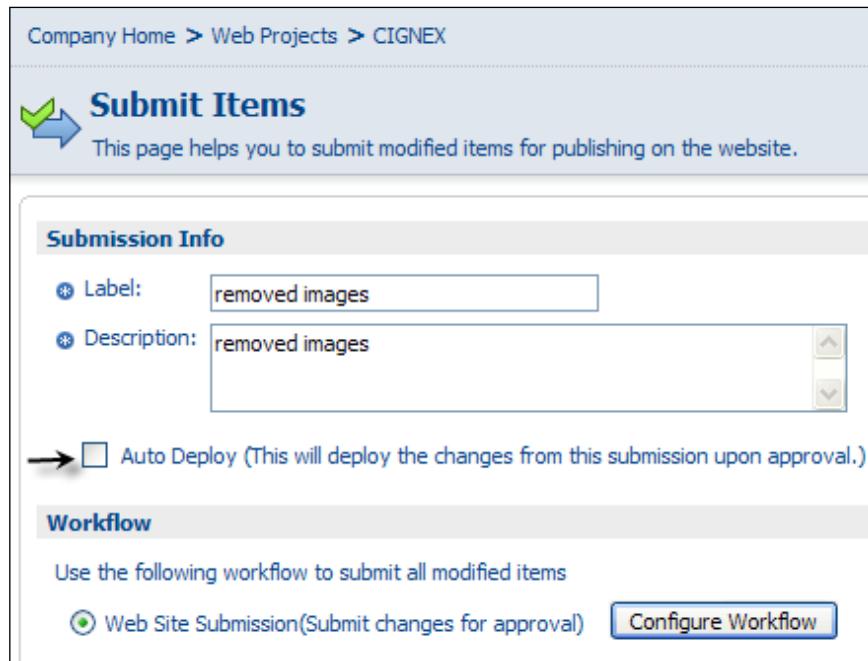
The ASR names the live and stores the same as the staging store. This is fine in most cases except for where the ASR is used to deploy to the same machine that has the authoring instance. In this case, the ASR attempts to avoid a circular dependency (or overwriting the source) by appending the destination store name with '-live'.

In order to have consistency between deploying to a local instance and deploying to a remote instance, the AVM deployment target always names live stores with the '-live' prefix. For example, if your web project name is "cignex" (on Staging Server), then on the ASR destination Alfresco server, the name of the web project would be "cignex-live".

Auto deployment

Alfresco WCM staging has an autodeploy option in its default workflow, allowing end users, at the time of submit, to enforce automatic deployment of approved changes directly to the live website without having to manually initiate deployment.

The **Submit Items** window has an **Auto Deploy** checkbox, as shown in the following screenshot:



Upon approval, if the auto deploy option is on, the workflow will perform a deployment to those live servers that have the **Include In Auto Deploy** option enabled. For more details about enabling this option, refer the *Configuring a web project to use FSR* section in this chapter.

Deploying to a test server

The Test Server Deployment functionality provides in-context preview by allowing a contributor to deploy their content to an external target (either an ASR or FSR), from which it can be rendered by any web application technology that can either read from a filesystem or access an ASR via HTTP (which includes all of the major web application technologies in use today, including Java, .NET, PHP, Ruby, Python, CGI, and so on).

Once a test server has been deployed to, it is allocated to the user or workflow that performed the deployment. Once the user or workflow has finished with the test server it is released and returned to the pool of test servers. This happens automatically in the case of a workflow sandbox and manually via a UI action for User Sandboxes.

The following process has to be followed to use the test server:

1. Set up a test server pool.
2. Deploy to a test server.
3. Preview the content.
4. Release the test server.

Setting up a test server pool

The following are the steps to configure a Web Project to use an FSR.

1. Navigate to Company Home | Web Projects | <web project name>.
2. Select the **Edit Web Project Settings** from the **Action** menu.
3. Click on **Next** to reach the **Configure Deployment Servers** window.
4. Click on the **Add Deployment Receiver** link as shown in the following screenshot:

Step Two - Configure Deployment Servers
Configure deployment servers for the web project.

[Add ASR \(Deprecated\)](#) [Add Deployment Receiver](#)

Provide details of the Deployment Receiver to add

Type: ←

Display Name: CIGNEX Test1

Display Group:

Transport Name: default

Host: localhost *

Port:

URL:

Username:

Password:

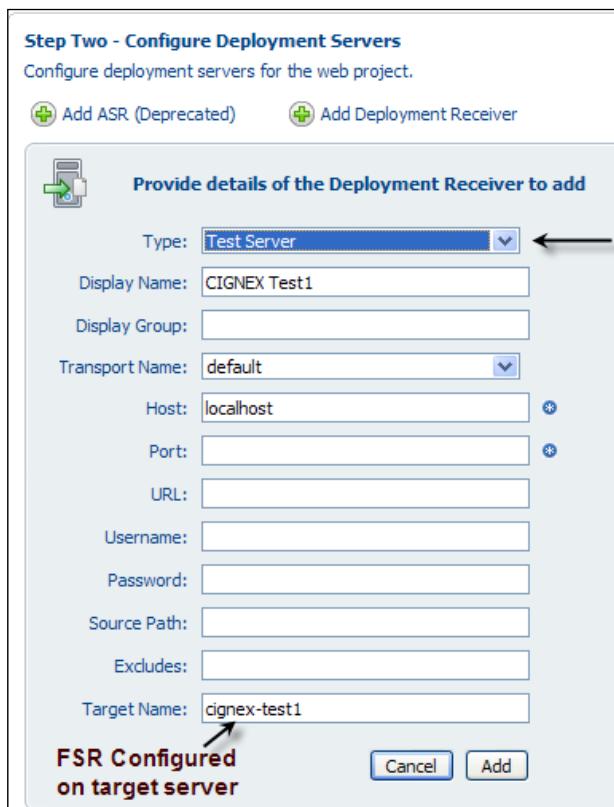
Source Path:

Excludes:

Target Name: cignex-test1

**FSR Configured
on target server**

[Cancel](#) [Add](#)



- For **Type**, select **Test Server**, specify the **Display Name**, **Host name**, and the **Target Name**. Click on the **Add** button.

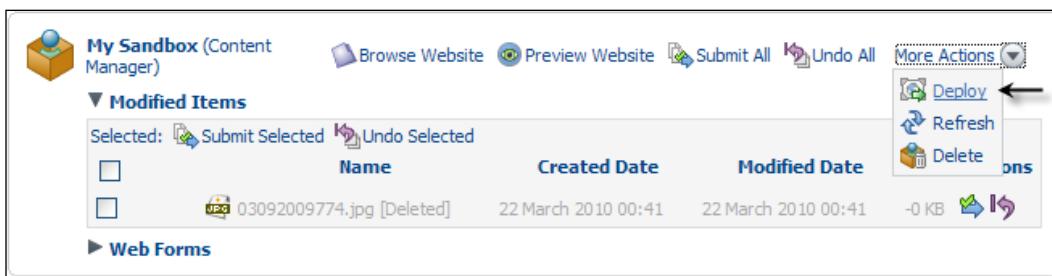
Similarly configure another test server, say with "cignex-test2" as the target.

[ Ensure that the FSR is running on the test server. The targets "cignex-test1" and "cignex-test2" are configured in FSR.]

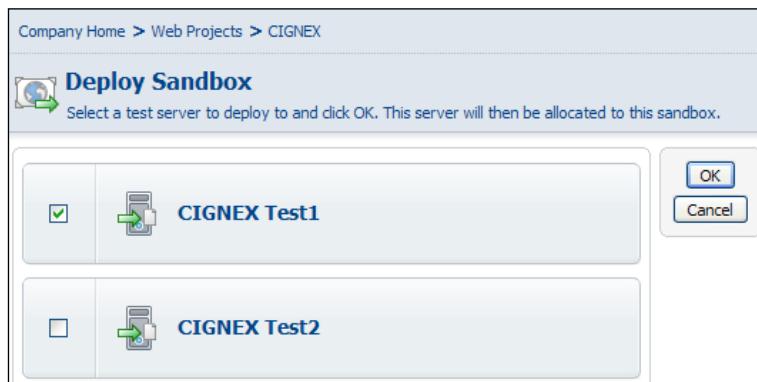
Deploy to a test server

Let's say, you as a content manager would like to deploy your User Sandbox to the test server for testing purposes.

Go to your User Sandbox and from the **More Actions** menu choose **Deploy** as shown in the following screenshot:



The **Deploy Sandbox** window displays, listing all of the unallocated test servers as shown in the next screenshot. Select a test server to use (only one test server can be allocated to a sandbox at a time), and click on **OK**. The Monitor Deployment information displays once the deployment completes. If an error occurs, the reason for the error is shown under the **Deployment Failed** message:



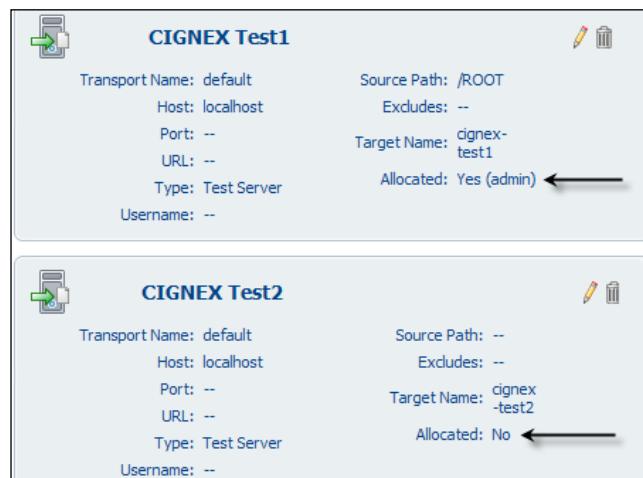
Preview the content

You can preview the content deployed on the test server either using Apache or application servers such as Tomcat or PHP, as per your FSR configuration settings.

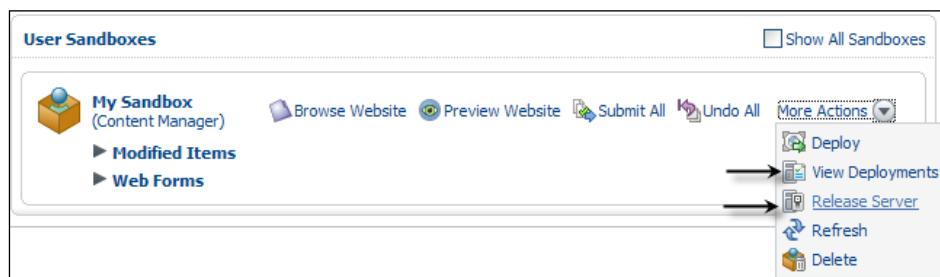
Release the test server

Once a test server has been deployed to, it is allocated to the user or workflow that performed the "Deploy".

The administrator or content manager can see what test server is allocated to which sandbox by going to the **Edit Web Project Wizard** and viewing the **Allocated** field. Hovering over the **Yes** label will reveal the actual store name as a tooltip.



Test servers allocated to User Sandboxes can be released by the owner of the sandbox, the administrator, or the content manager. The **Release Server** option is available in the **More Actions** menu as shown in the following screenshot. A user can also view the deployment history by clicking on the **View Deployments** option available in the **More Actions** menu:



Test servers allocated to review sandboxes are automatically released by the system upon completion of the workflow. Once the user or workflow has finished with the test server, it is released and returned to the pool of test servers.

Deploying from workflow

When you have a test server configured, you can deploy the content to that test server from the workflow process, similar to the way you deployed it from User's Sandbox. Refer to the following screenshot. You don't have to explicitly release the test server. Test servers allocated to the workflow review sandboxes are automatically released by the system upon completion of the workflow:

The screenshot shows the 'Manage Task: Review' interface. At the top, there is a header bar with the title 'Workflow Review Task'. Below the header, the main area is divided into sections: 'Task Properties', 'Reviewers', and 'Resources'.

- Task Properties (General):**
 - Identifier:** 11
 - Description:** submitted to Amita for review
 - Submission Label:** submit for review
 - Launch Date:** None
 - Auto Deploy:**
 - Status:** Not Yet Started
 - Comment:**
- Reviewers:**
 - Type of Review:** Serial
 - Reviewers:** Amita Bhandari [amita]
- Resources:**

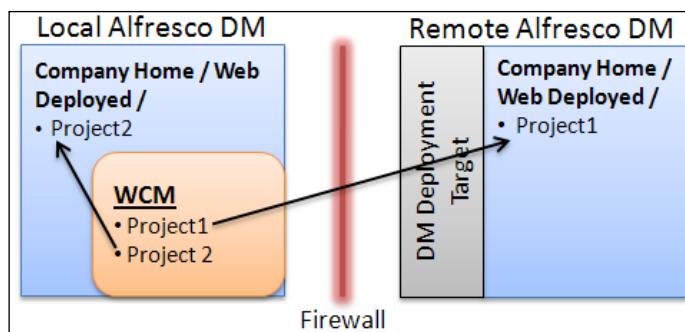
Name	Description	Path	Created	Modified	Expiration Date	Actions
03092009774.jpg [Deleted]		/ROOT	22 March 2010 00:41	22 March 2010 00:41		

A red callout with the text 'Click to deploy to Test Server' points to the 'Deploy' button located in the 'Reviewers' section. The 'Deploy' button has a green icon and the word 'Deploy' next to it.

Deploying from Alfresco WCM to DM repository

Starting from Alfresco 3.3 version onwards, the WCM deployment facilities have been enhanced to add an additional deployment target. This additional deployment receiver allows the WCM content, authored and stored in Staging Sandbox, to be deployed to local and remote Alfresco repositories (Alfresco DM) as shown in the following figure.

The DM Deployment Target receives a deployment from an Alfresco WCM authoring environment and puts the content into the workspace spaces store where it can be used to support a dynamic website. This feature provides greater flexibility in moving an approved staging content from WCM to DM (Document Management). The Alfresco Deployment Receiver is configured as a sub-system, and a new Data Dictionary folder **Web Deployed** is configured by default as the deployment target.



Setting up Alfresco DM as the deployment target

In order to set up Alfresco DM as the deployment target, you will have to edit the global properties file and then restart Alfresco.

Go to the `/tomcat/shared/classes/` folder, open the `alfresco-global.properties` file, and add the following two lines:

```
deployment.dmr.consolidate=true  
deployment.dmr.name=alfresco
```

Now restart Alfresco to activate the deployment target.

The DM Deployment Target is a target that is registered with the repository-based WCM Deployment Engine. By default, its target name is "alfresco". Although of course, this can be changed through the configuration `deployment.dmr.name`.

The authoring environment for a WCM web project consists of a set of related AVM stores. The different stores have a naming convention for their store names. The consolidate flag (`deployment.dmr.name=true`) says to deploy all of these related stores to the same location. If it is turned off by setting `deployment.dmr.consolidate` to false, there will be a separate path for each store and content will be duplicated in the DM store.

Deploying to DM

Go to the Alfresco WCM web project and configure the DM deployment receiver. The following are the steps to configure the DM deployment receiver from Alfresco WCM Project.

1. Navigate to **Company Home | Web Projects | <web project name>**.
2. Select **Edit Web Project Settings** from the **Action** menu.
3. Click on **Next** to reach the **Configure Deployment Servers** window.
4. Click on the **Add Deployment Receiver** link and fill up the following values in the form:

Type = Live Server

Display Name = [Some Name]

Transport Name = default

Host = [localhost for local server or enter host name or IP address for external server]

Port = 50500

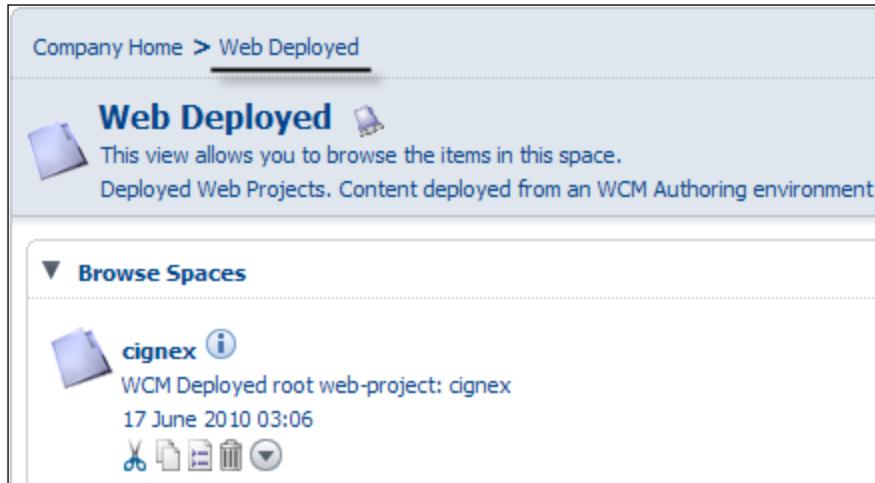
Username = Target Alfresco DM username

Password = Target Alfresco DM password

Target Name = alfresco

5. Click on the **Add** button and then the **Finish** button.
6. Now go to that WCM project's Staging Sandbox and deploy a snapshot to the new DM target.

If you log in to Target Alfresco DM and go to the **Company Home** space, you will notice a new project folder in the **Company Home | Web Deployed** space as shown in the following screenshot:



Summary

In this chapter we learned:

- Alfresco provides both static as well as dynamic delivery models.
- You can configure the Alfresco stage environment to deploy the selective content to external live servers and test servers.
- You can also set up the web project for auto deployment wherein the content is automatically deployed to live servers upon workflow approval.

8

Managing Multiple Websites Using WCM

This chapter covers information about managing multiple web projects using one installation of Alfresco WCM. This means you can leverage a single instance of Alfresco WCM to stage and manage many websites. This chapter focuses on reusing assets such as images, forms, and workflows across multiple web projects. This chapter also introduces you to a concept called "layered folder", where you could logically use a folder in many websites without copying the content in multiple places.

By the end of this chapter, you will have learned how to:

- Configure and use multiple web projects
- Reuse forms, templates, and workflows across many websites
- Use a single set of media assets across multilingual websites
- Set up and use layered folders

Multiple web projects

Within a single instance of Alfresco you can create as many web projects as you can.

The screenshot shows the 'Web Projects' view in Alfresco. At the top, there's a header bar with 'Company Home > Web Projects'. Below it, a sidebar says 'Web Projects' and 'This view allows you to browse the items in this space. Web Content Management Spaces'. On the right, there's a toolbar with 'Add Content', 'Create' (with a dropdown menu), 'More Actions', and 'Icon View'. The main area is titled 'Browse Spaces' and lists three entries:

Space Name	Created
CIGNEX	22 March 2010 02:02
Web Project1	4 April 2010 17:03
Web Project2	4 April 2010 17:03

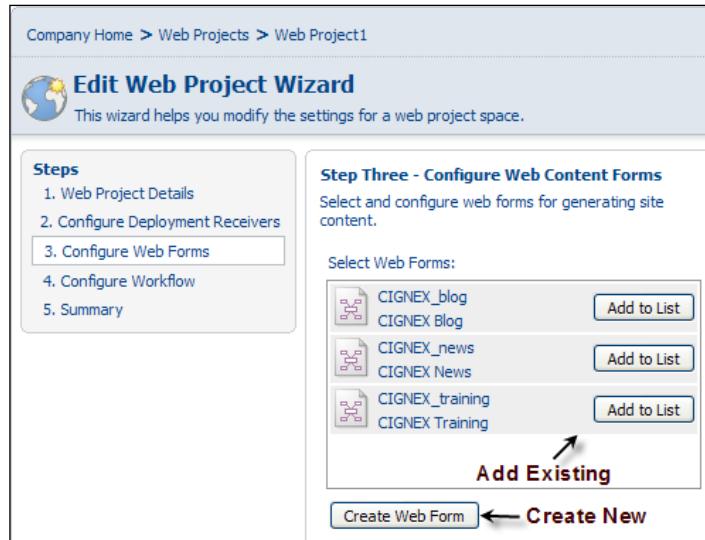
At the bottom, there's a pagination control: 'Page 1 of 1'.

You can have common assets shared across many of these web projects. Once created, you can use the same web forms, templates, workflows, and deployment targets across multiple projects. You can also have the same set of users managing these multiple sites. For example, a user could be a Content Manager on **Web Project1** and Content Reviewer on **Web Project2**.

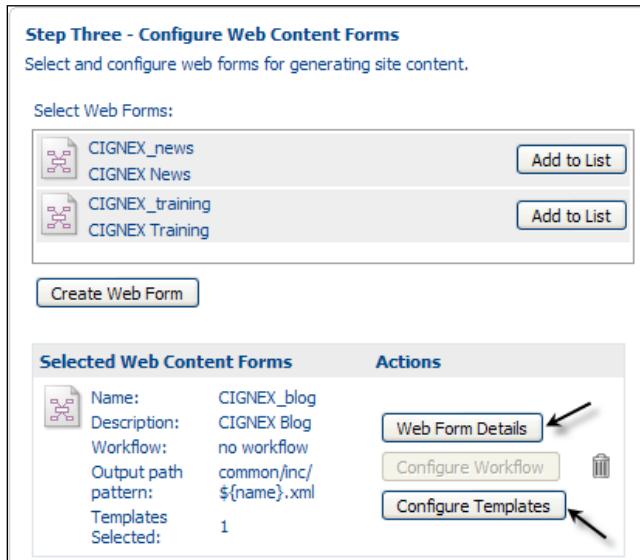
Reusing forms, templates, and workflows

When creating a new project or editing the web project settings, you will notice the list of all of the available web forms. Basically, the forms that are defined in **Company Home | Data Dictionary | Web Forms** are available to all of the web projects along with templates and workflows associated with those forms.

You can add the selected web forms to a project by clicking on the **Add to List** button, as shown in the following screenshot:



Once a web form is added to a web project, you can overwrite the form details such as the output filename pattern, you can configure the workflow locally as per the web project approval process, and you can also configure the output template settings as shown in the following screenshot. These are very flexible ways of defining the forms and workflows globally, and overwriting them locally:



Using a web project as a template

In some scenarios you might want to create web projects that are similar in nature. For example, marketing websites for each product might have similar features, and they might be managed by the same set of people following similar workflow approval processes. The customer extranet websites for each of your customers might fall into this category.

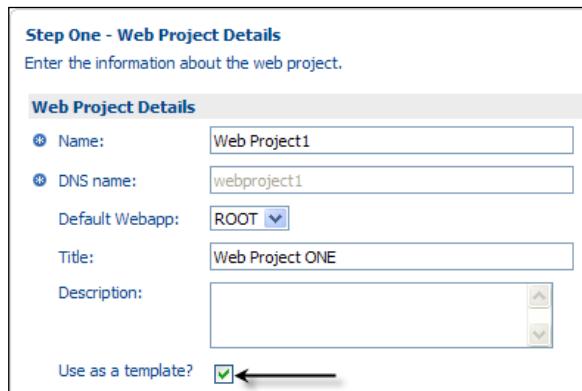
In such scenarios using a typical website as a template to recreate many such websites will not only save you time, but also sets a well-defined process in the organization. You can select such a web project as a template, as shown in the following screenshot:

Step One - Web Project Details
Enter the information about the web project.

Web Project Details

④ Name:	Web Project1
④ DNS name:	webproject1
Default Webapp:	ROOT
Title:	Web Project ONE
Description:	(empty)

Use as a template? ←



Now while creating a similar website, you can create a new web project based on an existing template web project. The Staging Sandbox structure, web forms, workflow, and users will be copied from the selected web project. This must be done while creating a new web project, as shown in the following screenshot:

Company Home > Web Projects

Create Web Project Wizard
This wizard helps you create a new web project space.

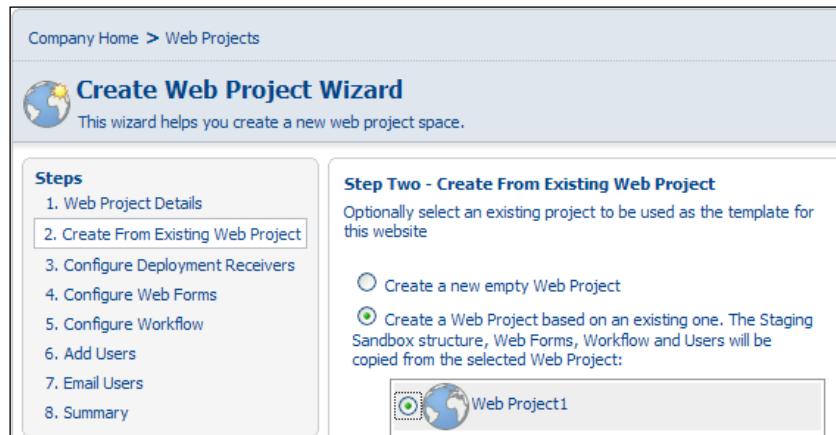
Steps

- 1. Web Project Details
- 2. Create From Existing Web Project
- 3. Configure Deployment Receivers
- 4. Configure Web Forms
- 5. Configure Workflow
- 6. Add Users
- 7. Email Users
- 8. Summary

Step Two - Create From Existing Web Project
Optionally select an existing project to be used as the template for this website

Create a new empty Web Project
 Create a Web Project based on an existing one. The Staging Sandbox structure, Web Forms, Workflow and Users will be copied from the selected Web Project:

 Web Project1



Once a new web project is created, you can always overwrite the existing configuration. You can also create new forms, workflows, and structures as needed.

Managing multiple websites using a single web project

You can also use a single web project to have multiple websites. You can group all of the assets related to a website into a folder in a web project. Hence, a web project could have many such folders and each one could be interpreted and managed as a website.

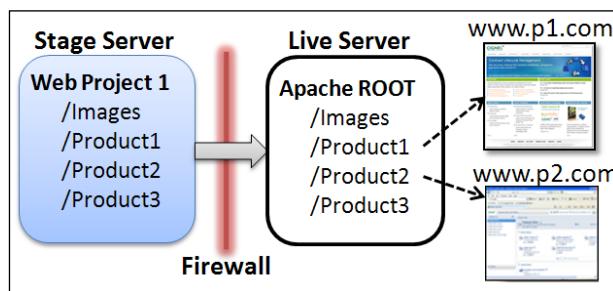
This approach is better when compared to having a separate web project for each website if you have the following requirements:

- All of the content in these websites have similar workflow approval processes
- The content is managed by the same set of people
- All of the websites have a similar look and feel
- The deployment (going live) to the product server or servers for all of these websites happens at the same time

Basically, this is good for small to medium websites where majority of the content is static.

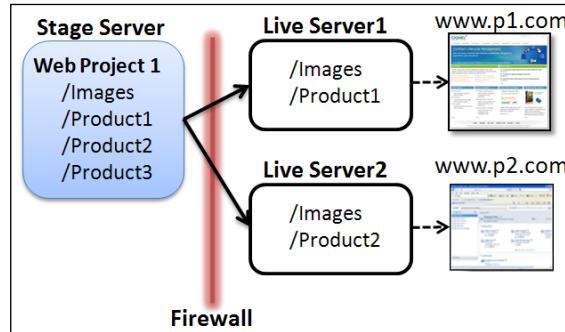
Setting up multiple URLs on the target server

One scenario could be deploying all of the folders to a specific target server. On that target server you could use web server proxy to have a specific URL pointing to a specific folder, as shown in the following diagram:



Setting up FSR for each target website

Another scenario could be deploying selected folders to a specific live server as shown in the following diagram. This can be done by using the Source Path and Excludes fields of deployment, while configuring the FSR on a stage server. Refer to *Chapter 7, Content Delivery and Deployment* for more details on deployment:



Creating many webapp folders

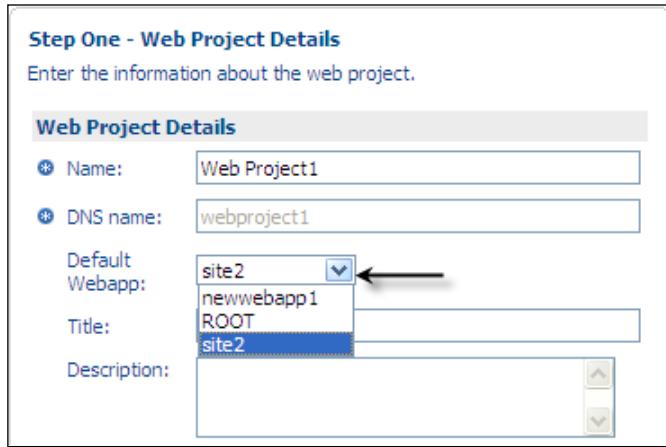
Another way of creating multiple websites within a web project is to use multiple webapp folders.

The webapp folder is the web application folder used in application servers such as Tomcat or JBoss. A web application exists as a structured hierarchy of directories. The root of this hierarchy serves as a root for serving files that are part of this context. For example, for a web application located at **site2** in a web server, the `index.html` file located at the base of the web application hierarchy can be served to satisfy a request to `http://some_url/site2/index.html`.

In Alfresco WCM, each web project has only one web application (webapp) folder and it is named as **ROOT**. To create another webapp folder for a specific web project, navigate to the **Actions** drop-down menu and click on **Create Webapp Folder**:

A screenshot of the Alfresco WCM interface. The URL in the address bar is 'Company Home > Web Projects > Web Project1'. The main content area shows 'Web Project1' with a globe icon. Below it, a dropdown menu 'Current Webapp Folder:' is set to 'site2'. An arrow points upwards from the 'Actions' menu to this dropdown. The 'Actions' menu itself is open, showing options: 'View Details', 'Create Webapp Folder' (which is highlighted with a blue border), 'Edit Web Project Settings', 'Invite Web Project Users', 'Delete All Deploy Reports', 'Browse Website', 'Prev', and 'Delete'. At the bottom of the interface, there is a footer with the text '[260]'.

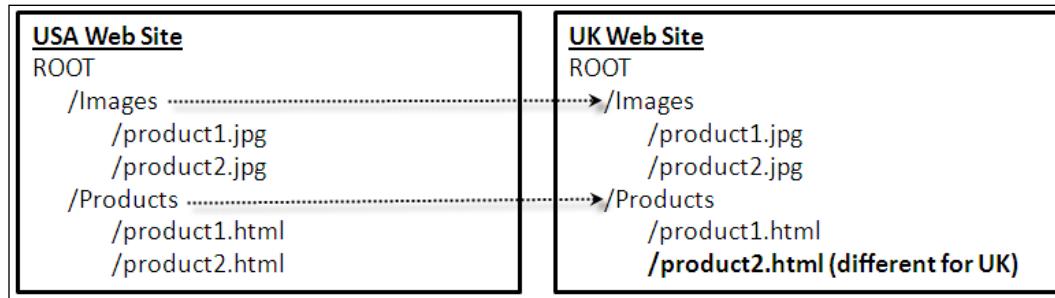
You can also change the default web application. Click on **Edit Web Project Settings** from the **Actions** drop-down menu and change the setting in the **Default Webapp** field as shown in the following screenshot:



Layered folders

Large enterprises usually run global operations. In order to have effective services, sales and marketing, they usually have many regional websites. Most often the information in these websites is 80 percent common (global) and 20 percent different (local). However, enterprises end up replicating or duplicating the 80 percent of such common information in each and every regional website.

Let's consider a scenario where an enterprise is having two websites, one for USA-based customers and the other one for UK-based customers with the pages and files as shown in the following image:



In order to make the USA website contents of the /Images and /Products directories on the UK website, you would have two options:

- Option 1 is to make a separate copy for the UK website. It means a specific file has two different copies. The issue with this approach is that if the USA website updates an image (say product1.jpg), then those updates are not applied on the UK website. You need to make sure you update those changes on the UK website as well. If the size of the website is large with hundreds and thousands of files and pages, then it will be very expensive to maintain two copies of the files.
- Option 2 is to make a symbolic link of the USA website for UK. Now any updates done on the USA website are seen on the UK website as well. It is like an alias or a Unix-style symbolic link. The advantage of this approach is that there is only one copy of files to manage, which is easier. This issue with this approach is that if the UK website needs to make a few changes to a page (say product2.html), then it is not possible.

Alfresco WCM provides a strong feature called **Transparent Folder** where it is possible to have both option 1 and 2. You can have only one copy of the files to maintain globally and also have the flexibility to update a few of them locally.

For example, consider that the UK website uses **Images** and **Products** as two transparent folders from the USA website. It means that the USA website has the master copy (only one copy) and the UK website has symbolic links to those folders.

Now any updates done in the USA website will automatically be seen on the UK website. Say you updated product1.jpg on the USA website, then the changes are seen on the UK website.

If you make any modifications to the files on the UK website, then a separate copy of that file alone will be created on the UK website. For example, if you update **product2.html** on the UK website, then a separate copy is made for that file for the UK website. From this point onwards, there will be two separate copies of the **product2.html** file with different versions and content in these two websites.

Creating a transparent folder

Let's create a web project with a few folders and files.

1. Create a web project called **USA Web Site**.
2. Go to **User Sandbox** and create folders called **Images** and **Products**.
3. In the **Images** folder, add two images (say product1.jpg and product2.jpg).

4. In the **Products** folder, add two HTML pages (say product1.html and product2.html).
5. Submit all of these files and folders to the Staging Sandbox.
6. Browse the Staging Sandbox to view these folders and files.

Company Home > Web Projects > USA Web Site

Website 'USA Web Site' sandbox 'admin'
Use this view to browse the files and folders within the sandbox for a web project.

ROOT

Search Website:

Browse Folders

Name	Creator	Created Date	Modifier	Modified Date	Type	Actions
Images	admin	5 April 2010 01:37	admin	5 April 2010 01:37	Folder	
Products	admin	5 April 2010 01:37	admin	5 April 2010 01:37	Folder	

Now create another web project called **UK Web Site**. Do not create any folders or files in **UK Web Site**. Go to the Staging Sandbox of the UK Web Site and click on the **Create Layered Folder** link as shown in the following screenshot:

Company Home > Web Projects > UK Web Site

Website 'UK Web Site' sandbox 'Staging'
Use this view to browse the files and folders within the sandbox for a web project.

[Create Layered Folder](#)

You can create a layered folder using a target folder from a target web project. Create the **Images** folder in the UK Web Site, using the **Images** target folder from the **USA Web Site** target web project as shown in the following screenshot:

Company Home > Web Projects > UK Web Site

Create Layered Folder
Create a layered folder in the website.

Properties

Name:
Title:

Target

Web Project:

Target Path:



It is not mandatory to use the same name for the layered folder as the target folder name.



You can use any name for the folder in the UK Web Site. It is not mandatory to use "Images" as folder name in the UK Web Site. However, to maintain simplicity and to avoid naming conflicts, it is advised to use the same folder names.

Similarly, you can link to any number of web projects and folders. For example, you can use the "Images" folder from the USA Web Site and "Videos" folder from the Germany website and so on.

You will notice that a new folder called **Images** is created in the **UK Web Site** with the **Type** specified as **Layered Folder** as shown in the following screenshot. Click on the **Images** folder and notice two images. These were the images that were created in the USA Web Site and are automatically available to the UK Web Site due to layered folder.

Name	Creator	Created Date	Modifier	Modified Date	Type	Actions
Images	admin	5 April 2010 02:45	admin	5 April 2010 02:45	Layered Folder	

Similarly, create a transparent folder for the "Products" folder. Now you will notice two transparent folders in UK Web Site.

Updating a source file

Updating the file in **USA Web Site** will automatically update the file in **UK Web Site**.

Browse to **USA Web Site | User Sandbox | Images** and update the **product1.jpg** file. Submit the changes to the Staging Sandbox. Notice the timestamp of the file and also the versions as shown in the following screenshot.

Company Home > Web Projects > USA Web Site

Details of 'product1.jpg'

View details about the file.

Links		
View In Browser	Download Content	
Preview File	Alfresco Node Reference	
Properties		
Name:	product1.jpg	
Title:	product1.jpg Edit	
Description:	Product one image Edit	
Creator:	admin	
Created Date:	5 April 2010 01:22	
Modifier:	admin	
Modified Date:	5 April 2010 02:55	
Version History		
Version	Modified Date	Actions
2	5 April 2010 02:52	View Revert
1	5 April 2010 01:22	View Revert

Now browse to the **UK Web Site** and notice that the image is updated automatically including the versions as shown in the next screenshot. Basically, the image is not updated; we are seeing the same image information through the link.

Company Home > Web Projects > UK Web Site

Website 'UK Web Site' sandbox 'Staging'

Use this view to browse the files and folders within the sandbox for a web project. [Preview Website](#) [Create](#)

ROOT > Images

Search Website:

Browse Files							Items
Name	Size	Creator	Created Date	Modifier	Modified Date	Type	Actions
product1.jpg	270.51 KB	admin	5 April 2010 01:22	admin	5 April 2010 02:55	File	View Edit
product2.jpg	270.51 KB	admin	5 April 2010 01:35	admin	5 April 2010 01:35	File	View Edit

If a file is deleted in the source web project, the file will automatically be removed from the web projects where the parent folder is used as a layered folder.

Updating the destination file

If you make any modifications to the files in layered folder, then a separate copy of that file alone will be created. From that point onwards, they are considered as two different files.

Refer to the following screenshot. For example, if you update `product1.jpg` on the USA Web Site, then a separate copy is made in the UK Web Site. From this point onwards, there will be two separate copies of the `product1.jpg` file with different versions and content on these two websites.

The screenshot displays two side-by-side web pages for the file 'product1.jpg'.
Left Page (USA Web Site):
- Title: Details of 'product1.jpg'
- Properties:

- Name: product1.jpg
- Title: product1.jpg
- Description: Product one image
- Creator: admin
- Created Date: 5 April 2010 01:22
- Modifier: admin
- Modified Date: 5 April 2010 02:55

Right Page (UK Web Site):
- Title: Details of 'product1.jpg'
- Properties:

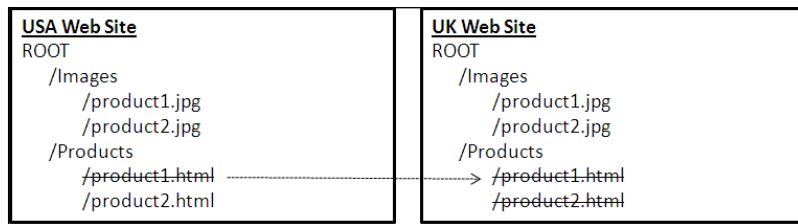
- Name: product1.jpg
- Title: product1.jpg
- Description: Product one image
- Creator: admin
- Created Date: 5 April 2010 01:22
- Modifier: admin
- Modified Date: 5 April 2010 03:16

Both pages show a 'Version History' table:

Version	Modified Date	Actions
2	5 April 2010 02:55	[View] [Revert]
1	5 April 2010 01:22	[View] [Revert]

Deleting files

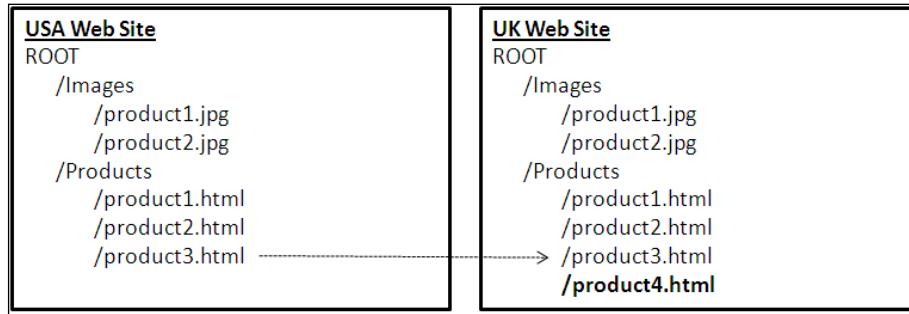
If you delete a file from the source, then the file will automatically be deleted on all of the destinations. For example, if you delete a file `product1.html` in **USA Web Site**, then the file will automatically be removed from **UK Web Site** as well.



On the other hand, if you remove **product2.html** from **UK Web Site**, then it is considered as a separate copy and the original file in source location (**USA Web Site**) will not be deleted.

Adding new files

If you add a file in a source, then the file will automatically be added in all the destinations. For example, if you add a file **product3.html** in **USA Web Site**, then the file will automatically be added to **UK Web Site**.



On the other hand, if you add **product4.html** in **UK Web Site**, then it is considered as a new addition only in the UK Web Site.

Summary

In Alfresco WCM, you can use a single web project to manage multiple websites. You can have many web projects in a single instance of Alfresco. These web projects can share common web forms, templates, and workflow. The layered folder concept is very powerful, where you get both efficiency and flexibility in managing content across multiple websites.

9

Alfresco Surf and Web Editor

Alfresco Surf is an application framework for developing and delivering dynamic websites. Surf enables the rapid creation of next-generation, productivity-focused web applications designed to deliver your content to websites and web users.

Alfresco Web Editor is an application developed using the Spring Surf platform. It provides in-context editing capabilities for Alfresco repository content. In this chapter you will learn:

- Surf architecture and single-tier/two-tier applications
- Surf APIs used by presentation tier and model objects
- Designing a web application on the Surf platform
- Communication of Surf Applications with Alfresco WCM using web scripts
- Creating a rich User Interface application using YUI libraries
- About Alfresco Web Editor
- Alfresco Web Editor Tag Library
- Sample Web Application using Alfresco Web Editor

Alfresco Surf platform

Alfresco Surf is a lightweight, scriptable web framework built on top of the Alfresco web scripts technology and Templating runtime. It provides a highly extensible and customizable web framework for page layout and a component framework for building web applications. It works well with web scripts, can remotely call web services (that is, REST), and also has multiple programming options. Applications built using Surf can be deployed on a standalone server. It has zero dependencies on the Alfresco repository but use it as a backend "model". A web application built using the Surf framework has many objects in places like pages, templates, themes, and components. Any application built using Surf can be deployed from the Alfresco Web Project, taking full advantage of Alfresco WCM's enterprise features.

Surf has been designed to leverage web-enabled tools that expose powerful capabilities such as inline editing, drag-and-drop positioning of content, point-and-click template selection, and so on. These tools make it possible for users with little technical training to manage dynamic websites. Alfresco Surf is a lightweight, scriptable Java-based web framework that provides the ability to quickly create dynamic websites using concepts such as pages, templates, themes, and components. The Surf platform was designed to specifically make it:

- Lightweight (around 8 to 9 MB) and scriptable. No server restart or content reloading is required. You only need to refresh.
- Easy to customize by using XML configuration files and it also has a Developer API.
- Work well with Alfresco WCM – it reads and writes from the Alfresco repository.
- Equipped with an out-of-the-box and extensible Site Dispatcher, Component model, or Site construction model.

Applications built with Alfresco Surf can be considered as standalone and deployed on any application server. The application can also be used in Alfresco WCM and deployed from Alfresco Web Project spaces to production servers while making use of Alfresco WCM features. This gives the advantage to use many features of Alfresco WCM that are required for developing any website. Some of the features are listed as follows:

- Approval of Surf components using workflows
- Allows to make and preview changes without any additional server requirement
- All of the changes that are on the Staging box can roll back to any version
- Alfresco Surf-powered applications can retrieve all of their runtime data from the local disk, the classpath, or from the Alfresco repository via REST

The Surf platform consists of several integrated parts. To help facilitate this, the Surf platform uses a lightweight XML-driven model to store all model objects that make up the website. Model objects are things such as pages, templates, components, themes, and chrome.

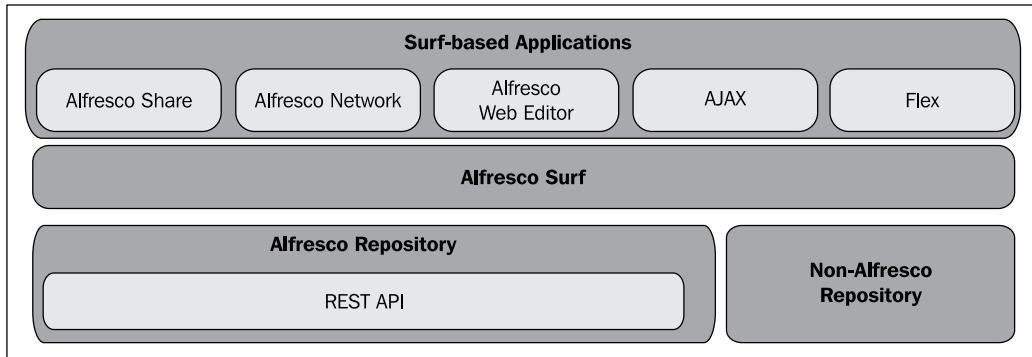
Alfresco Surf RESTful platform uses Yahoo's YUI AJAX library and Adobe Flash for a more interactive user experience. Surf allows you to take a page and split it up into template and components. This allows for a total separation of the look and feel and the components. Both the template and the components have renderers to generate outputs. You can even use PHP renderers, that is, you can use the model of Alfresco but the actual page is rendered in PHP. Most of the Surf components are envisioned to be written using web scripts. With web scripts and their rich underlying API for extension, no Java coding or server restarts are the most common, and the most powerful, rendering facility. You can also build your own renderers or make use of the out-of-the-box renderers such as JSP, Java Bean FreeMarker Templates, and HTML available. Templates make it possible to define a page layout once and then reuse it across a large set of pages. In each of these template languages, the essential goal is to layout the placement of regions (or slots) into which components can be bound. You can create single- and two-tier applications.

Applications using the Alfresco Surf platform

The Surf platform is used by the Alfresco 3.0 release of products to provide a uniform suite of interchangeable functionality. These applications include:

- **Alfresco Share:** Enhanced collaboration and document management services. It provides out-of-the-box web client templates for creating social networking sites. These sites in turn use a rich library of downloadable and shareable themes, components, templates, and application logic.
- **Alfresco Dynamic Website:** This is aimed especially at business users, which allows the rapid self-assembly of web applications and web experiences with minimal IT intervention. Comes with an in-context, drag-and-drop toolkit and allows teams to construct their own web applications that feature user interactions and collaborations.
- **Alfresco Network:** Provides access to Alfresco news, ticket details and resolution, and new services, such as the Knowledge Base and the Web Component library. This is available to the Alfresco Enterprise users.
- **Alfresco Web Editor:** A visual, drag-and-drop designer for building websites with the Alfresco Surf platform. It features graphical overlays that facilitate the building of your website's pages, navigation structure, templates, and presentation layout. We will learn in detail about Web Editor in a later section of the chapter.

All of these applications are presently being built on top of the Surf platform. The Surf platform itself is a product that our community and customers may wish, and are encouraged, to use in building their own web applications.



Alfresco Surf architecture

Surf architecture shows a deep appreciation for the Web 2.0-inspired architecture and web application mash-ups. It's important to point out that although tools exist to reduce the need for technical skills, the platform does not eliminate or discourage access to its internals for those who need it.

Surf utilizes the MVC pattern using various components and objects. MVC stands for Model-View-Controller.

MVC pattern

The Dispatcher Servlet can be considered as a controller. It is responsible for rendering the view. It considers the incoming request context and decides what and how to render. It constructs the page view by looking at the related model components or templates to render using different possible rendering engines. The dispatcher can receive requests for the following:

- **A specific page:** A page ID may be given directly to the dispatcher.
- **A type of page:** For instance, the user may be requesting the login page. The application may actually have several login pages (for customers, employees, and so on) and it must select one and render it back.
- **An object:** An object ID may be given to the dispatcher. The dispatcher must then figure out which page to use to render this object.

- **The Surf model:** This can be considered as the model of the MVC architecture. Model is an entity that holds data. The Surf platform uses a lightweight XML-driven model to store all model objects that implement the majority of common websites and web application object types. XML files are effectively bound together to allow a page to be loaded and rendered. You can also define your own object model and extend the out-of-the-box object model. These include objects such as Pages, Templates, Components, Chromes, and Themes.

The XML is stored in sub-directories in the specified location: <install-application>/WEB-INF/classes/alfresco/site-data.

Within this directory, you will see the following sub-directories, one for each type of model object that is managed by the Surf platform:

```
/site-data/chrome  
/site-data/component-types  
/site-data/components  
/site-data/configurations  
/site-data/content-associations  
/site-data/page-associations  
/site-data/page-types  
/site-data/pages  
/site-data/template-instances  
/site-data/template-types  
/site-data/themes
```

View (Renderers): View is for the presentation layer. The model objects such as components, templates, and page can be rendered using any of Surf's supported rendering engines, which internally return data in HTML, XML, JSON, and many more.

The view can be constructed using several rendering engines. These include:

Web scripts: Web scripts are used most commonly to render components. Web scripts themselves follow MVC architecture. The Web scripts take advantage of JavaScript (behaving as a controller), FreeMarker (for presentation processing), and model objects (for business logic). They can be stored in the specified location: <install-application>/WEB-INF/classes/alfresco/site-webscripts.

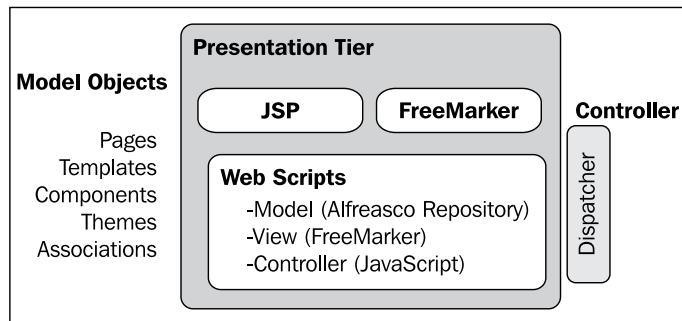
JSP: The JSP renderer is used to render either components or templates, and provide you with access to the Java language and the Surf platform Java API. JSPs can be stored in the specified location: <install-application>/tomcat/webapps/alfresco/jsp/components.

FreeMarker: FreeMarkers can be stored in the specified location: <install-application>/WEB-INF/classes/alfresco/templates.

By using the Surf framework you can create both single-tier and two-tier applications. Tiers can be considered as the physical deployment of different application layers.

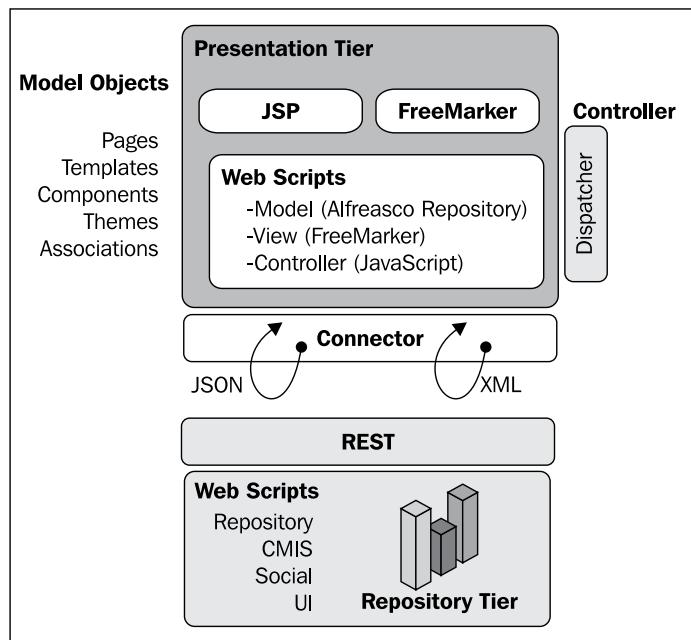
Single-tier application

The user interface of a web application, the middleware, and the data access all are contained in one package. A system where all the layers are bundled together on a single server is generally single tier. Single-tier architecture has the interface, business logic, and database highly coupled. Installing Surf as is, with just the page layout and component framework, is single tier. A one-tier application also follows MVC. Here, model represents objects like pages, components, and templates, controller represents Dispatcher servlets, and renderer represents FTL and Web scripts. In this case, FTL code will be heavy, and rather than depending on other parties, it will call the User Interface framework optionally and will message data. It will also create data in the form of HTML or XML. If we have a complex User Interface, it is not advisable to keep the complete logic in one Web Script. In that case, we can develop the application using two tiers.



Two-tier application

With two-tier client/server architectures the user system interface (Client) is usually located on the user's desktop environment and the database management services are usually in a server that is a more powerful machine that services many clients. Processing management is split between the user system interface environment and the database management server environment. The client performs most of the business logic as well as the presentation, updating shared data by communication with the database. Here we separate the data retrieval and data presentation part. So, one set of web scripts (Data web script), will be responsible for data retrieval and sending them in JSON, XML (ATOM), or other serialized data format present on the Repository tier. Another set of web scripts (Presentation web script) with some UI framework, may be YUI (Yahoo User Interface), will call those Data web scripts and will be responsible for rendering data present on Presentation tier in HTML format. In case of two-tier FTL or presentation web scripts, the code will be minimal and will be more centric to creating data in JSON/XML format. Consider the client is on one server (Presentation tier) accessing the Alfresco repository on a different server (Repository tier) using remote calls. Alfresco Share and Web Editor are two-tier applications, as they remotely connect to a different Alfresco repository using remote calls. The Alfresco Web Editor performs most of the business logic and presentation. It uses the Alfresco repository to fetch and store the data using Alfresco web scripts.



Surf model objects

In the Surf framework models are represented in the form of XML. XML files are effectively bound together to allow a page to be loaded and rendered. A model object contains information about site construction. Model defines pages, page layouts, and components within the page. We already had an overview about model in the previous section. Let's have an overview about these elements:

Objects	Description
Configuration	A Configuration object is a collection of XML, mostly used by the internal system. It is defined for a site.
Component Type	A Component Type is something that the website builder can grab at and instantiate in many places across the web application. They bind the new instances into pages by snapping them into regions (or slots). They can be considered as widgets.
Component	A Component is an instance of a Component Type that has been "bound" into a region.
Template Instance	It points to Free Marker renderer classes.
Template Type	It specifies FreeMarker or web scripts as the renderer.
Template renderer	It defines the look and feel of the page.
Page	A Page is the navigation page in a web application. It consists of templates and components.
Page Association	The Page Association objects allow you to link two pages together.
Content Association	It associates a document to the page. It can associate either a specific document or all documents of specific type and can be associated to a page instance or page type.
Chrome	Chrome describes border elements around a region or a component. Using Chrome, you can also include styling elements and drag-and-drop capabilities into a Page.

Surf API

We will learn some of the surf APIs used by presentation tier. The objects and methods are available to templates and components within Alfresco Surf. The main point to be noted is that these methods are not available for Repository-tier web scripts. Both FreeMarker template API and JavaScript API use the common object model. The list of commonly used root objects and methods is as follows:

Root objects	Properties and Methods	Description
context (context of current page)	<code>id</code> <code>pageId, page</code> <code>templateId, template</code> <code>theme, themeId</code> <code>user</code> <code>ContentId, content</code>	The internal manage ID for the current request. The ID of page and page object being rendered. The ID of template and template object being rendered. The ID of theme and current theme object being rendered. The current user. The ID of content and content object being rendered.
	<code>properties</code> <code>authenticated</code>	Associative array of all context values. Returns true if current user is not a guest.
user (current user)	<code>id</code> <code>properties</code> <code>name, fullName, firstName, MiddleName, lastName, mail, telephone, mobilePhone, location, biography</code> <code>organization, jobTitle, companyPostcode, companyTelephone, compnayFax, companyEmail, companyAddress1, companyAddress2, companyAddress3</code> <code>skype, instantMsg</code> <code>save()</code> <code>getUser(userId)</code>	User identifier. Associative array of all user values. The details related to user identification. The details related to user's organization. The user's online contact details. Saves changes to the user's properties. Retrieves user object.

Root objects	Properties and Methods	Description
content (only available if an object ID is provided as part of the page URL)	id	The ID of content object.
	typeId	The typeId of content object.
	properties	Associative array of all object properties.
	timestamp	The time when the object was loaded.
	endpointId	The ID of endpoint from which object was loaded.
	isLoaded	Whether the object is successfully loaded.
	statusCode, statusMessage	The status code and message when the object is loaded.
	text, xml	The content in text and XML format.
	url	The URL helper object.
	id	The ID of page object.
Page (available within context of page renderer)	title, titleId	Title of Page definition.
	description, descriptionId	Description of Page Definition.
	theme	Theme ID.
	properties	Properties of custom page definition.
	context	Page root context path.
url	servletContext	Page root servlet path.
	uri	Page URI.
	url	Page URL.
	queryString	Query string for the URL.
	args	Map of URL arguments.

Root objects	Properties and Methods	Description
remote (connect to remote services, that is, repository tier)	<p>endpointIds</p> <p>connect</p> <p>connect (endpointId)</p> <p>call (uri)</p> <p>GetEndpointName (endpointId), GetEndpointDescription (endpointId)</p> <p>GetEndpointURL (endpointId)</p>	<p>A string [] array of endpoints.</p> <p>Default end point connector.</p> <p>Connector to specified end point.</p> <p>Invokes specific URI on default endpoint.</p> <p>Gets the name and description for an end point.</p> <p>Gets the URL for an end point.</p>
sitedata (site construction helper)	<p>rootPage</p> <p>siteConfiguration</p> <p>components, pageAssociations, pages, templates, contentAssociations</p> <p>componentsMap, pageAssociationsMap, pagesMap, templatesMap</p> <p>newPage (title, description), newTemplate (templateTypeId, title, description), newObject (objectId, objectTypeId)</p> <p>findComponents (scopeId, sourceId, regionId, componentTypeId), findChildPages (sourceId), findPageAssociations (sourceId, destId, assocType)</p> <p>associateTemplate (templateId, pageId), associatePage (sourceId, destId)</p> <p>unassociateTemplate (pageId), unassociatePage (sourceId, destId)</p> <p>getPage (objectId), getTemplate (objectId)</p>	<p>Root Page object for the website.</p> <p>Configuration object for the website.</p> <p>Arrays of all these objects. There are many more methods available.</p> <p>Maps of all these objects. There are many more methods available.</p> <p>Creates new objects. There are many more methods.</p> <p>Looks up objects and returns array of results. There are many more methods available.</p> <p>Binds object together. There are many more methods available.</p> <p>Unbinds objects together.</p> <p>Looks up individual objects.</p>

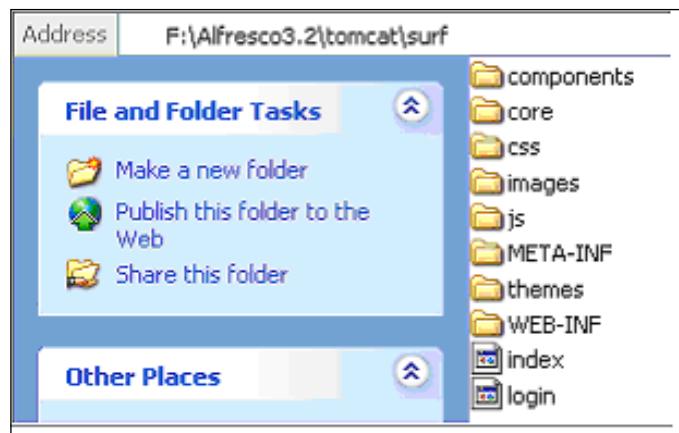
Rendering engines

There are two most commonly used rendering engines – components and templates. Specific objects are available to these engines:

Objects	Template	Components
sitedata	Yes	Yes
context	Yes	Yes
instance	Yes	Yes
user	Yes	Yes
content	Yes	Yes
page	Yes	Yes
theme	Yes	Yes
htmlid	Yes	Yes
url	Yes	No
head	Yes	No

Design site navigation

We will first learn how to create a standalone application using the Surf framework without the use of Alfresco. For this you need to download `surf.war` and keep it in your web or application server. For this chapter, you can place the WAR file in the `<install-alfresco>/tomcat/webapps` folder. This doesn't mean that we are using Alfresco for our development. You can also place the WAR file in any standalone server. Start your server. You will find that the `surf.war` file has exploded and the `surf` folder is created. You can see the following structure created by default:



Design a page

Follow the steps below to create a sample page:

- **Step 1:** Go to the <install-alfresco>/tomcat/webapps/surf/WEB-INF/classes/alfresco/site-data/configurations folder.

- **Step 2:** Open the default.site.configuration.xml file. Insert the highlighted code as mentioned below to create our home page:

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

<configuration>
    <title>Sample Site Configuration</title>
    <description>Sample Site Configuration</description>
    <source-id>site</source-id>
    <properties><root-page>index</root-page></properties>
</configuration>
```

- **Step 3:** Go to the <install-alfresco>/tomcat/webapps/surf/WEB-INF/classes/alfresco/site-data/pages folder.

- **Step 4:** Create a new file with the name given previously in the root-page tag. Create the index.xml file. Insert the following code:

```
<?xml version='1.0' encoding='UTF-8'?>
<page>
    <id>index</id>
    <title>CIGNEX | Open Source ECM, BPM , E Commerce , Portals
    </title>
    <description>Sample Cignex home page</description>
    <template-instance>index</template-instance>
    <authentication>none</authentication>
</page>
```

- **Step 5:** Go to the <install-alfresco>/tomcat/webapps/surf/WEB-INF/classes/alfresco/site-data/template-instances folder.

- **Step 6:** Create a new file with the name given previously in the template-instance tag. Create the index.xml file. Insert the following code:

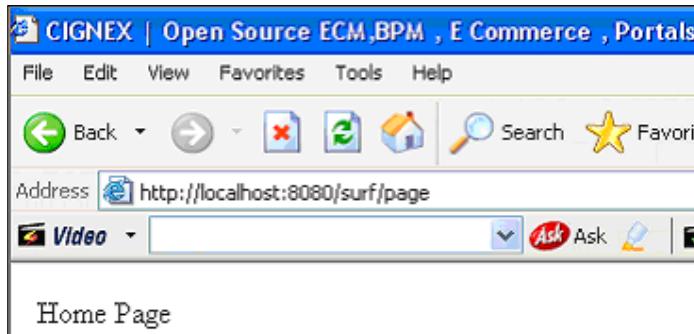
```
<?xml version='1.0' encoding='UTF-8'?>
<template-instance>
    <template-type>index</template-type>
</template-instance>
```

- **Step 7:** Go to the <install-alfresco>/tomcat/webapps/surf/WEB-INF/classes/alfresco/templates folder.

- **Step 8:** Create a new file with the name given previously in the template-instance tag. Create the index.ftl file. Insert the following code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>${page.title}</title>
    </head>
    <body>
        Home Page
    </body>
</html>
```

- **Step 9:** Test the application. You will find your home page is ready now. You should be able to test by hitting the URL of your web or app server. Navigate to the home page URL: `http://<server-name>:<port>/surf`.



You will face a caching issue while using the Surf framework. In order to save server restart during development of an application, you can disable cache for the time period.

1. Edit the /WEB-INF/classes/alfresco/web-framework-application-context.xml file.
2. Search for the word "updateDelay" and change the value to 0. There are two occurrences and properties to change:
`<property name="updateDelay"><value>0</value></property>`
3. Restart your application server for the change to take effect.

OR

By reloading the page, the changes will not be reflected. You have to explicitly clear the cache. Browse to the cache as follows:

`http://<server-name>:<port>/surf/control/cache/invalidate`

Use of a component in a page

The Surf framework binds the component to the specific region of a page and also specifies scope. Using this one can decide the position of the component to be placed in the page.

For instance, the footer should always be at the bottom of the page and the header should always be at the top of the page. As discussed earlier, each component has scopes to be defined. The Surf framework defines three categories of scopes:

- **Global:** Available across all pages. For example, the header and the footer template can have global access, which will be available on any page of the application. These are sections of the page that will not change from one page to the next.
- **Page:** Limited access to a page. The component will be available for the same page where it is defined.
- **Template:** Accessible to a few pages. The component is to be used for some pages but not for all.

Let's try to create a header component, the scope of which will be global. The naming convention for the component is very important. The name is defined as follows:

`Scope.regionId.sourceId.xml`

According to this, we are going to create `global.header.xml`. Here `global` is the scope, `header` is the region, and `global` is the `sourceId`. However, we have not used the third parameter, as it is a global scope component and therefore is used on all pages.

- **Step 1:** Let's create the `global.header.xml` file in the `<install-application>/tomcat/webapps/surf/WEB-INF/classes/alfresco/site-data/components` folder and insert the following code:

```
<?xml version='1.0' encoding='UTF-8'?>
<component>
    <scope>global</scope>
    <region-id>header</region-id>
    <source-id>global</source-id>
    <url>/component/common/header/header</url>
</component>
```

The `<url>` tag is used to call web scripts. You will learn about web scripts in *Chapter 10*. Refer to the chapter to know more details about the web scripts. Web scripts are placed in the `<install-application>/tomcat/webapps/surf/WEB-INF/classes/alfresco/site-webscripts` folder.

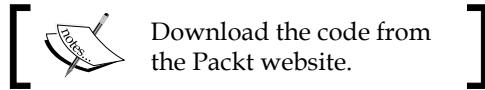
- **Step 2:** Create the header folder inside the components/common folder. Create a new file header.get.desc.xml in the header folder. Insert the following code:

```
<webscript>
    <shortname>Global Header Component</shortname>
    <description>Header component used across the whole
        application</description>
    <url>/component/common/header/header</url>
</webscript>
```
- **Step 3:** Create another new file, header.get.html.ftl, in the same location and insert the following code:

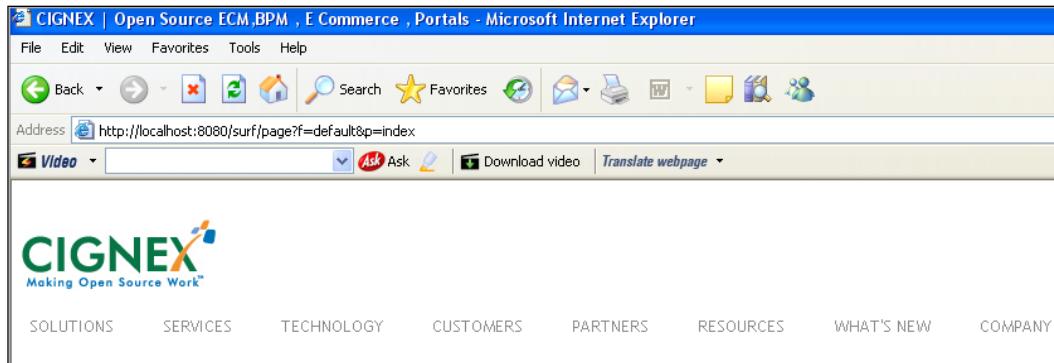
```
<meta http-equiv="Content-Type" content="text/html;
    charset=UTF-8" />
<meta name="description" content="Small Corporation" />
<meta name="keywords" content="small, corporation" />
<link rel="stylesheet" type="text/css" href="${app.context}/css/
style.css" media="screen" />
<link rel="stylesheet" type="text/css" href="${app.context}/css/
base.css" media="screen" />
<div id="site-header">
    <table cellpadding="0" cellspacing="0" border="0">
        <tr>
            <td align="left" valign="top" >
                
            </td>
        </tr>
    </table>
</div>

<ul id="menu">
    <li><a class="current" href="#">SOLUTIONS</a></li>
    <li><a href="#">SERVICES</a></li>
    <li><a href="#">TECHNOLOGY</a></li>
    <li><a href="#">CUSTOMERS</a></li>
    <li><a href="#">PARTNERS</a></li>
    <li><a href="#">RESOURCES</a></li>
    <li><a href="#">WHAT'S New </a></li>
    <li><a href="#">RESOURCES</a></li>
</ul>
```

Our component is now ready to be placed in the page. In this template we are using stylesheets and images. You can create stylesheets in the <install-application>/tomcat/webapps/surf/css folder. For images you have to navigate to the <install-application>/tomcat/webapps/surf/images folder.



- **Step 4:** Refresh the Web Script by entering the following URL:
`http://<server-name>:<port>/surf/service.`
- **Step 5:** Insert the highlighted code in `index.ftl` (that you have created earlier within the <install-application>/tomcat/webapps/surf\WEB-INF/classes/alfresco/templates folder within the <body> tag. Here we are using the region and scope defined in the component created earlier.
`<@region id="header" scope="global" />`
- **Step 6:** Test the component. You will find the header component is added in the home page. You should be able to test by entering the URL of your web or app server `http://<server-name>:<port>/surf/`.



Design page navigation

In the previous section, we learned about page creation and putting a component into a page. In this section, we shall learn about navigation from one page to another page.

For this, create one more page. Now we are going to link this page to the page we created earlier.

Create the new page. Create the new "template instance" so that the new template can be referenced from a page component. Create the new FTL template file.

- **Step 1:** Create news.xml in the <install-alfresco>/tomcat/webapps/surf/WEB-INF/classes/alfresco/site-data/pages folder. Insert the following code:

```
<?xml version='1.0' encoding='UTF-8'?>
<page>
  <id>news</id>
  <title>What's New</title>
  <description>Sample Cignex News page</description>
  <template-instance>news</template-instance>
  <authentication>none</authentication>
</page>
```

- **Step 2:** Create news.xml in the <install-alfresco>/tomcat/webapps/surf/WEB-INF/classes/alfresco/site-data/template-instances folder. Insert the following code:

```
<?xml version='1.0' encoding='UTF-8'?>
<template-instance>
  <template-type>news</template-type>
</template-instance>
```

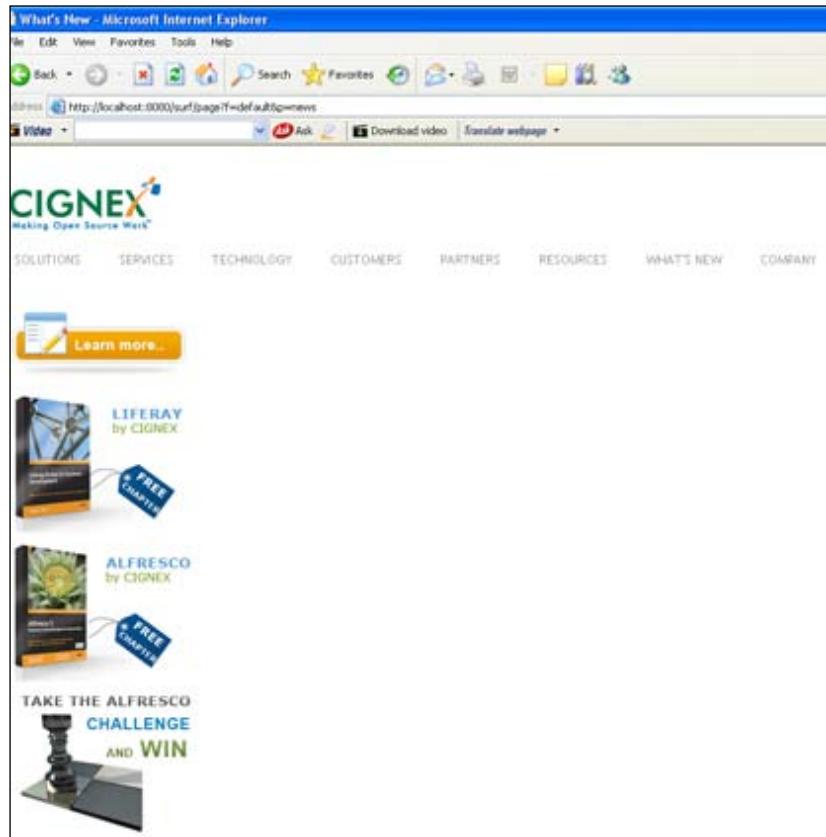
- **Step 3:** Create news.ftl in the <install-alfresco>/tomcat/webapps/surf/WEB-INF/classes/alfresco/templates folder. Insert the following code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>${page.title}</title>
    ${head}
  </head>
  <body>
    <@region id="header" scope="global" />

    <table width="100%" cellspacing="0" cellpadding="0" border="0">
      <tr>
        <td valign="top">
          
        </td>
      </tr>
      <tr>
        <td valign="top">
          
        </td>
      </tr>
    </table>
```

```
</td>
</tr>
<tr>
    <td valign="top">
        
    </td>
</tr>
<tr>
    <td valign="top">
        
    </td>
</tr>
</table>
</body>
</html>
```

By this time one more page is ready. Now we have two pages and one header component in place. We will see how one page can be linked to another page.



- **Step 4:** Go to the <install-alfresco>/tomcat/webapps/surf/WEB-INF/classes/alfresco/site-data/page-associations folder.

- **Step 5:** Create a new file index-news.xml. With this we are associating index page to news page and vice versa. Insert the following code:

```
<?xml version='1.0' encoding='UTF-8'?>
<page-association>
    <source-id>index</source-id>
    <dest-id>news</dest-id>
    <assoc-type>child</assoc-type>
    <order-id>1</order-id>
</page-association>
```

- **Step 6:** Go to the <install-alfresco>/tomcat/webapps/surf/WEB-INF/classes/alfresco/site-webscripts/components/common/header folder.

- **Step 7:** Create a new file header.get.js. Insert the following code:

```
// renderer attribute
var renderer = instance.properties["renderer"];
if(renderer == null)
{
    renderer = "horizontal";
}
model.renderer = renderer;

// set up rendering attributes
model.rootpage = sitedata.getRootPage();
model.linkbuilder = context.getLinkBuilder();
```

- **Step 8:** Modify the header.get.html.ftl file. Insert the following code.

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"
/>
<meta name="description" content="Small Corporation" />
<meta name="keywords" content="small, corporation" />
<link rel="stylesheet" type="text/css" href="${app.context}/css/
style.css" media="screen" />
<link rel="stylesheet" type="text/css" href="${app.context}/css/
base.css" media="screen" />
<table>
    <tr>
        <td>
            
        </td>
    </tr>
</table>
<#if renderer == "horizontal">
    <@horizontal page=rootpage showChildren=true/>
</#if>
```

```

<#macro horizontal page showChildren>
<#assign currentPageId = "">
<if context.pageId?exists>
    <#assign currentPageId = context.pageId>
</if>

<ul id="menu">
    <li>
        <#assign href = linkbuilder.page(page.id, context.formatId)>
        <#assign classId = ''>
        <if page.id == currentPageId>
            <#assign classId = 'current'>
        </if>
        <a href='${href}' id='${classId}'>SOLUTIONS</a>
    </li>

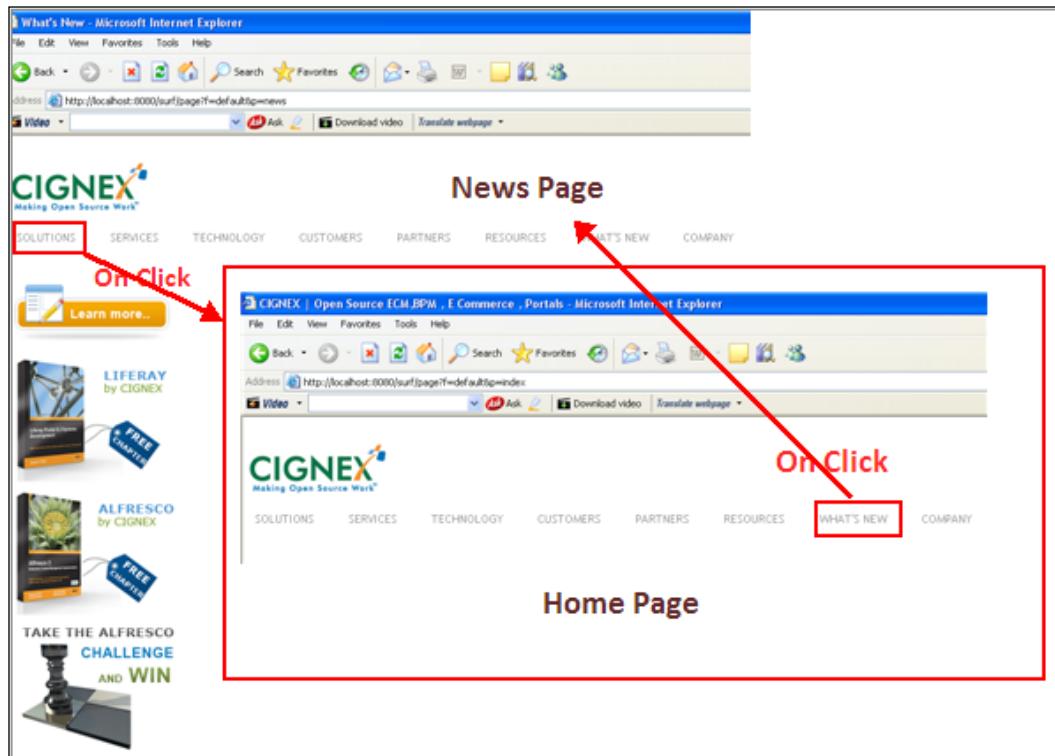
    <li>
        <#assign href = linkbuilder.page(page.id, context.formatId)>
        <a href='${href}' id='current'>SERVICES</a>
    </li>
    <li>
        <#assign href = linkbuilder.page(page.id, context.formatId)>
        <a href='${href}' id='current'>TECHNOLOGY</a>
    </li>
    <li>
        <#assign href = linkbuilder.page(page.id, context.formatId)>
        <a href='${href}' id='current'>CUSTOMERS</a>
    </li>
    <li>
        <#assign href = linkbuilder.page(page.id, context.formatId)>
        <a href='${href}' id='current'>PARTNERS</a>
    </li>
    <li>
        <#assign href = linkbuilder.page(page.id, context.formatId)>
        <a href='${href}' id='current'>RESOURCES</a>
    </li>
<!-- Children of Home Page -->
<#list sitedata.findChildPages(page.id) as parentPage>
    <li>
        <#assign href = linkbuilder.page(parentPage.id,
            context.formatId)>
        <#assign classId = ''>
        <if parentPage.id == currentPageId>
            <#assign classId = 'current'>
        </if>
        <a href='${href}' id='${classId}'>${parentPage.title}</a>
    </li>
</#list>

```

```
<li>
  <#assign href = linkbuilder.page(page.id, context.formatId) >
  <a href='${href}' id='current'>COMPANY</a>
</li>

</ul>
</#macro>
```

- **Step 9:** Refresh the Web Script by entering the URL
`http://<server-name>:<port>/surf/service.`



So far we have learned about the creation of a single-tier application. Now if we want to create a two-tier application, we shall create one more web script, which will be calling web scripts that are defined in Alfresco WCM.

Communicating with Web Content Management

Alfresco Surf works hand-in-hand with Alfresco Web Content Management and provides virtualized content retrieval, preview, and test support for User Sandboxes and web projects. Applications built with Alfresco Surf can be deployed from Alfresco Web Project spaces to production servers while taking full advantage of Alfresco WCM's Enterprise class features. These include:

- **Safe editing of all aspects of Surf-powered sites:** Snapshots allow your Surf site to roll backward and forward in time. Audit trails provide a rich log of changes made to the site.
- **Integrated workflow for approval of all Surf components:** These include pages, templates, components, chrome, and much more.

User Sandboxes provide freedom for experimentation and iteration in design. Change anything about your Surf site with the assurance that you can always discard changes or promote the bits you are happy with.

The following example is for integrating WCM web scripts with the Surf application. This Web Script is actually located on the remote Alfresco server and therefore we have to configure Surf to search the location for web scripts.

- **Step 1:** Go to the <install-alfresco>/tomcat/webapps/surf/WEB-INF/classes/alfresco folder.
- **Step 2:** Modify the web-framework-config.xml file. Insert the following code before </web-framework>. The highlighted text *cignex* is the DNS name of the web project. This lets Surf bind to web application resources of the "cignex" web project.

```

<resource-resolver>
  <id>webapp</id>
  <name>Alfresco Web Application Resource Resolver</name>
  <description>
    Resolves data access for web application assets
  </description>
  <class>
    org.alfresco.web.framework.resource.
    AlfrescoWebProjectResourceResolver
  </class>
  <alias-uri>/files</alias-uri>
  <store-id>cignex</store-id>
</resource-resolver>
```

- **Step 3:** Modify the file `webscript-framework-config.xml`. Uncomment the following code that is placed in the file. This configures the remote Alfresco repository as an endpoint.

```
<endpoint>
  <id>alfresco-system</id>
  <name>Alfresco - System access</name>
  <description>System account access to Alfresco</description>
  <connector-id>alfresco</connector-id>
  <endpoint-url>http://localhost:8080/alfresco/s
  </endpoint-url>
  <identity>declared</identity>
  <username>admin</username>
  <password>admin</password>
  <unsecure>true</unsecure>
</endpoint>
```

The rest of the case study is covered in *Chapter 10*, in the *Alfresco WCM –Surf-based Web application integration* section.

Now we shall create one more web script that will call other web scripts that are stored in Alfresco WCM to fetch data.

- **Step 1:** Create a news folder inside the `site-webscripts/components/common` folder. Create a new file `news.get.desc.xml` in the news folder. Insert the following code:

```
<webscript>
  <shortname>News</shortname>
  <description>Calls the remote Alfresco web script</description>
  <url>/component/common/news/news</url>
</webscript>
```

- **Step 2:** Create a template file, `news.get.html.ftl`, in the same location and insert the following code:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"
/>
<meta name="description" content="Small Corporation" />
<meta name="keywords" content="small, corporation" />
<link rel="stylesheet" type="text/css" href="${app.context}/css/
style.css" media="screen" />
<link rel="stylesheet" type="text/css" href="${app.context}/css/
base.css" media="screen" />
<table width="100%" cellspacing="0" cellpadding="0" border="0">
<tr>
  <td valign="top">
```

```
<#list news as n>
    ${n.Headline}
    <br>
</#list>
</td>
</tr>
</table>
```

- **Step 3:** Create a JavaScript file, `news.get.js`, in the same location and insert the following code:

```
{
    // get a connector to the Alfresco repository
    var connector = remote.connect("alfresco-system");
    var jsonFeed = connector.get("http://localhost:8080/alfresco/
service/org/cignex/news/getNewsHeadlines.json?storeId=cignex");
    var obj = eval('(' + jsonFeed + ')');
    if (obj)
    {
        model.news = obj["newsItems"];
    }
}
```

Using YUI (Yahoo User Interface) library

YUI is a set of utilities and controls, written with JavaScript and CSS, for building richly interactive web applications. It is an open source JavaScript library using techniques such as DOM scripting, DHTML, and AJAX. YUI consists of several CSS components and nearly three dozen JavaScript components, all of which are designed to empower the rapid creation of web applications that run in all of the major web browsers. It provides a rich set of widgets and utilities, such as animation, drag-and-drop, image loader, menu, paginator, uploader, and many others. Mainly, it has six components:

- **YUI Core:** It is a light set of tools for event management. It consists of Yahoo Global objects within which all of the YUI library code resides. It also contains an Event utility that provides developers with easy and safe access to browser events such as mouse clicks, key press, and so on.
- **YUI Utilities:** It provides various utilities like animation, data source, storage, SWF, JSON, image loader, drag-and-drop, and many more.
- **YUI Widgets:** It provides a variety of easy-to-use controls, such as Data Table, menu, paginator, progress bar, uploader, image cropper, and many more.

- **CSS Resources:** It consists of CSS Reset (removes inconsistent styling of HTML elements provided by browsers), CSS fonts (standardized cross-browser font families and size rendering), and CSS base (an optional CSS file that complements YUI's core CSS foundation).
- **Development tools:** It contains Logger Control (provides a quick and easy way to write log messages to an onscreen console), Profiler (allows you to specify exactly the parts of your application to the profile), ProfilerViewer control (used in combination with Profiler to provide rich visualizations of your profiling data), and YUI test utility (testing framework for browser-based JavaScript solutions).
- **Build Tool:** YUI Compressor is a build-process component written in Java that minifies your JavaScript and CSS. It is a JavaScript and CSS minifier designed to be 100 percent safe and yield a higher compression ratio than most other tools.

As we discussed earlier, we have a set of predefined controls that we can use by instantiating an object and calling a set of methods on them.

For instance, use the horizontal menu plugin provided by the YUI library and observe how easy it is to make a rich menu.

Modify the file `header.get.html.ftl` located at `site-webscripts/components/common/header`. Insert the downloaded code from the Packt website. You will notice a change in the menu style.



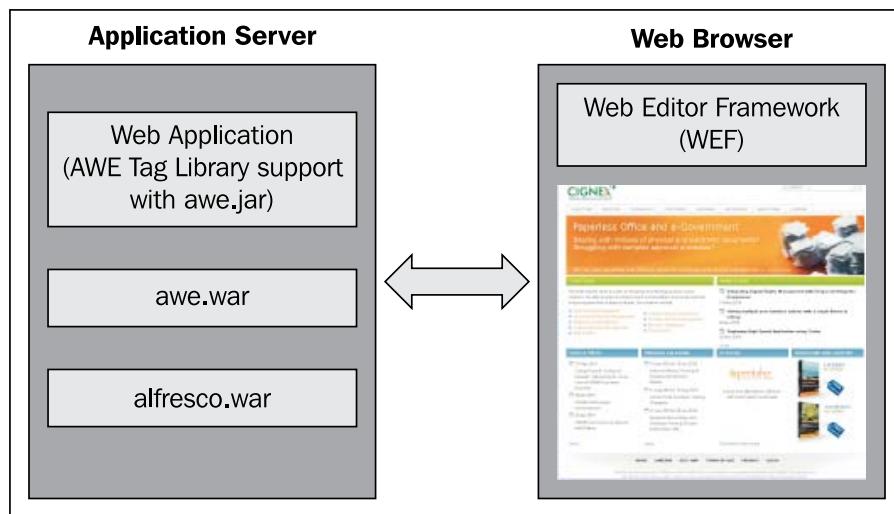
[ Download the code samples
from the Packt website.]

Alfresco Web Editor

Alfresco Web Editor is an application developed using the Spring Surf platform. It provides in-context editing capabilities for Alfresco repository content. Because of the in-context editing capability of the Web Editor, it's very easy for any non-technical person to edit Alfresco content directly from the web page. It supports Presentation editing for Surf-based websites.

Alfresco Web Editor is a Surf application incorporating the Alfresco Form Engine. It uses the Form Service default template.

Alfresco Web Editor is packaged as a standalone WAR file (`awe.war`). Hence you can deploy it as a standalone application on the same server where Alfresco is running or also can deploy it as a part of any Surf Application remote to the Alfresco server.



Deploying and using Alfresco Web Editor

First you need to download the Alfresco Web Editor. A file `alfresco-enterprise-webeditor-3.3.zip` will consist of the `awe.war` file and other required files. To deploy AWE, follow the steps mentioned below:

1. Browse to <https://network.alfresco.com> and download the `alfresco-enterprise-webeditor-3.3.zip` file. Refer to the Alfresco wiki to download the community version: http://wiki.alfresco.com/wiki/Alfresco_Community_Edition_3.3.

2. Stop your Alfresco server if it's already running.
3. Deploy the `awe.war` file (which is there in the downloaded ZIP file) into the same application server where Alfresco is running.
4. Restart your Alfresco server.

Once you start the Alfresco server, you can navigate to `http://localhost:8080/awe`. You will see the login page; provide the same username and password you are using for Alfresco (`admin/admin`).

The first page you will see will be the metadata page; here you can pass any Alfresco content noderef as `itemId` parameter and can update the metadata of that content as shown in the following screenshot:

The screenshot shows a web browser window titled "Alfresco Web Editor > Metadata". The URL in the address bar is `http://localhost:8080/awe/page/metadata?itemId=avm://wwwcignex--admin/-1;www;avm_webapps;ROOT;common;inc;CignexNews.xml`. The page displays a form with the following fields and their values:

* Required Fields	
Name: *	CignexNews.xml
Node DB Identifier: *	336
Last Accessed Date:	DD/MM/YYYY HH:MM (24 Hour) 30/5/2010 23:56
Store Identifier: *	wwwcignex-admin
Node Identifier: *	UNKNOWN
Modified Date: *	DD/MM/YYYY HH:MM (24 Hour) 30/5/2010 23:56
Created Date: *	DD/MM/YYYY HH:MM (24 Hour) 30/5/2010 23:56
Store Protocol: *	avm

Deploying Web Editor to a Spring Surf Application

As mentioned in the introduction of the Alfresco Web Editor, we can also deploy the Web Editor to any Surf Application, which is remote to the Alfresco server.

In this section, we will discuss how we can deploy and what are the extra configurations we need to do for using it in a Surf-based application. The Alfresco Web Editor distribution contains all of the required files for configuring it for the Surf-based application.

1. Copy the following files to the WEB-INF/lib directory:

- yui-2.7.0.jar
- spring-webeditor-1.0.0.CI-SNAPSHOT.jar
- alfresco-forms-client-3.3.jar
- alfresco-webeditor-plugin-3.3.jar

2. Configuring the Tag Library:

Alfresco Web Editor provides tag library. If you intend to use the tags provided in this library, you need to copy the alfresco-webeditor-taglib-3.3.jar file in the WEB-INF/lib folder.

This tag library mainly includes three tags: startTemplate, markContent, and endTemplate. We will discuss about these tags later in this chapter.

3. Configuring the Servlet Filter:

If you are using the previously mentioned tags, startTemplate, markContent, and endTemplate will only render the output if you have configured the Web Editor filter in web.xml as follows:

```
<filter>
<filter-name>Alfresco Web Editor Filter</filter-name>
<description>Enables support for the Alfresco Web Editor</description>
<filter-class>org.alfresco.web.awe.filter.WebEditorFilter</filter-class>
</filter>
<filter-mapping>
<filter-name>Alfresco Web Editor Filter</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

You can also set the following two optional parameters to control the contextPath when the URLs to the Web Editor are generated and for the debug mode:

```
<init-param>
<param-name>contextPath</param-name>
<param-value>/quickstart</param-value>
</init-param>
<init-param>
<param-name>debug</param-name>
<param-value>true</param-value>
</init-param>
```

4. Configuring Web Editor Forms:

The Alfresco Web Editor uses Forms to edit the content and the metadata. By default, it displays the title, description, and content fields. An alternative form can be used by providing a markContent tag with the formId attribute.

The node type is presumed to be cm:content. If you have custom types or wish to specify other properties, you can use the form's configuration techniques.

Alfresco Web Editor tag library

The Alfresco Web Editor tag library consists of three tags: StartTemplate, markContent, and endTemplate.

- startTemplate tag

Purpose:

This tag bootstraps the Web Editor Framework using a script element that executes a Web Script. Place this tag in the head section of the page.

Attributes:

toolbarLocation

This attribute specifies the location of the toolbar.

<awe:StartTemplate toolbarLocation="top" /> specifies the location as top in the window for the toolbar.

The possible values are: top, left, and right. This is an optional attribute; the default is top.

- `markContent` tag

Purpose:

This tag indicates the editable area on the page. This renders an Edit icon; when clicked, it displays a form to edit the content or metadata or both.

Attributes:

`id`: This attribute specifies the noderef of the Alfresco node. This is a mandatory attribute.

`title`: This attribute specifies a title for the marked editable area. The same title will be used in quick edit drop-down, form edit pop-up dialog, and so on.

This is a mandatory attribute.

`formId`: This attribute specifies which form will be used when the marked area is edited. This is an optional attribute.

`nestedMarker`: This attribute defines whether the editable area is nested within another HTML tag that represents the content being edited. If set to "true" the whole parent element is highlighted when the area is selected in the quick edit drop-down menu. If set to "false", only the edit icon is highlighted.

- `endTemplate` tag

Purpose:

This tag initializes the Web Editor with details of all of the marked content areas on the page. It also renders a script element that executes the WEF resources web script, which starts the process of downloading all of the assets required to render and display the toolbar and all configured plugins. Place this tag just before the closing body element

Attributes:

This tag doesn't have any attribute.

Sample Web Application using Alfresco Web Editor

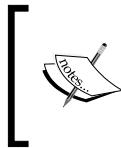
When you download the Web Editor distribution, you will find one sample application, `customer.war`. This is a simple JSP-based application, which uses the tag library provided by AWE.

In this application, we will use the already created content on our custom JSP and will allow users to have an in-context editing facility for that content.

For example, you can create content in Alfresco. Go to **Company Home** and a specific space where you want to create the content. Use the **Add content** functionality to upload any file. Here in this example, I have uploaded one HTML file called `Cignex_Merge_News.html` to the Alfresco repository.

In the `noderefs.jsp` file located at `customer/includes`, provide the noderef of the previously created content from the Alfresco Repository as shown next (it is the modified content of `noderefs.jsp`):

```
<%
String mainTextNodeRef = "workspace://SpacesStore/3b12b9a8-8f9f-414d-
8a9d-40047822d1cf";
%>
```



Note that `workspace://SpacesStore/3b12b9a8-8f9f-414d-8a9d-40047822d1cf` is the noderef of Alfresco content `Cignex_Merge_News.html`. To find the nodeRef of your content, go to the **View Details** page of the content and click on **Alfresco Node Reference**.



Also update the `body.jsp` located at `customer/includes` accordingly.

Once you specify the noderefs as earlier and click on `http://localhost:8080/customer`, you can see the screen shown in the following screenshot:

The screenshot shows a web browser window with the Alfresco logo in the top left. The title bar reads "CIGNEX AGS merger announcement". The page content includes a header with the date "News date: 26 Apr 2010". Below the header is a section titled "CIGNEX Holding Corporation: A New Synergistic Force in Enterprise Open Source Solutions". The text discusses the merger between CIGNEX Technologies, Inc. and AGS Technology Group, mentioning their global leadership in Open Source consulting and various business solutions. It highlights the merged entity's focus on scaling and providing robust services. The footer contains contact information for Dave Jennings, including a phone number and email address.

This application has custom tags to display the property and content of the specified noderef.

To display any property of the noderef, use a property tag like:

```
<customer:property nodeRef="<%mainTextNodeRef%>"  
property="cm:description" />
```

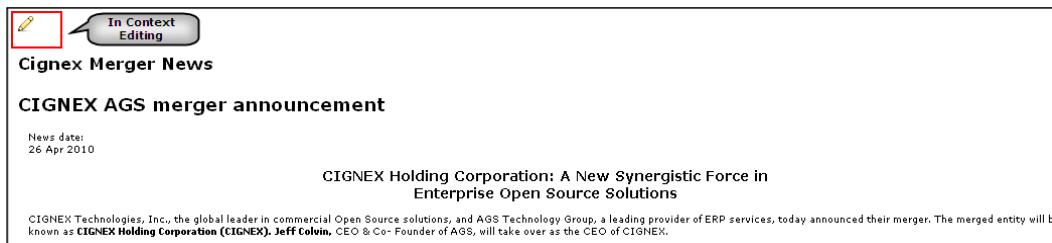
To display the content of the noderef, use a content tag like:

```
<customer:content nodeRef="<%mainTextNodeRef%>" />
```

As discussed earlier, the startTemplate tag will have a toolbarLocation attribute that specifies the location of the toolbar (the default is "top"). The following screenshot shows the toolbar on the page:

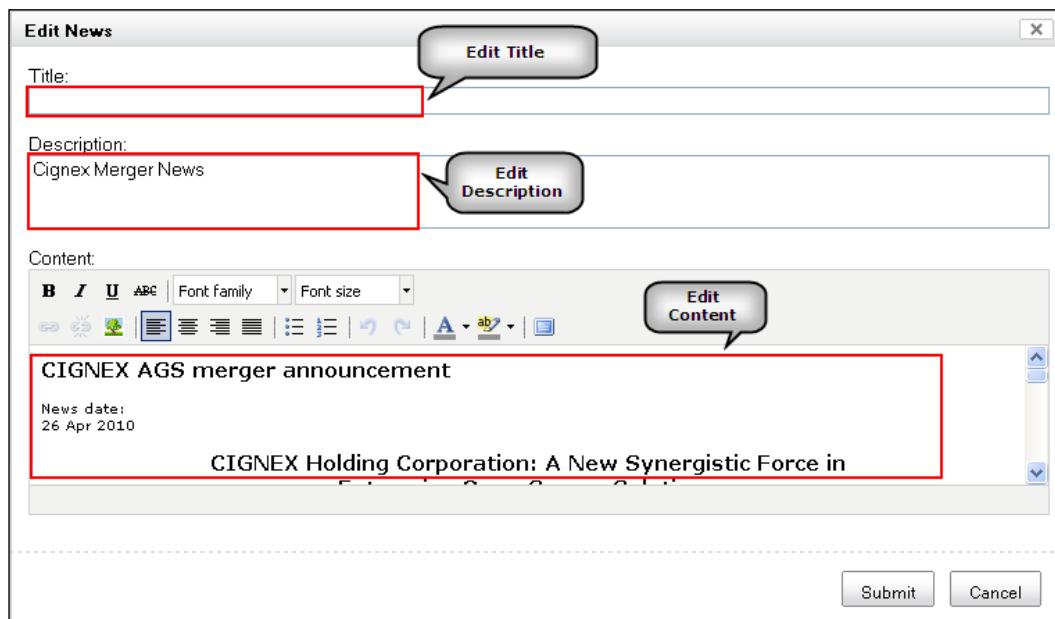


You can directly click on the link available in the drop-down here in the toolbar. Another option is the **Edit** icon that is also available with the content. The following screenshot shows how you can use the in-context editing feature to edit the content.



When you click on the **Edit** icon, it will open the window in which you can edit the title, description, or content. This window is using Alfresco's form service. You can specify which form ID you want to use for in-context editing. As described in the earlier section, with the `markContent` tag, you can specify the `formId` attribute, which specifies the form to use for in-context editing. For example:

```
<awe:markContent id="<%=<<NODEREF>>%>" formId="description"
title="Edit News Description" nestedMarker="true" />
```



Download the `customer.war` with modified files from the Packt website.

Web Editor Framework

Web Editor Framework consists of core components and UI widgets. The framework allows you to easily develop and add the plugin. This is achievable because of the nature of the framework, which is supplied by the support of hooks around core methods. The web editor is an event-driven framework. Every core method of a plugin fires a before and an after event for that method.

Core WEF Components

The Core WEF Components are as follows:

- **WEF:** The WEF object is responsible for bootstrapping the Web Editor framework onto the page.
- **Loader:** The loader is responsible for setting up resources for modules (and any of their dependencies) to be loaded by the browser.
- **ConfigRegistry:** The ConfigRegistry object is responsible for maintaining a registry of configuration objects for a specified plugin.
- **PluginRegistry:** The PluginRegistry maintains a registry of plugins.
- **Base:** The WEF.Base is the core class that all WEF components are based on.
- **Plugin:** The Plugin component builds on Base and every plugin is expected to be the one that extends Plugin or one of its subclasses.
- **Widget:** A widget is a UI element that can be interacted with. A dialog is a good example of a widget.
- **App:** An App is just a plugin and adds no additional methods of its own.

Core WEF Widgets

The Core WEF Widgets are as follows:

- **Ribbon:** The Ribbon component is responsible for displaying the ribbon at the top of the page and also for managing and/or registered toolbars and plugins.
- **Toolbar:** The Toolbar component is a simple component that manages a collection of buttons.
- **Tabbed Toolbar:** The Tabbed Toolbar manages a collection of toolbars that are rendered in different tabs.



For more details on WEF, refer to the link: http://wiki.alfresco.com/wiki/Web_Editor_Framework.



Summary

The Alfresco Surf platform and tools are designed to help users create the next-generation WCM platform that covers both core management aspects and delivery capabilities. The Alfresco Web Editor is a Surf-based application incorporating the Alfresco Form Engine, which facilitates in-context editing of Alfresco repository content.

10

Integrating WCM Using Web Scripts

In this chapter, we will learn about web scripts and integration of WCM with external systems using web scripts. In previous chapters, you got an idea about why to use WCM, how to create web projects, and how to produce content in Alfresco WCM along with deployments to a different server. So far we discussed how we use Alfresco as a content production and delivery system and for serving the content. You may have your own application and may want to use Alfresco purely for web content management, which will just produce the content in Alfresco. But the presentation part of the application will not be handled by Alfresco. Here you can have your frontend application in any technology and you can use Alfresco WCM as the backend, which will serve the content to your application. Web script's RESTful service is one of the most easily available and widely used solutions to integrate the Alfresco WCM with any external system or application.

By the end of this chapter, you will have learned about:

- REST architecture
- Web script framework in Alfresco
- Components of web scripts
- WCM-specific JavaScript and template APIs
- Implementing web scripts in Alfresco
- Using web scripts to integrate Alfresco WCM with any external system or application

Concepts of WCM web scripts

In this section, you will get an overview of web scripts in WCM. We will talk about the REST architecture and web script framework in Alfresco. We will also talk about the new features introduced for web scripts in Alfresco 3. At the end of this section, you will have a very good idea about the architectural benefits of using RESTful web services, how it helps in providing solutions with the Alfresco WCM, and also how easy it is to integrate WCM with an external system.

Overview of REST architecture

In this section, we will discuss about the REST architecture and RESTful web services. You will also get an idea about how RESTful web services are different than our traditional SOAP-based web services and the advantages of using this approach over a traditional SOAP-based approach.

What is REST

REST (Representational State Transfer) is used to describe a Web Oriented Architecture rather than a Service-Oriented Architecture. It is an architectural style following the principles referred to as "RESTful". RESTful web services are resource-oriented services. You can identify and locate resources by a **Uniform Resource Identifier (URI)**. By making everything URL (URI)-based, you can access everything via that URL and use standard web commands such as Get and Post to access and update those things, something we call resources.

REST's main principles

- Resource-oriented states and functionalities that are abstracted into resources
- Unique way of identifying resources
- The idea of how operations on these resources are defined in terms of a single protocol for interacting with resources
- A protocol that is client-server, stateless, cacheable, and layered
- REST-oriented system design leads to systems that are open, scalable, extensible, and easy to understand

This URL-based update of information contrasts with a very Remote Procedure Call-like interface of SOAP. The following points give an idea about how RESTful web services differ from SOAP-based web services:

- Design approach: REST focuses on resources, whereas web services focus on messages, which are operations.
- Less dependencies as there is no SOAP required.
- Cacheable representation leads to improved response time and reduced server load.
- No need for a separate resource directory due to the use of hyperlinks in representations.
- Improves server scalability by reducing the need to maintain session state. This means that different servers can be used to handle different requests in a session.

Alfresco web scripts overview

Alfresco provides a web script framework based on REST architecture, which was described in the earlier section. RESTful web scripts provide a very suitable and flexible solution to integrate Alfresco with any other application, and is usually the best solution out of all of the available options. With the help of web scripts, you can easily access the Alfresco repository and perform operations such as content searching, retrieval, manipulation, and so on with the help of services available in Alfresco, even from outside Alfresco.

What is a web script

A REST web script is simply a service bound to a URI and based on HTTP. So the technology that the external application is implemented on is irrelevant, that is, they are cross platform and cross language. You are not locked into any programming language or development environment.

The web script framework lets you roll your own APIs, thereby allowing you to fine-tune the remote APIs that you expose to the external application. REST provides a convenient bridge between any native application and the content management. This allows you to place controls on your enterprise content to manage it and at the same time provide uniform access for a wide variety of client applications and services, such as browser, portal, search engine, or custom application.

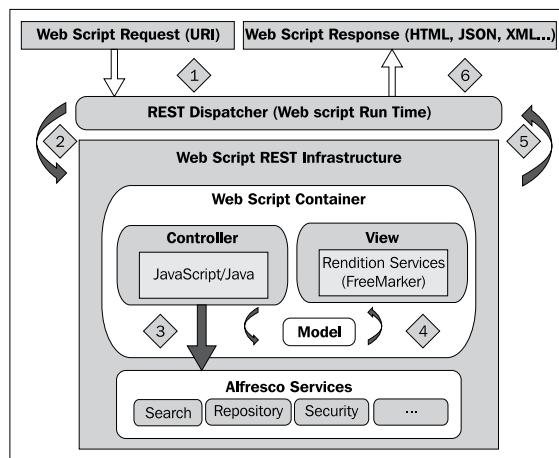
Why to use web scripts

The advantages of using a web script for integrating external applications with Alfresco WCM are:

- Build custom URI-identified and HTTP-accessible content management web services
- Turn your Alfresco repository into a content management-powered HTTP server
- Instead of having a single, monolithic system responsible for all aspects of a content-centric web application, this provides a design of loosely coupled sub-systems that are being integrated to create more agile solutions
- This approach allows your content management system (Alfresco WCM) to have a flexible and lightweight interface
- Provides the flexibility of not being locked into a presentation approach based on the content repository
- Uses lightweight scripting technologies such as JavaScript and FreeMarker

Alfresco web script framework

Alfresco's web script framework follows the **Model-View-Controller (MVC)** pattern, which makes a developer's life easy. The Model-View-Controller design pattern decouples data access, business logic, and data presentation. By applying the MVC pattern, you can separate core business model functionality from the presentation and control logic that uses this functionality. Such separation allows multiple views to share the same enterprise data model, which makes supporting multiple clients easier to implement, test, and maintain.



As shown in the previous figure, the **Controller** here is the server-side JavaScript. It can also be JavaBeans or a combination of both. The request first comes to the **REST Dispatcher**, and **Web script Run Time** forwards this to the **Controller**, which is responsible for processing the request. It performs the required business logic with the help of Alfresco core services and populates the model with data, and then forwards the request to the **View**. A **View** is one or more FreeMarker templates and is responsible for generating the response in the proper format (XML, HTML, JSON, and so on) as a presentation layer. Finally, the **Web script Run time** will send the response back to the client.

- **Controller:** JavaScript/JavaBeans
- **View:** One or more FreeMarker templates
- **Model:** Data structure, used to pass data between View and Controller

What's new in Alfresco 3 web scripts

Alfresco has introduced web scripts framework in older versions such as 2.1. With new releases of Alfresco, efforts are being made to enhance the capabilities of web scripts. Following are some of the enhancements for web scripts available from Alfresco 3 onwards:

- Standalone component

With this new Alfresco version 3, the web script architecture is refactored in such a way that you need not use the Alfresco repository server to host, but you can host it in any environment and even in tiers. This will allow web scripts to be used in a presentation layer to render UI. For example, the Alfresco Surf platform hosts the web script framework for reusable UI components.

- URI templates comply with JSR-311 (JAX-RS)

A URI template is simply a URI containing tokens that may be substituted with actual values. From Alfresco 3.0, this template syntax complies with the template syntax of JSR-311 (JAX-RS).

- Run as user

You can use the `runas` attribute with authentication in the description document for a web script. This will allow declaring an alternative username to run the web script, as this is valid for class path-stored web scripts only.

- Ability to customize web scripts with web script-specific configuration

Configuration is accessed via the `config` root object, which is available during both controller script and template execution. So, with the help of this, you can have web script-specific configuration, which can then be used in a controller or rendering template.

- Per web script message bundle

To localize the web script response, with Alfresco 3.0 and later, you can have a message bundle for each web script.

- Improved form processing

When posting a request of mime type `multipart/form-data`, `formdata` is available to the controller script as a root object. This root object allows a web script to read all fields. These multi-part form fields are also mapped to `args`/`argsM`.

- Access to request headers

With this version, Alfresco provides one more JavaScript root object named `headers`, which is an associative array of all request headers. In this way, you can access the request header as well from the web script controller script.

- Request body processing

The root object `requestbody`, which represents the content of a request body, is available with Alfresco 3.

- Content negotiation

You can have a `negotiate` element in a description document, which associates an `Accept` header mime type to a specific web script format of response; the value (mandatory) specifies the format while the `accept` attribute (mandatory) specifies the mime type. You can have zero or more negotiation elements, but to enable content negotiation you need the definition of at least one `negotiate` element.

- Declarative and programmatic cache control

You can have a `cache` element specified in the description document (declarative) for caching and set some child elements to configure the cache. Also, you can have a `cache` root object available in the controller, which helps to control the cache runtime (programmatic).

You can override some definition time controls and also set expiry criteria with this root level object `cache`.

- Web script families

If you want to categorize similar or related web scripts in groups, you can configure a `family` element in the description document.

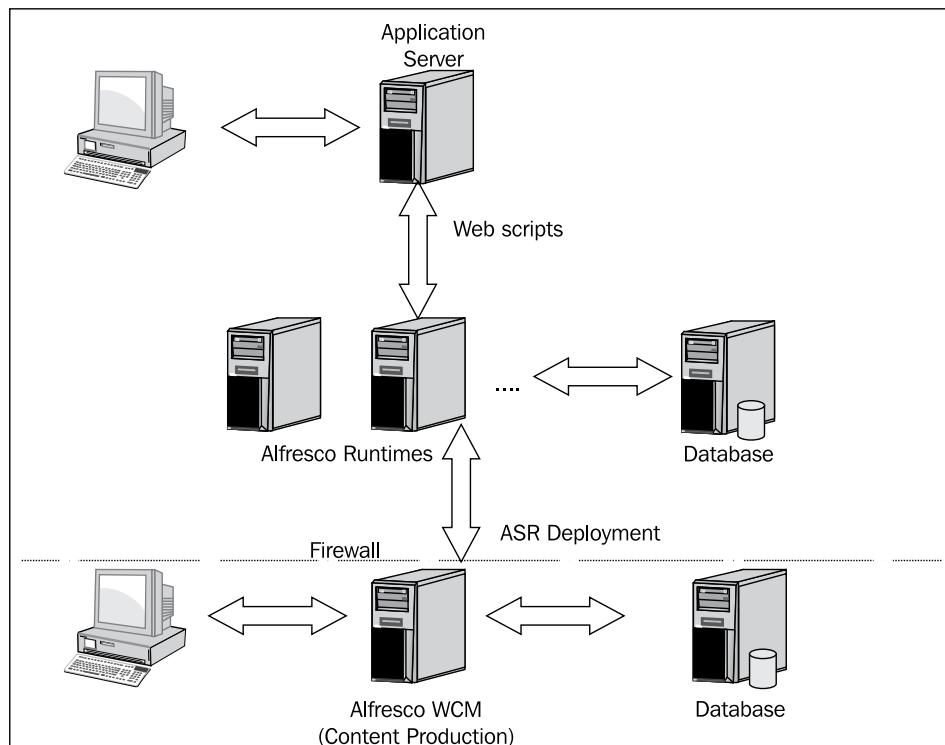
The `family` tag may be repeated if the script belongs to more than one family.

- Specialized kinds of web scripts

A web script "kind" allows a Java-backed implementation to be named, and for any web script description document to state if it wants to use that implementation instead of the out-of-the-box one provided by Alfresco.

Using web scripts with Alfresco WCM

As described in *Chapter 1, A Publishing Style Web CMS*, Alfresco WCM is used for content production and content delivery. Alfresco supports two main models for content delivery – static and dynamic. With the dynamic delivery model, content is served from Alfresco; however, the rendering part will not be handled by it, but by some other application. This provides flexibility in deciding what content to display and how it will be displayed on a page. Basically, a web script will act as a communication channel to fetch the content, which is being stored in Alfresco. With the dynamic content delivery model, content will be published to an Alfresco Runtime (as shown in the following figure) with ASR deployment, thereby making the content available for dynamic queries by basically any web technology (J2EE, PHP, Python, .NET, AJAX, Flash, Cold Fusion, and so on). Here the web script is a remote API used to query and retrieve the content from Alfresco Runtime:



In this way, you can use Alfresco purely for web content management, which will be responsible for just producing the content, and then any application outside Alfresco will be responsible for the actual page composition. Hence, you will have the flexibility to separate content production and content rendering/presentation. This allows us to easily integrate Alfresco with any other application. Later in this chapter, we will see integration with Liferay and Drupal, which will give you an overview of how we can use Alfresco as a content production environment. Then we will use that content in any application outside Alfresco.

Implementing web scripts for WCM

Now after getting a good overview of web scripts, REST architecture, and the Alfresco web script framework, it's time to implement the web script for Alfresco WCM. In this section, we will discuss how to create a web script and the required components of web scripts in detail.

Components of web scripts

Before creating any web script, it is necessary to know about the different components of a web script. So now we are going to talk about them.

Description document

Description document is the entry point and required component for a web script. Here you have to specify the URI of a web script, which will initiate that web script. Along with the URI of the web script, a description document describes the short name, description, authentication, and transactional details. The description document provides Alfresco with all of the necessary information to bind the web script to one or more URLs.

Controller script

Controller script is generally used if you want to perform some business logic such as querying the database and fetching some content or updating the repository with some data or content. You can generate a data model within the controller, which will be used by one or more response templates. This is an optional component of a web script.

One or more response templates

Response template is basically the view part of a web script MVC framework, which is responsible for rendering the output in the proper format. You can specify more than one response format in the description document and create those templates for different formats. You can use the model generated by the controller in response templates. This is a mandatory component of a web script. You should have at least one response template for each web script.

Configuration document

There are different kinds of **configurations** defined in the XML document, which can be accessed via the config root object. This config object is available during both controller script and response template. This is an optional component of a web script.

Locale message bundle

From Alfresco version 3, a web script supports the localization of responses of a script. Each web script can have separate **message bundles** to store locale-specific messages.

Creating a description document

As stated earlier, a description document is the entry point for a web script, which initiates the web script. This is the mandatory component for a web script. You need to follow the naming convention for a web script description document, which is as follows:

```
<serviceId>.<httpMethod>.desc.xml
```

where serviceId is the web script ID and httpMethod can be GET, POST, PUT, or DELETE.

For example, if we have a web script that is responsible for getting the news headlines, we can have the filename as `getNewsHeadline.get.desc.xml`.

The content of this description document can be:

```
<webscript>
  <shortname>Listing Of News Headlines</shortname>
  <description>Listing Of News Headlines for Cignex home page thru
  Webscript</description>
  <url>/org/cignex/news/getNewsHeadlines.xml?storeId={storeId}</url>
  <authentication>user</authentication>
</webscript>
```

Basic elements of description document

- `shortname`: Human readable name for the web script and a mandatory element.
- `description`: Documentation for the web script. This is an optional element.
- `url`: A URI template to which the web script is bound. There can be one or more variants of the URI template for different formats.
- `format`: Decides how the required response type is stated with the URI. This is an optional element. Possible values (optional) are as follows:
 - `argument`: Describes the required response type, which can be declared via the `format` URI argument. For example, `/news?format=json`.
 - `extension`: Specifies that the required response type can be declared via the URI extension. For example, `/news.json`.
 - `any`: Specifies that either of the previous two is supported.

If not specified, the default value for `format` is `any`. And if the URI doesn't contain any response type, a default content type is taken from the default attribute of the `format` element, which is optional. And, if not specified there as well, the default `format` is `html`.

- `authentication`: The required level of authentication. This element is optional. If not specified, the default authentication is `none`. Valid values are as follows:
 - `none`: Specifies that no authentication is required at all
 - `guest`: Specifies that at least guest authentication is required
 - `user`: Specifies that at least named user authentication is required
 - `admin`: Specifies that at least a named admin authentication is required

The optional `runas` attribute can be used to declare an alternative username to run the web script as. This is valid for classpath-stored web scripts only.

Advanced configuration for a description document

Following are the elements used for advanced configuration of a description document. All of them are optional elements:

- `transaction`: Specifies the required level of transaction. This is an optional element. The default value is `none`. The valid values are:
 - `none`: Specifies that no transaction is required at all

- required: Specifies that a transaction is required and will inherit an existing transaction, if any is open
- requiresnew: Specifies that a new transaction is required; this will not inherit an existing transaction, if any is open
- family: Used to specify a tag for categorizing similar or related web scripts. You can have more than one family tag if this script belongs to more than one family.
- cache: Specifies the required level of caching. Available child elements are:
 - never: Specifies whether caching should be applied at all; default option is true. Valid values are:
 - true: Specifies that the web script response should never be cached
 - false: Specifies that the web script response may be cachedIf never is not specified, the default is true.
 - public: Specifies whether authenticated responses should be cached in a public cache; default is true. If public is not specified, then the default is false. Valid options for the public element are:
 - true: Specifies that the web script authenticated response may be cached in a public cache
 - false: Specifies that the web script authenticated response may not be cached in a public cache
 - mustrevalidate: Specifies whether a cache must revalidate its version of the web script response in order to ensure freshness. The default value for this is true. Available valid values are:
 - true: Specifies that validation must occur
 - false: Specifies that validation may occur
- negotiate (zero or more): Associates an Accept header MIME type to a specific web script format of response; the value (mandatory) specifies the format and the mandatory accept attribute specifies the MIME type when content negotiation is enabled with the definition of at least one negotiate element.
- kind: An attribute discriminator for overriding the kind of web script. See the list of available web script kinds. An attribute kind allows a Java-backed implementation to be named, and for any web script description document to state, it wants to use that implementation instead of the out-of-the-box one provided by Alfresco.

- `lifecycle`: An attribute for describing the lifecycle of the web script. Web scripts typically start out in some sort of a draft state, are promoted to full production use, and are then retired at the end of their life. The possible values for this attribute are as follows:
 - `none`: States this web script is not part of a lifecycle.
 - `sample`: Indicates this web script is a sample and is not intended for production use.
 - `draft`: Indicates this method may be incomplete, experimental, or still subject to change.
 - `public_api`: Specifies this method is part of the Alfresco public API and should be stable and well tested.
 - `draft_public_api`: Specifies this method is intended to eventually become part of the public API, but is incomplete or still subject to change.
 - `deprecated`: States this method should be avoided. It may be removed in future versions of Alfresco.
 - `internal`: Indicates this script is for Alfresco use only. This script should not be relied upon between versions.

Response templates (URI templates)

According to the MVC architecture of a web script, a template acts as the role of view, which is responsible for rendering the web script output. You can have more than one template file for different response types. However, at least one template is a mandatory component for any web script. You need to follow the naming convention for a web script response template, which is as follows:

```
<serviceId>.<httpMethod>.<format>.ftl
```

where `serviceId` is the web script ID and `httpMethod` can be `GET`, `POST`, `PUT`, or `DELETE`. `format` specifies the response type format.

For example, if we have a web script that is responsible for getting the news headlines in the XML format, we can have the filename as `getNewsHeadline.get.xml.ftl`.

Response type formats

Any web script can have one or more response types. By default, formats supported by Alfresco for response type are as follows:

- html: text/html
- text: text/plain
- xml: text/xml
- atom: application/atom+xml
- atomentry: application/atom+xml;type=entry
- atomfeed: application/atom+xml;type=feed
- rss: application/rss+xml
- json: application/json
- opensearchdescription: application/opensearchdescription+xml
- mediawiki: text/plain (Media Wiki markup)
- portlet: text/html
- fbml: text/html
- php: text/html
- js: text/javascript
- calendar: text/calendar

There are two ways to specify the format of the response type:

- As format argument in the web script URL. For example, `http://<host>:<port>/<contextPath>/<servicePath>/getNewsHeadline?format=xml`.
- As a part of web script URL directly. For example, `http://<host>:<port>/<contextPath>/<servicePath>/getNewsHeadline.xml`.

This specifies that the list of formats is the out-of-the-box supported format types by Alfresco. If you want to introduce a new format, configure the following Spring Bean declaring the format map with your new format:

```
<bean parent="webscripts.formatmap">
    <property name="formats">
        <props>
            <prop key="wav">audio/x-mpeg-3 </prop>
        </props>
    </property>
</bean>
```

Root objects of FreeMarker

Alfresco provides a default model provided by the JSF template component and template servlet, which provides a number of common root objects useful for most FreeMarker templates. The list of these root objects which you can use for a web script template is as follows:

- `companyhome`: The company home template node. This is available only if authenticated.
- `userhome`: Current user's home space template node. This is available only if authenticated.
- `person`: Current user's person object template node. This is available only if authenticated.
- `args`: A map of any URL parameters passed via the Template Content Servlet.
- `session`: Session-related information. It provides a single-value `session.ticket` for the current authentication ticket.
- `classification`: Read access to classifications and root categories.
- `url`: An object providing access to the URL or parts of the URL that triggered the web script.
- `workflow`: Read access to information on workflow objects and the currently executing workflows for a user.
- `avm`: Root-level object for a WCM web project. Using this API, it is possible to build templates that display content from files in the staging area or sandbox of a web project-based website.
- `argsM`: An associative array of any URL parameters where each key is a parameter name and each value is an array containing the parameter values, even if only one is supplied.
- `guest`: A Boolean indicating whether the web script executes as "Guest".
- `date`: A date representing the date and time of the web script request.
- `server`: An associative array of metadata properties that describe the hosting Alfresco server.
- `roothome`: The repository root node.
- `webscript`: An associative array of metadata properties that describe this web script.

Apart from all of these root objects, the template also has access to root objects created by the execution script, if any script has been associated with the web script. You can also have JavaBeans associated with the web script, and you can create objects there as well, which will be accessed by this FreeMarker template for the web script.

FreeMarker methods for the AVM repository

Each of these root objects has different APIs. For more detail on this you can refer Alfresco wiki's **Template Guide** page at http://wiki.alfresco.com/wiki/Template_Guide and see some examples at http://wiki.alfresco.com/wiki/FreeMarker_Template_Cookbook. In this section, we will talk about the FreeMarker APIs available for the AVM repository.

AVM API

As stated earlier, `avm` is the root object for FreeMarker templates; with this root object, you can access the AVM repository. The available APIs for an AVM root node are as follows:

- `stores`: This returns all store objects in the AVM repository
- `lookupStore(storeId)`: This returns the store object for the specified store ID
- `lookupStoreRoot(storeId)`: This returns the root node for the specified store ID
- `lookupNode(path)`: This returns a single AVM node based on the given full path including store to the node
- `webappsFolderPath(storeId)`: This returns the root path to the AVM webapps folder for the specified store ID
- `getModifiedItems(storeId, username, webapp)`: This returns a sequence of nodes representing modified items for the specified User Sandbox store for a specific webapps folder in the store
- `stagingStore(storeId)`: This returns the Staging Sandbox store for the specified store ID
- `avm.userSandboxStore(storeId, username)`: This returns the User Sandbox store for the specified user and store
- `websiteStagingUrl(storeId)`: This returns the preview URL to the Staging Sandbox for the specified store ID

- `websiteUserSandboxUrl(storeId, username)`: This returns the preview URL to the User's Sandbox for the specified store ID and username
- `assetUrl(path)`: This returns the preview URL to the specified fully qualified AVM path for the asset node
- `assetUrl(storeId, path)`: This returns the preview URL to the specified store and asset path

AVM store API

When you use any of the AVM APIs mentioned in the previous section, you will get the object of an AVM store. This store object can have the APIs described below:

- `id`: This returns the internal ID of the store
- `name`: This returns the name of the store
- `creator`: This returns the user who has created this store
- `createdDate`: This returns the date on which the store was created
- `lookupRoot`: This returns the root node for the store
- `lookupNode(path)`: This returns the node for the specified path relative to `avm webapps` root folder for the store
- `luceneSearch(query)`: This executes the Lucene Search query against the store and returns the search results as a sequence of nodes

AVM node API

Following are the APIs for an AVM node:

- `version`: Returns the node's version, generally it will be -1 (Head revision).
- `path`: Returns fully qualified AVM path to the node.
- `parentPath`: Returns fully qualified AVM path to the parent of the node.
- `isDirectory`: Returns `true` if this node is a directory.
- `isFile`: Returns `true` if this node is a file.
- `isLocked`: Returns `true` if this node is currently locked.
- `isLockOwner`: Returns `true` if this node is locked by the current user.
- `hasLockAccess`: Returns `true` if this user has the permission to perform operations on the node when locked. This is true if the item is either unlocked, locked by the current user, or locked and the current user has the Content Manager role in the associated web project.
- `rename(name)`: This renames the node.

Response status

Web script response is used to inform the status to the client. It uses HTTP response status code for the following scenarios:

- To inform the client about the cause of an error
- To inform the client about the occurrence of an event
- To instruct a client to perform a follow-up request
- To inform a client about the success of a web script

To set this status code, the status root object is available in the web script. And if you are using a Java-backed web script, then overload the API with `status` as one of the parameters. In the script, you can use the status code as follows:

```
status.code = 400;  
status.message = "Required parameter's value has not been provided.";  
status.redirect = true;
```

When you redirect like this, it will render the custom response template for the specified code. Searching of corresponding status templates will be performed in the following order:

- Web script-specific template named `<scriptid>.<method>.<format>. <code>.ftl`
- Web script-specific template named `<scriptid>.<method>.<format>. status.ftl`
- Package-level template named `<format>.<code>.ftl`
- Package-level template named `<format>.status.ftl`
- Default template named `<code>.ftl`
- Default template named `status.ftl`

Web script controller

In this section, we will discuss about the web script controller. Controller is an optional attribute of any web script. If you don't need it, you can omit this section.

Objectives of a controller

This is an optional component of a web script, but generally used when you want to access the repository and perform certain operations. Similar to the role of a controller in the MVC architecture, here also the main objective of the controller is to dictate what to do behind the scenes and what to display in the view. The controller builds a data model that is passed to the FreeMarker response templates and rendered. Web scripts in Alfresco support two types of controllers:

- JavaScript controller
- Java-backed controller

In the following sections, we will talk about these both in detail.

JavaScript controller

A controller is used in order to perform some business logic, and one of the available controllers is JavaScript, which is also known as execution script. You need to follow the naming convention for a web script execution script, which is:

```
<serviceId>.<httpMethod>.js
```

where `serviceId` is the web script ID and `httpMethod` can be GET, POST, PUT, or DELETE.

For example, if we have a web script that is responsible for getting the news, we can have the filename as `getNewsHeadline.get.js`.

Root objects for an execution script

Alfresco provides a set of default root objects also known as Script Node objects that wrap the common Alfresco repository concepts such as nodes, aspects, associations, and properties.

The list of these root objects, which you can use for an execution script, is as follows:

- `companyhome`: The company home template node. This is available only if authenticated.
- `userhome`: Current user's home space template node. This is available only if authenticated.
- `person`: Current user's person object template node. This is available only if authenticated.
- `args`: An associative array of any URL parameters.

- `search`: The host object providing access to Lucene Search.
- `people`: The host object providing access to Alfresco people and groups.
- `actions`: The host object providing invocation of registered Alfresco actions.
- `logger`: The host object providing access to console logging facilities for debugging of scripts.
- `session`: Session-related information such as the current authentication ticket.
- `classification`: Access to the root elements of the classification API.
- `utils`: Access to a library of useful helper functions not provided as part of the generic JavaScript.
- `avm`: Access to WCM objects such as avm stores and web projects.
- `webprojects`: The root of the WCM JavaScript API. It provides access to web projects, sandboxes, WCM assets, and WCM project membership.
- `crossrepository`: Cross repository copy support. This enables copying of nodes between document management spaces and WCM spaces.
- `argsM`: An associative array of any URL parameters.
- `url`: Provides access to the URL (or parts of the URL) that triggered the web script.
- `formdata`: Provides access to multi-part/formdata requests allowing the upload of files via web scripts.
- `model`: An empty associative array that may be populated by the JavaScript. Values placed into this array are available as root objects in web script response templates.
- `roothome`: The repository root node.
- `guest`: A Boolean indicating whether the web script executes as "Guest".
- `server`: An associative array of metadata properties describing the repository server hosting the web script.

JavaScript methods for the AVM repository

Each of these root objects has different APIs. For more details on this, you can refer to the Alfresco wiki's **JavaScript API** page at http://wiki.alfresco.com/wiki/3.2_JavaScript_API and see some examples at http://wiki.alfresco.com/wiki/JavaScript_API_Cookbook. In the next section, we will talk about FreeMarker APIs that are available for the AVM repository.

AVM API

`avm` is the root object for JavaScript APIs, and with this root object you can access the AVM repository. The available APIs for the `avm` root node are as follows:

- `stores`: This returns all store objects in the AVM repository
- `lookupStore(storeid)`: This returns the store object for the specified store ID
- `lookupStoreRoot(storeid)`: This returns the root node for the specified store ID
- `lookupNode(path)`: This returns a single AVM node based on the given full path, including store to the node
- `webappsFolderPath(storeid)`: This returns the root path to the AVM webapps folder for the specified store ID

AVM store API

When you use any of the previous mentioned AVM API, you will get the object of the AVM store. This store object can have the APIs described next:

- `id`: This returns the internal ID of the store
- `name`: This returns the name of the store
- `creator`: This returns the user who has created this store
- `createdDate`: This returns the date on which the store was created
- `lookupRoot`: This returns the root node for the store
- `lookupNode(path)`: This returns the node for the specified path relative to the AVM webapps root folder for the store
- `luceneSearch(query)`: This executes a search against the store and returns an array of AVM nodes as the result

AVM node API

- `version`: Returns node's version. Generally it will be -1 (Head revision).
- `path`: Returns fully qualified AVM path to the node.
- `parentPath`: Returns fully qualified AVM path to the parent of the node.
- `isDirectory`: Returns `true` if this node is a directory.
- `isFile`: Returns `true` if this node is a file.
- `isLocked`: Returns `true` if this node is currently locked.

- `isLockOwner`: Returns `true` if this node is locked by the current user.
- `hasLockAccess`: Returns `true` if this user has the permission to perform operations on the node when locked. This is true if the item is either unlocked, locked by the current user, or locked and the current user has the Content Manager role in the associated web project.
- `rename(name)`: This renames the node. In AVM you cannot change the `cm:name` property directly to change the name.

Java-backed controller

If the script and template are not sufficient enough to achieve the required functionality, you can have a Java Bean where you will have full control over the Alfresco Repository and also the Java API. With Web script, you can bind a Java Bean as a Spring Bean in Alfresco. When you have a Java bean, you can also have a combination of web Java script with it. The sequence of execution will be first Java script will be executed first and then Java bean if you have both used for any web script.

How to declare a Java Bean

To bind a Java Bean as a Spring Bean for any web script, you need to make a bean entry in the `web-script-custom-context.xml` file in the `/WEB-INF/classes/alfresco/extension` folder. Also, you have to follow a proper naming convention for that bean ID, which will be:

```
id="webscript.<packageId>.<serviceId>.<httpMethod>"
```

where `packageId` is the package in which the web script descriptor file is located, `serviceId` is the web script ID, and `httpMethod` can be `GET`, `POST`, `PUT`, or `DELETE`.

Also, the parent of this bean should be `webscript` while configuring this:

```
<bean id="webscript.org.cignex.news.getNewsHeadline.get"
      class="com.cignex.web.scripts.bean.news.GetNewsHeadline"
      parent="webscript" />
```

Creating a Java Bean class

When creating a Java Bean class, `GetNewsHeadline`, as in the previous section, this class must implement the `org.alfresco.web.scripts.WebScript` interface, which has the method:

```
public void execute(WebScriptRequest req, WebScriptResponse res)
```

You need to implement this method in your Java Bean class. There are two helper classes provided to simplify the development of a Java-backed web script: `org.alfresco.web.scripts.AbstractWebScript` and `org.alfresco.web.scripts.DeclarativeWebScript`.

Implementing web scripts

Implementation of a web script consists of mainly four parts. These are explained in the following sections.

Creating a web script

Create the required files for a web script. A description document and a rendering template are required. You can also have a controller script and other components described in the earlier sections of this chapter, if required.

Storing the web script

There are two ways of storing a web script in Alfresco. We will discuss both these methods:

Storing it on the filesystem

If you want to store it on the filesystem, you need to store it in the `<alfresco_server>/tomcat/shared/extension/templates/webscripts` folder. You can create the desired folder structure inside this folder and then put all of the web script files in there.

Storing it in Alfresco Explorer

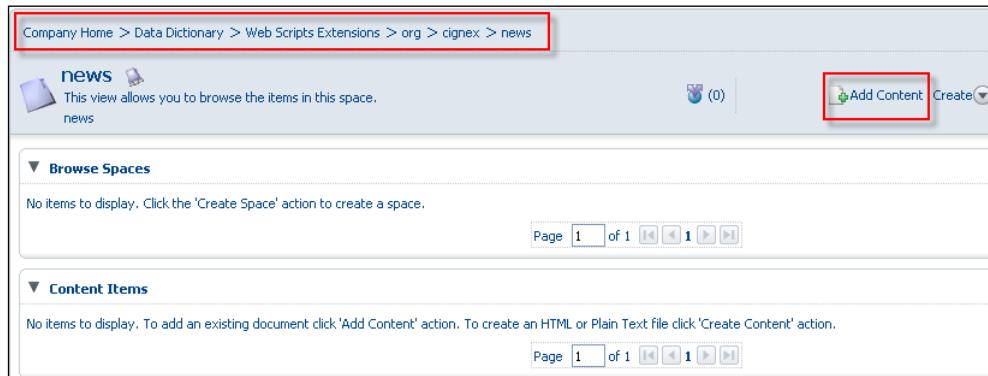
You can store the web script files in the **Company Home | Data Dictionary | Web Scripts Extensions** page. Create the required folder structure under this path as shown in the next screenshot:



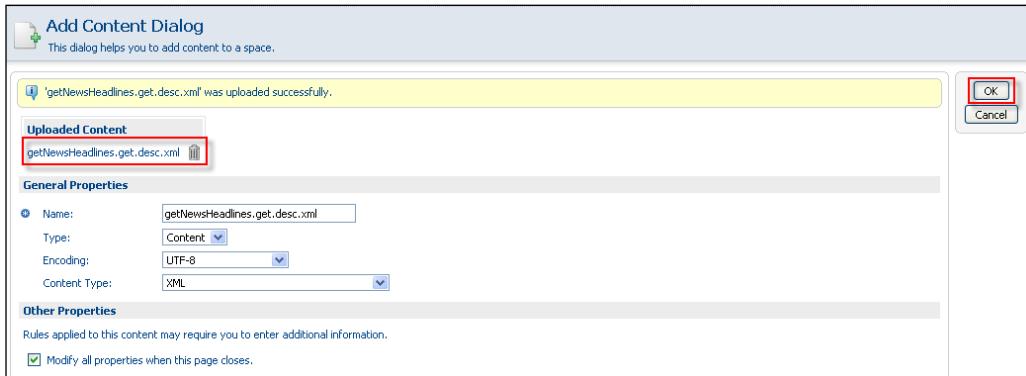
Click on **Create Space** and provide a name for the folder you want to create and click on the **Create Space** button.



Once done with the creation of folders, place the web script files, description document, rendering template, and controller script (if you have one) in that folder with the help of the **Add Content** option, as shown in the following screenshot:

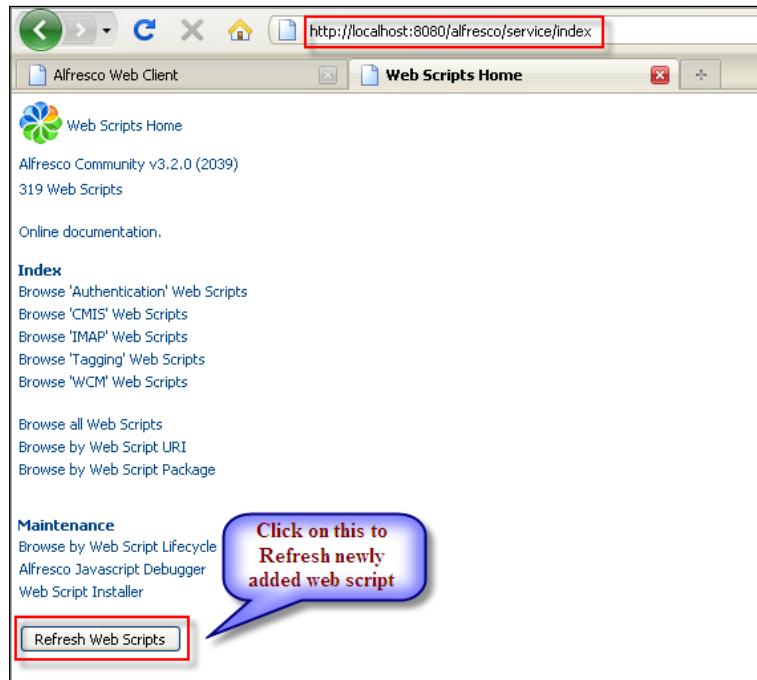


To upload the files, click on **Add Content** as shown in the previous screenshot. Then browse to the file on your local desktop and click on **OK** to upload that content to the specified space:

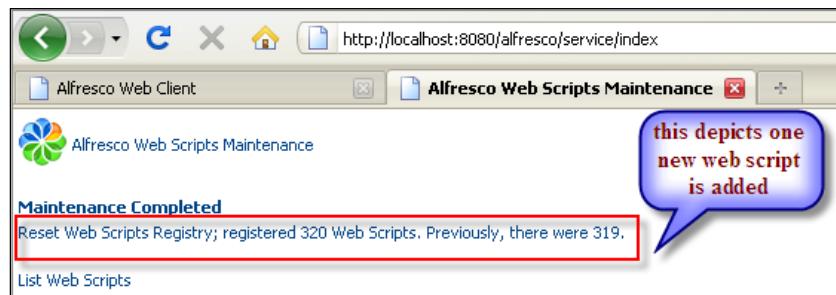


Registering the web script

Once you are done with storing the required web script files, you need to register them in Alfresco. For registering the web scripts stored in Alfresco Explorer, navigate to <http://localhost:8080/alfresco/service/index> and click on the **Refresh Web Scripts** button, as shown in the following screenshot:



You will see a message about the completion of maintenance of web scripts, which also specifies the number of web scripts:



Click on the **Refresh Web Scripts** button. You will see a message showing how many web scripts have recently been found and registered.

While registering web scripts stored on the filesystem for the first time, you need to restart the Alfresco server. After the web script is registered, if you change something in the execution script or presentation template, you can click on the **Refresh Web Scripts** button on the index page of web scripts; there is no need to restart the server.

In this way, Alfresco supports **Hot Deployment** of web scripts.

Listing the web scripts for external access

Once the web script is registered, you can start using it with the URL. Navigate to `http://localhost:8080/alfresco/service/index`, and you will see the different browse options there. If you want to check it, you can click on any of the available browse options. Clicking on **Browse by Web Script URI** will browse all of the web scripts' URI. By doing so, you can verify your web script is there:



Integrating WCM with external applications—case studies

In an earlier chapter on web forms, we talked about how we can use Alfresco WCM for content production. Now in this section, we will discuss some of the case studies where the actual application is outside of Alfresco, but the content will be stored in Alfresco WCM and needs to be served from Alfresco only. For that, we can use Alfresco web scripts, which will act as an integration point for Alfresco WCM content and the actual application.

Integrating Alfresco WCM and Liferay with a news portlet

Let's consider the case where we want to have a news portlet in some Liferay Portal Application. So here, rendering of the portlet will be handled in Liferay, but the actual news content and the headlines for the news are stored in Alfresco WCM.

We have some news content stored in the /ROOT/common/inc folder, and we will try to fetch the headlines for these news items.

Web script for getting news headlines

In this example, we will create a web script for fetching the news headlines from Alfresco. We will use JavaScript as the controller and will return the response in XML format. This XML response will then be processed by a portlet in Liferay and the final news headline will be displayed on the news portlet.

Description document

The description document, `getNewsHeadline.get.desc.xml`, which describes the URL and the authentication mechanism for the web script is as follows:

```
<webscript>
  <shortname>Listing Of News Headlines</shortname>
  <description>Listing Of News Headlines for Cignex home page thru
  Webscript</description>
  <url>/org/cignex/news/getNewsHeadlines.xml?storeId={storeId}</url>
  <authentication>user</authentication>
</webscript>
```

Execution script

JavaScript controller, `getNewsHeadline.get.js`, executes the search query to fetch the news items' nodes from the WCM repository. This is done using the Lucene Search API to search the content of a specific web form. Here in this example, it's the news web form. Keep in mind that this Lucene Search will only be applied to the Staging Sandbox of a web project. Here we have `storeId` as one of the web script parameters, but that should point to the Staging Sandbox of a web project:

```
var storeId = args["storeId"];
var newsNodes = new Array();
if(storeId != null) {
    var webProjectStore = avm.lookupStore(storeId);
    if(webProjectStore != null) {
        var queryForNews = "@wca\\:parentformname:news AND @cm\\:\\name:\\*.
xml\"";
        var resultNodes = webProjectStore.luceneSearch(queryForNews);
        if(resultNodes != null) {
            logger.log("Reslut nodes: " + resultNodes.length);
            for (var i=0; i<resultNodes.length; i++) {
                {
                    newsNodes[i] = resultNodes[i];
                }
            }
        }
        if(newsNodes != null) {
            model.m_newsNodes=newsNodes;
        }
    }
else{
    logger.log("ERROR: 'store' argument not specified.");
}
```

Response template

The rendering template for the XML response, `getNewsHeading.xml.ftl` file, is as follows:

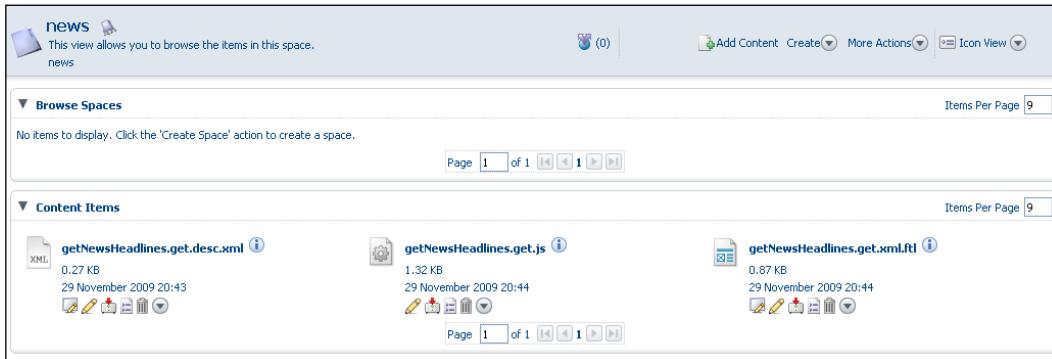
```
<#ftl ns_prefixes={"D", "http://www.alfrescobook.com/webforms"}>
<news>
<if m_newsNodes?exists>
    <#list m_newsNodes as newsNode>
        <#assign newsItemDom = newsNode.xmlNodeModel/>
        <if newsItemDom?exists>
            <#assign newsItem = newsItemDom.news>
            <if newsItem?exists>
                <newsItem>
                    <newsId>${newsNode.name}</newsId>
```

```
<title><! [CDATA[ ${newsItem.shortTitle} ]]></title>
<header><! [CDATA[ ${newsItem.contentHeader} ]]></header>
<date>${newsItem.newsDate}</date>
</newsItem>
</#if>
</#if>
</#list>
</#if>
</news>
```

Storing/registering a web script in Alfresco

Once done with the development of a web script with all of the three mentioned files, we need to store it in an appropriate place. Let's store it through Alfresco Explorer. We will store the web script files in the **Company Home | Data Dictionary | Web Scripts Extensions** folder.

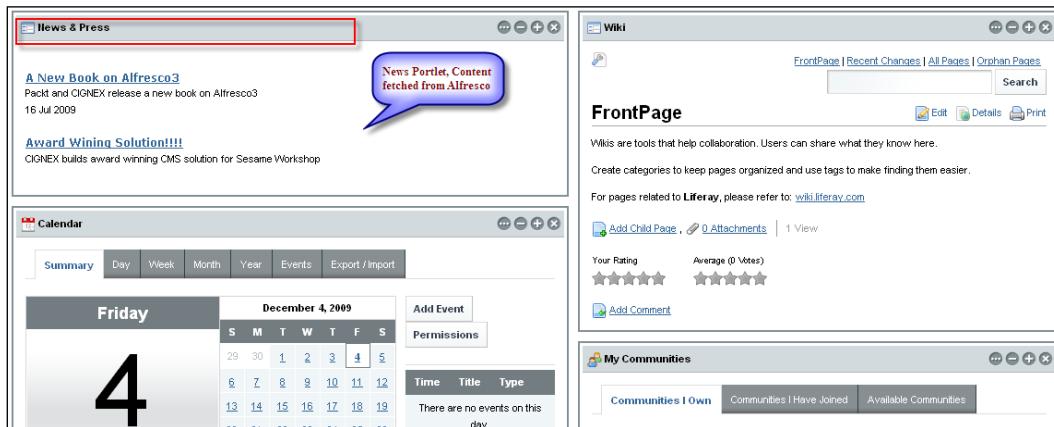
Create the folder structure as `org/cignex/news` inside the **Web Scripts Extensions** folder and upload all of the three web script-related files here, as mentioned in a previous section on storing a web script. After uploading, you should see all of the three files in the **news** folder, as shown in the next screenshot:



Once done with this, register the web scripts by navigating to `http://localhost:8080/alfresco/service/index` and clicking on the **Refresh Web Scripts** button as mentioned before in the section on registering a web script. Now that the new web script is registered, we can start using it with the URL. If you want to check it, you can click on any of the browse options. Clicking on **Browse by Web Script URI** will browse all of the web scripts' URI. By doing so, you can verify your web script is there. Now we will see how to use this web script in Liferay.

Portlet in Liferay

Now in Liferay, you need to create a new portlet and call our web script from that portlet. The following screenshot shows a snap of a portal page where one of the portlets is **News & Press**. The content of this portlet is fetched from the Alfresco WCM:



You can download the code samples for web script-related files and the news portlet .war file for Liferay from the Packt website.

Integrating Alfresco WCM and Drupal with monthly blogs

In this case study, we will see the integration between Alfresco WCM and Drupal. This will be an integration of two different technologies, Java and PHP. We are going to take an example from monthly blogs. As in *Chapter 4, Web Content Production with Web Forms*, we have a web form for creating different blog items in the WCM. So we do have some blog items available with us in the WCM, which we will display on a Drupal page.

Web script for getting monthly blogs

Here we will have a web script that will fetch the blog content from Alfresco WCM for a particular month based on the published date of the blog content. We will use JavaScript as a controller and will return the response in HTML format. So in Drupal, we do not need to do any complex processing; just display the HTML as it is.

Description document

The description document, `getMonthlyBlogs.get.desc.xml`, which describes the URL and the authentication mechanism for the web script is as follows:

```
<webscript>
    <shortname>Listing Of Blogs</shortname>
    <description>Listing Of Blogs for Cignex home page thru Webscript
    </description>
    <url>/org/cignex/blogs/getBlogs.xml?storeId={storeId}&month=
{monthnumber}</url>
    <authentication>user</authentication>
</webscript>
```

Execution script

The JavaScript controller, `getMonthlyBlogs.get.js`, executes the search query to fetch the blog items' nodes from the WCM repository. This is done using the Lucene Search API to search the contents of a specific web form; here it's the blog's web form. Keep in mind that this Lucene Search will only be applied to the Staging Sandbox of a web project. Here we have `storeId` as one of the web script parameters, but that should point to the Staging Sandbox of a web project:

```
var storeId = args["storeId"];
var blogNodes = new Array();
if(storeId != null) {
    var webProjectStore = avm.lookupStore(storeId);
    if(webProjectStore != null) {
        var queryForBlogs = "@wca\\:\\parentformname:blog AND @
cm\\:\\name:\\*.xml\\\"";
        var resultNodes = webProjectStore.luceneSearch(queryForBlogs);
        if(resultNodes != null) {
            logger.log("Result nodes: " + resultNodes.length);
            for (var i=0; i<resultNodes.length; i++) {
                {
                    blogNodes[i] = resultNodes[i];
                }
            }
        }
        if(blogNodes != null) {
            model.m_blogNodes=blogNodes;
        }
    }
} else{
    logger.log("ERROR: 'store' argument not specified.");
}
```

Response template

The rendering template for the HTML response, `getMonthlyBlogs.get.html.ftl` file, is as follows:

```
<#ftl ns_prefixes={"D", "http://www.alfrescobook.com/webforms"}>
<if args["month"]?exists>
    <#assign monthnumber=args["month"]>
<else>
    <#assign dateformat="yyyy-MM-dd">
    <#assign monthnumber=date?string(dateformat)?substring(5, 7)>
</if>
<switch monthnumber>
    <case '1'>
        <#assign month="January">
        <#break>
    <case '2'>
        <#assign month="February">
        <#break>
    <case '3'>
        <#assign month="March">
        <#break>
    <case '4'>
        <#assign month="April">
        <#break>
    <case '5'>
        <#assign month="May">
        <#break>
    <case '6'>
        <#assign month="June">
        <#break>
    <case '7'>
        <#assign month="July">
        <#break>
    <case '8'>
        <#assign month="August">
        <#break>
    <case '9'>
        <#assign month="September">
        <#break>
    <case '10'>
        <#assign month="October">
        <#break>
    <case '11'>
        <#assign month="November">
        <#break>
    <case '12'>
        <#assign month="December">
        <#break>
</switch>
```

```
<html>
<body>
<if m_blogNodes?exists>
    <font face="Verdana" color="CC3333">
        <h3>Blogs for Month - ${month} </h3>
    </font>

<table width="100%" cellspacing="0" cellpadding="0" border="0">
    <#list m_blogNodes as blogNode>
        <#assign blogItemDom = blogNode.xmlNodeModel/>
        <if blogItemDom?exists>
            <#assign blogItem = blogItemDom.blog>
            <if blogItem?exists>
                <if blogItem.publishedDate?substring(5,7) == monthnumber>
                    <tr>
                        <td>
                            <table>
                                <tr>
                                    <td>
                                        <p> <a href="#"> <b>${blogItem.mainTitle} </b> </a> </p>
                                    </td>
                                </tr>
                                <tr>
                                    <td>
                                        <p> ${blogItem.publishedDate?substring(0,10)} </p>
                                    </td>
                                </tr>
                            </table>
                        </td>
                    </tr>
                </if>
            </if>
            </if>
        </if>
    </#list>
</table>
</if>
</body>
</html>
```

Here we are fetching the blog items for a particular month with the parameter as month in the web script URL. But if you don't specify this parameter, then the current month will be considered to fetch the blog items.

Storing/registering the web script in Alfresco

Once done with the development of a web script with all of the mentioned files, we need to store it to the **Web Scripts Extensions** folder in Alfresco Explorer. Navigate to the **Company Home | Data Dictionary | Web Scripts Extensions** folder. Create the folder structure as `org/cignex/blogs`. Upload all of the three mentioned files here in this folder, as shown in the following screenshot:

Company Home > Data Dictionary > Web Scripts Extensions > org > cignex > blogs

blogs This view allows you to browse the items in this space.

Add Content Create More Actions

Browse Spaces

No items to display. Click the 'Create Space' action to create a space.

Content Items

getBlogs.get.desc.xml ⓘ 0.27 KB 1 December 2009 16:34 Edit Delete View Download	getBlogs.get.html.ftl ⓘ 1.74 KB 1 December 2009 15:13 Edit Delete View Download	getBlogs.get.js ⓘ 0.63 KB 1 December 2009 16:35 Edit Delete View Download
--	--	--

Once done with this, we need to register our web script. Navigate to `http://localhost:8080/alfresco/service/index` and click on the **Refresh Web Scripts** button. Now that our web script is registered, we can start using it with the URL.

Calling the web script in Drupal

Now in Drupal, we need to create a new page where we will display this month's blog entries. For that we will create a new module:

localhost

User login

Username:

Password:

Create new account
 Request new password

Home

Monthly Blogs
Blogs for Month - October

A simple solution for merging various documents for your business needs by amita

2009-10-26

Drupal



You can download the code samples for web script-related files and blog module-related files for Drupal from the Packt website.



Integrating Alfresco WCM with any J2EE web application

In the previous example, we saw how we can display the news headlines on the news portlet. Now, in this case study, we will learn to view the details about a particular news item. Suppose you have any J2EE-based application and you want to display the news page with some specific news item. This can be integrated with the earlier portlet example as well, like say, on the portal home page, you have one news portlet, which displays the news headlines, but when you click on any news item, you should be able to see that particular news item. However, here we will take the example of any J2EE web application and we will call this web script from a normal JSP page.

Web script for getting the details of a particular news item

In this case study, as mentioned before, we will talk about the web script that will fetch the details of a particular news item. In this example, we will develop a Java-backed web script so that you can have an idea about how to create Java-backed web scripts. The response format will be HTML.

Description document

The description document, `getNewsItem.get.desc.xml`, which describes the URL and the authentication mechanism for the web script is as follows:

```
<webscript>
  <shortname>Listing Of News Item</shortname>
  <description>Listing Of News Item for Cignex News page thru
    Webscript</description>
  <url>/org/cignex/news/getNewsItem.html?storeId={storeId}&
    newsId={newsId}</url>
  <authentication>user</authentication>
</webscript>
```

Java-backed Bean for a web script

In this example, instead of using JavaScript as the execution script controller, we will use Java Bean as a controller and perform our business logic to fetch the particular content and process that is in this Java class.

We will create a Java Bean class named `GetNewsItem`, which extends the `DeclarativeWebScript` class and will override the `execute` method as follows:

```
public class GetNewsItem extends DeclarativeWebScript {
    protected Map<String, Object> executeImpl(WebScriptRequest req,
                                                Status status) {
        // Perform the business logic here in this method
    }
}
```

Now, we need to configure this Bean in a Spring configuration file as a controller for a web script. For that we will create a configuration XML file named `web-script-custom-context.xml` in `tomcat/webapps/alfresco/WEB-INF/classes/alfresco/extension`.

Make an entry for this Java class as a Spring Bean in `web-script-custom-context.xml` as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE beans PUBLIC '-//SPRING//DTD BEAN//EN'
  'http://www.springframework.org/dtd/spring-beans.dtd'>
<beans>
    <!-- Web Script Storage -->
    <bean id="webscript.org.cignex.news.getNewsItem.get"
          class="com.cignex.web.scripts.bean.news.GetNewsItem"
          parent="webscript">
        <property name="contentService" ref="ContentService" />
        <property name="avmService" ref="AVMService" />
    </bean>
</beans>
```

Response template

The rendering template for the HTML response, `getNewsItem.get.html.ftl` file, is as follows:

```
<#ftl ns_prefixes={"D", "http://www.alfrescobook.com/news"}>
<if m_news?exists>
    <table width="100%" cellspacing="0" cellpadding="0" border="0">
        <tr>
            <#assign newsItem = m_news>
```

```
<td>
<h2>${newsItem.contentHeader}</h2>
<if newsItem.newsDate != "<">
    <strong>News Date:</strong>
    ${newsItem.newsDate}
</if>
<if newsItem.contentSubHeader != "<">
    <h4>${newsItem.contentSubHeader}</h4>
</if>

<table width="50%" cellspacing="0" cellpadding="5" border="0">
<tr>
    <td>
        <tr>
            <h4> ${newsItem.imageTitle}</h4>
            <if newsItem.contentGraphic != "<">
                <td>
                    
                </td>
            </if>
            </tr>
        </td>
        <td>
            <strong> ${newsItem.imageCaption}</strong>
        </td>
    </tr>
    <tr>
        <td>
            ${newsItem.contentText}
        </td>
    </tr>
</table>
</td>
</tr>
</table>
</if>
</html>
```

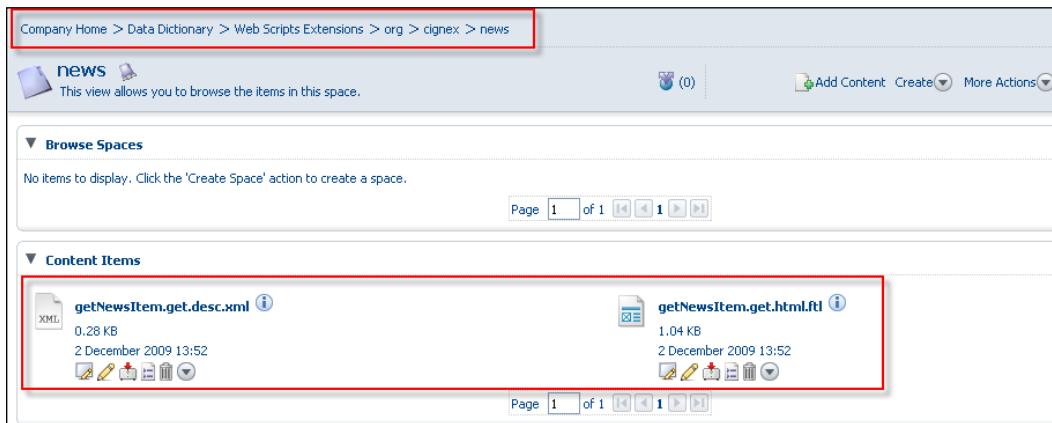
Here we are fetching the details of a news item for a particular news ID, based on the ID passed; the details for that news item will be returned from Alfresco.

 In this example, we have not used the Lucene Search API; rather we have used AVM APIs and this web script is flexible enough to search in any of the sandboxes for any web project. Suppose you want to search for some content in your own User Sandbox (the content is still not promoted to the Staging Sandbox); you can pass the store ID as `wwwcignex-admin`. Here '`wwwcignex`' is the web project name and '`admin`' is the username, so `wwwcignex-admin` will be the sandbox of the '`wwwcignex`' web project for a user '`admin`'.

This will help in testing the content, which you have created, before submitting this change to the Staging Sandbox.

Storing/registering the web script in Alfresco

Once done with development of the web script with all of the three mentioned files, we need to store it to the **Web Scripts Extensions** folder in Alfresco Explorer. Navigate to **Company Home | Data Dictionary | Web Scripts Extensions** and create the folder structure as `org/cignex/news`. Upload all of the three mentioned files here in this folder, as shown in the next screenshot:



Once we have done this, we need to register the web scripts.

Go to `http://localhost:8080/alfresco/service/index` and click on the **Refresh Web Scripts** button. Now that our new web script is registered, we can start using it with the URL.

Calling web scripts from a JSP page

Now we will create a simple JSP page from which we will make a call to our web script to render the news page. This JSP can be a part of any of your web applications.

To call a web script from the JSP, this will be an HTTP call. Just add the following scriptlet in the JSP file wherever you want to include the HTML output of this web script:

```
<%
    String resultString = "";

    HttpClient client = new HttpClient();

    client.getState().setCredentials(
        new AuthScope("localhost", 8080, "Alfresco"),
        new UsernamePasswordCredentials("admin", "admin")
    );
    GetMethod get = new
    GetMethod("http://localhost:8080/alfresco/service/org/cignex/news/
    getNewsItem.html?storeId=wwwcignex&newsId=newsAlfrescoBookRelease.
    xml");
    get.setDoAuthentication(true);
    try {
        int status = client.executeMethod( get );
        resultString = get.getResponseBodyAsString();
        out.println(resultString);
    } finally {
        get.releaseConnection();
    }

%>
```

Because we are using an HTTP client, we also need to import the used classes here in this scriptlet before using it. To import those, we can use the page directive of JSP as follows:

```
<%@ page import="org.apache.commons.httpclient.HttpClient" %>
<%@ page import="org.apache.commons.httpclient.
UsernamePasswordCredentials" %>
<%@ page import="org.apache.commons.httpclient.auth.AuthScope" %>
<%@ page import="org.apache.commons.httpclient.methods.GetMethod" %>
```

This will import all of the required classes.



The source files for the whole JSP page can be downloaded from the Packt website.

The following screenshot is the output of that JSP page. This JSP page is part of a sample web application, which is outside of Alfresco and deployed on a separate Tomcat web server:

The screenshot shows a web browser window with the URL <http://localhost:28080/sample/common/pages/TestNews.jsp>. A blue callout bubble points to the URL bar with the text "Jsp page from a web application outside of Alfresco". Another blue callout bubble points to the main content area with the text "News Content fetched from Alfresco WCM".

The page itself is a news article titled "Packt and CIGNEX release a new book on Alfresco3". It includes a photo of five people at a book release event, a news date of "16 Jul 2009", and a subtitle "Build a Java-based Enterprise CMS using New Book".

A red box highlights the left sidebar, which contains links for "LIFERAY by CIGNEX" and "ALFRESCO by CIGNEX", each with a "FREE CHAPTER" link. Below this is a section titled "TAKE THE ALFRESCO CHALLENGE AND WIN" with an image of a trophy.

At the bottom of the page, there is a paragraph about Alfresco 3 and a link to "Alfresco 3 Enterprise Content Management Implementation".

Enhancing the news item web script

In this case study, we will make enhancements to the previously mentioned news web script. Consider a case where you have some part of the content that is being used in multiple places, such as some contact number, e-mail address, link URLs, and so on. When this kind of content needs to be modified in the future, you will need to modify it at multiple places. Instead, we can have a concept where we store this kind of content at one place and then refer that in all of the places when we want to actually use it. In this way, it will enable reusability and it will be easier to modify it in the future. No longer will we have to modify the same content at multiple places; we just need to modify it at one place and that will be reflected in all of the places where it is being used.

Consider this kind of content as tag and store it separately in one place only. We can then refer to it with its tag ID in the actual content when it needs to be used. This means we will have one tag mapping XML file where we will store all these kinds of tags with some ID and value. And then while creating any content, if we want to use it, we will refer it with the ID, and at the time of rendering the response through the web script, it will replace those tag IDs with actual values.

Create a TagMapping.xml file in the /ROOT/common folder under your web project using the tag_mapping web form.



Download the XSD file for this web form from the Packt website.



In the news content used in the previous case study, the website URL for the book is used at multiple places. We will create a tag for this and then use this tag in the news content.

A sample TagMapping.xml file looks like:

```
<?xml version="1.0" encoding="UTF-8"?>
<items xmlns:alf="http://www.alfresco.org" xmlns:chiba="http://
chiba.sourceforge.net/xforms" xmlns:ev="http://www.w3.org/2001/xml-
events" xmlns:xf="http://www.w3.org/2002/xforms" xmlns:xhtml="http://
www.w3.org/1999/xhtml" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<item>
<id>phone_number</id>
<value>1-408-923-4231</value>
</item>
<item>
<id>book_website</id>
```

```
<value>http://www.packtpub.com/alfresco-3-enterprise-content-
    management-implementation/book/mid/160609knbhtv</value>
</item>
</items>
```

Here we have created a tag for the URL of the website for our Alfresco book with ID `book_website`, and in value we have specified the URL for this.

Now when we create XML content for the news, wherever we want to mention the book URL, instead of writing the actual URL, we will refer this tag with ID `book_website` as:

```
<a href="${book_website}">Alfresco 3 Enterprise Content Management
Implementation</a>
```

We need to use it as `${tag_id}` only (syntax for a FreeMarker template), because we are going to use the template service to replace this tag's runtime with the actual value of this tag.

Web script for getting the details of a particular news item

Now let's look at the web script changes for this. In this example, we will see how we can fetch the details of an individual news item.

Description document

The description document, `getNewsItem.get.desc.xml`, will be the same as used in the previous example.

Java-backed Bean for web scripts

We use the same Java Bean class, `GetNewsItem`, which extends the `DeclarativeWebScript` class, but we will modify this class as follows:

1. Render the intermediate template model (`getNewsItem.get.html.model.ftl`) with the `renderTemplate` method of template service as:

```
renderTemplate(templatePath, map, out);
```

where `templatePath` is the path for the file `getNewsItem.get.html.model.ftl`, `map` is the model map of the news item, and `out` is used to store the intermediate result of this rendering.

2. Create a tag model map by processing the TagMapping.xml file.
3. Then process this tag model over the result of that intermediate template rendering with the processTagging method of template service as:

```
result = templateService.processTemplateString(  
        "freemarker", template, modelMap);
```

where freemarker indicates the use of FreeMarker as the template processor, template is the intermediate result generated by rendering of the template in the previous step, modelMap is the model map created by processing the tag mapping file, and result will hold the final result after replacing all of the tags with actual values.

Here we are using the template service to render and process the FreeMarker template.

4. Now in the Spring Bean configuration file, web-script-custom-context.xml, in the tomcat/webapps/alfresco/WEB-INF/classes/alfresco/extension folder, we need to add template service as a property for that Bean as we are using this new service for our example as:

```
<?xml version='1.0' encoding='UTF-8'?>  
<!DOCTYPE beans PUBLIC '-//SPRING//DTD BEAN//EN'  
      'http://www.springframework.org/dtd/spring-beans.dtd'>  
<beans>  
    <!-- Web Script Storage -->  
    <bean id="webscript.org.cignex.news.getNewsItem.get" class="com.  
cignex.web.scripts.bean.news.GetNewsItem" parent="webscript">  
      <property name="contentService" ref="ContentService" />  
      <property name="avmService" ref="AVMService" />  
      <property name="templateService" ref="TemplateService" />  
    </bean>  
</beans>
```

Response template

The intermediate rendering template for the HTML response, the getNewsItem.get.html.model.ftl file is as follows:

```
<#ftl ns_prefixes={"D", "http://www.alfrescobook.com/news"}>  
<if m_news?exists>  
  <table width="100%" cellspacing="0" cellpadding="0" border="0">  
    <tr>  
      <#assign newsItem = m_news>  
      <td>  
        <h2>${newsItem.contentHeader}</h2>  
        <if newsItem.newsDate != "">
```

```
<strong>News Date:</strong>
${newsItem.newsDate}
</#if>
<#if newsItem.contentSubHeader != "" >
<h4>${newsItem.contentSubHeader}</h4>
</#if>

<table width="50%" cellspacing="0" cellpadding="5" border="0">
<tr>
<td>
<tr>
<h4> ${newsItem.imageTitle}</h4>
<#if newsItem.contentGraphic != "" >
<td>

</td>
</#if>
</tr>
</td>
<td>
<strong> ${newsItem.imageCaption}</strong>
</td>
</tr>
<tr>
<td>
${newsItem.contentText}
</td>
</tr>
</table>
</td>
</tr>
</table>
</#if>
</html>
```

And finally, the rendering template, the `getNewsItem.get.html.ftl` file, is:

```
<html>
${finalXML}
</html>
```

Storing / registering the web script in Alfresco

There is no change in the description document, but upload the new template file `getNewsItem.get.html.model.ftl` in the **Company Home | Data Dictionary | Web Scripts Extensions | org | cignex | news** folder. Also, modify the `getNewsItem.get.html.ftl` file that already exists there accordingly.

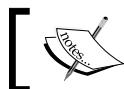
Navigate to `http://localhost:8080/alfresco/service/index` and click on the Refresh Web Scripts button to view the changes.

Calling the web script from a JSP page

There will not be any changes for this. The same JSP that was used in the previous case can be used to call this web script, and the same result will be displayed:



As shown in the previous screenshot, the actual book page URL being displayed in the status bar comes from the tag mapping file.



You can download the code samples for web script-related and blog module-related files for this case study from the Packt website.



Integrating Alfresco WCM and a Surf-based web application

In this case study, we will refer to the same web script we used in the first case study (*Integrating Alfresco WCM and Liferay with a news portlet*). For integrating this with a Surf-based web application, we need to create a JSON response format for the same web script.

Refer to the *Integrating Alfresco WCM and Liferay with a news portlet* section for more details on the web script for getting a news headline.

Response template

The rendering template for the JSON response, the `getNewsItem.get.json.ftl` file, is as follows:

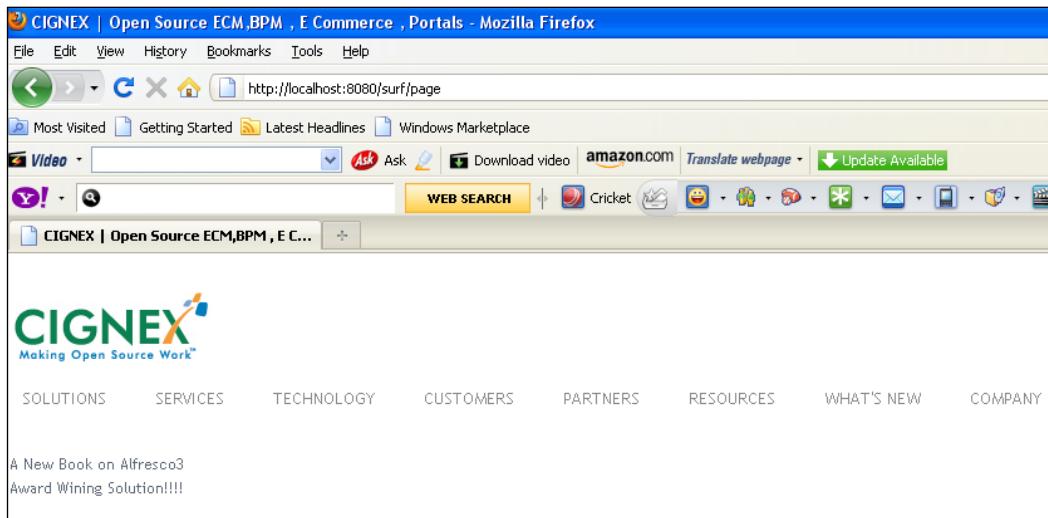
```
<#escape x as jsonUtils.encodeJSONString(x) >
{
    "newsItems": [
        <if m_newsNodes?exists>
            <list m_newsNodes as newsNode>
                <assign newsItemDom = newsNode.xmlNodeModel/>
                <if newsItemDom?exists>
                    <assign newsItem = newsItemDom.news>
                    <if newsItem?exists>
                        {
                            "Headline": "${newsItem.contentHeader}"
                        }
                    </if>
                </if>
                <if newsNode_has_next>, </if>
            </list>
        </if>
    ]
}
</escape>
```

Integrating web scripts with a SURF application

In the *Communication with Web Content Management* section of *Chapter 9*, we have already explained the configuration required at Surf side to call web scripts from Alfresco, along with a UI web script that will be responsible for rendering the page. And the web script we mentioned earlier in this section is a data web script, which will be responsible for fetching content from the repository.

In Surf, we have already created a header web script to display the main page. We have also created a web script to display the news header in Surf. Now we will call the news web script in the same (header) web script to integrate news headline on the same page.

The following screenshot shows the main page from the Surf application where we have integrated the WCM web script. Content is fetched from the repository:



You can download the code samples
from the Packt website.



Summary

In this chapter, we covered the REST architecture at high level and saw how an Alfresco web script allows users to integrate Alfresco with any external application easily, independent of the technology. In this way, you can use Alfresco WCM purely for content authoring and as a content production system without worrying about the frontend rendering of it. Hence, the same Alfresco WCM content can be used and rendered in different ways with multiple applications. Again, web scripts support different output formats, including XML, JSON, and so on, so you can generate the output in the format expected by the frontend application very easily. Mainly we discussed:

- REST architecture and web script framework in Alfresco
- Different components of a web script
- How to develop web scripts in Alfresco
- How to integrate Alfresco WCM with external systems using web scripts

In the next chapter, we will discuss how we can leverage the Alfresco framework for WCM.

11

Leveraging Alfresco Framework for WCM

Alfresco is the leading provider of open source Enterprise Content Management and provides Enterprise grade, scalable, robust, portable, and reliable solutions for managing any type of content, including documents, digital assets, and web content. Alfresco has different modules to manage different kinds of content. Document Management (DM) and Web Content Management (WCM) are the two main and widely used modules of Alfresco.

Alfresco Document Management captures, shares, and retains content, enabling users to version, search, and simply build their own content applications. The Alfresco Web Content Management allows organizations to rapidly create and more effectively maintain dynamic Internet, intranet, and extranet sites, enabling a shortened web development cycle, providing high return on investment and low cost of ownership. Alfresco framework is built on state-of-the-art open source frameworks such as Spring, Hibernate, Lucene, and JSF. DM and WCM are two different feature sets that are built on the common infrastructure framework of Alfresco and also have services such as security, workflow, library, search, and so on, which can be used across the application for any module. In this chapter, we will discuss how you can leverage the Alfresco DM features for the WCM and cover the following topics for WCM:

- Membership and Security Mechanisms
- FFmpeg Integration
- DM content in WCM
- Image Transformation
- Advance Search
- Metadata Extractor

Membership and Security Mechanism

The Alfresco security model is flexible and allows you to choose either its built-in security or an external security model defined by your organization, by using systems such as LDAP and Active Directory. You will understand various security models and learn to choose the one that is most suited to your enterprise's requirements. The Alfresco membership system is highly scalable and can cater to a number of users and content managers. The Alfresco WCM can also leverage the security mechanism provided by Alfresco DM.

Consider a case where we have people from different departments for any company who need to access the Alfresco WCM system. These companies already have their own directory-based Central Authentication System. Here you can have two possibilities:

- Use Alfresco's out-of-the-box membership system and create the user accounts for all those users in Alfresco.
- Configure Alfresco with LDAP for centralized Identity Management where all the users from existing directory LDAP will be imported in Alfresco if they need to access Alfresco and will be authenticated via Central Authentication System-LDAP.

As the company is already having their own directory, the second approach would be proffered. We can configure Alfresco with LDAP and import the users in Alfresco. Once the users are available in Alfresco, we can easily associate the users to the web project in Alfresco WCM. This is already described in *Chapter 3, Getting Started with Alfresco WCM*. You can refer to this chapter for further details.

Configuring LDAP for centralized identity management

LDAP evolved from X.500 OSI Directory Access Protocol. LDAP directory is the central authentication engine for the enterprise, and serves as the yellow pages for user access and profile information. The biggest advantage of LDAP is that your enterprise can access the LDAP directory from almost any computing platform, using any one of the increasing number of readily available LDAP-aware applications. In fact, LDAP is finding much wider industrial acceptance because of its status as an Internet standard.

You can use LDAP with any directory server, such as iPlanet, Novell's eDirectory, Microsoft's Active Directory, or OpenLDAP. If you are planning to implement an LDAP directory in your organization, you may consider OpenLDAP, Active Directory, or eDirectory. OpenLDAP is a stable and widely accepted open source directory server.

LDAP configuration with Active Directory

Active Directory supports LDAP-based authentication. It can also support authentication using JAAS+Kerberos and NTLM authentication. Only NTLM will give you a Single-Sign-On solution. It is possible to use any authentication methods against an Active Directory server and extract user and group information via LDAP.

For the LDAP to work with Alfresco, you have to make some changes in the configuration files.

Follow the steps given below to configure LDAP-based authentication with Active Directory.

1. Open the `<alfresco>/tomcat/webapps/alfresco/WEB-INF/classes/alfresco/subsystems/Authentication/ldap/ldap-authentication.properties` file and modify the properties to your required value as follows. All other properties can be kept as it is in the file.
 - `ldap.authentication.active=true` (this flag enables the LDAP as authentication mechanism; if set to false, LDAP will be used only for synchronization)
 - `ldap.authentication.userNameFormat=CN=%s,DC=company,DC=com` (this maps the user ID entered by the user to pass through LDAP; the %s is replaced with whatever the user types in as their user ID on the login screen)
 - `ldap.authentication.java.naming.provider.url=ldap://openldap.domain.com:389` (the name and port of your LDAP server; the standard port for LDAP is 389)
 - `ldap.authentication.java.naming.security.authentication=simple` (the authentication mechanism you want to use)
 - `ldap.authentication.defaultAdministratorUserNames=admin,User1` (LDAP users' names who should be considered as administrators, separated by a comma)

2. Open the `<alfresco>/tomcat/shared/classes/alfresco-global.properties` file and uncomment the following line:

```
authentication.chain=alfrescoNtlm1:alfrescoNtlm
```

To configure LDAP as an authentication mechanism, you need to change this property. Provide `ldap:ldap` for LDAP Authentication as follows:

```
authentication.chain=ldap:ldap
```

3. Open the `file-server-custom.xml` file. Add the following code:

```
<config evaluator="string-compare" condition="CIFS Server"
replace="true">
    <serverEnable enabled="false"/>
    <host name="${cifs.localname}A" domain="${cifs.domain}"/>
        <comment>Alfresco CIFS Server</comment>

        <!-- Set to the broadcast mask for the subnet -->
        <broadcast>${cifs.broadcast}</broadcast>

        <!-- Use Java socket based NetBIOS over TCP/IP and native SMB
            on linux -->
        <tcpipSMB platforms="linux,solaris,macosx"/>
        <netBIOSSMB platforms="linux,solaris,macosx"/>

        <!-- Can be mapped to non-privileged ports, then use firewall
            rules to forward requests from the standard ports -->
        <!--
        <tcpipSMB port="1445" platforms="linux,solaris,macosx"/>
        <netBIOSSMB sessionPort="1139" namePort="1137"
        datagramPort="1138" platforms="linux,solaris,macosx"/>
        -->

        <hostAnnounce interval="5"/>

        <!-- Use Win32 NetBIOS interface on Windows -->
        <Win32NetBIOS/>
        <Win32Announce interval="5"/>

        <!-- CIFS authentication -->
        <authenticator type="passthru">
            <LocalDomain/>
        </authenticator>

        <!--
        <WINS>
            <primary>1.2.3.4</primary>
```

```
<secondary>5.6.7.8</secondary>
</WINS>
-->
<sessionDebug flags="Negotiate,Socket"/>
</config>

<config evaluator="string-compare" condition="FTP Server"
replace="true">
<serverEnable enabled="false"/>
<!-- Run on a non-privileged port -->
<!--
<port>1121</port>
-->
<!-- FTP authentication -->
<authenticator type="alfresco"/>
<!--<debug flags="File,Search,Error,Directory,Info,DataPort"/>
-->
</config>

<config evaluator="string-compare" condition="Filesystem Security"
replace="true">
<authenticator type=" passthru ">
<!-- the name of your ldap server -->
<Server>openldap.domain.com</Server>
</authenticator>
</config>
```

This authentication mechanism sends usernames and passwords in plain text. It is the most simple to set up. This is supported by both Active Directory and OpenLDAP.

LDAP synchronization

As you have already configured LDAP with Active Directory, the next step will be to extract information from Active Directory. This synchronization of people and groups between the Alfresco repository and LDAP is supported by scheduled jobs. These jobs extract the user or group information from the LDAP repository and create the appropriate information as an Alfresco import XML file. This file is then imported into the repository.

Follow these steps to export users and groups from Active Directory:

1. Open the <alfresco>/tomcat/webapps/alfresco/WEB-INF/classes/alfresco/subsystems/Authentication/ldap/ldap-authentication.properties file. Modify the properties to the required value as follows. All other properties can be kept as is in the file:

- ldap.synchronization.java.naming.security.principal=admin

(the user that has read access to the group and people information to be extracted from Active Directory server)

- ldap.synchronization.java.naming.security.credentials=secret

(the password for the user defined above)

- ldap.synchronization.personQuery=(objectclass=inetOrgPerson)
 - ldap.synchronization.userSearchBase=dc=company,dc=com

(these two options combine to make the query to find people. In the previous example, you will find all objects of type `inetOrgPerson` anywhere in the directory)

- ldap.synchronization.groupQuery=(objectclass=groupOfNames)
 - ldap.synchronization.groupSearchBase=dc=example,dc=com

(these two options combine to make the query to find groups. In the previous example, you will find all objects of type `groupOfNames` anywhere in the directory)

2. Ensure that your earlier changes are saved. Start Alfresco. On restarting, you will be able to log into the Alfresco repository with LDAP users only.

Daisy Chaining

If you want to log into the Alfresco repository with Alfresco users also, then you have to make some more changes in the configuration files. This concept is called Daisy Chaining, allowing the users to configure multiple authentication components for Authentication.

With version 3.2, Alfresco has introduced the concept of sub-systems. A sub-system is a configurable module responsible for a subpart of Alfresco functionality. Authentication is one of such sub-systems available in 3.2, which is a stack of multiple components responsible for Authentication in Alfresco.

For more information on Alfresco Authentication sub-systems, you can refer to the Wiki link: http://wiki.alfresco.com/wiki/Alfresco_Authentication_Subsystems.

With this approach, Authentication sub-systems are easily chained. So, now it's very easy to configure the Chaining authentication. The steps to configure Chaining are as follows:

1. Open the <alfresco>/tomcat/shared/classes/alfresco-global.properties file and uncomment the following line:
authentication.chain=alfrescoNtlm1:alfrescoNtlm
2. To configure more than one authentication component for chaining, you can add as many authenticators as you want separated by a comma as mentioned below:

```
authentication.chain=ldap1:ldap,ldap2:ldap,  
alfrescoNtlm:alfrescoNtlm
```

Here we have three different authenticators for Chaining Authentication; ldap1 and ldap2 are for LDAP Authentication and alfrescoNtlm is for Default Alfresco Authentication. For authenticating the users, when a user logs into the system, Alfresco checks for authentication in the sequence as defined above.

3. Configuration files for ldap1 and ldap2 are created as follows:

```
Copy <alfresco>/tomcat/webapps/alfresco/WEB-INF/classes/  
alfresco/subsystems/Authentication/ldap/ldap-authentication.  
properties to <alfresco>/tomcat/shared/classes/alfresco/  
subsystems/Authentication/ldap1/ldap-authentication.properties.
```

```
Copy <alfresco>/tomcat/webapps/alfresco/WEB-INF/classes/  
alfresco/subsystems/Authentication/ldap/ldap-authentication.  
properties to <alfresco>/tomcat/shared/classes/alfresco/  
subsystems/Authentication/ldap2/ldap-authentication.properties.
```

4. Now, modify this property file for ldap1 and ldap2 accordingly to provide the LDAP configuration details in both the files. Configuring LDAP is already explained in the previous section.
5. Restart the server and you can verify the chaining by logging in with an Alfresco local user and the LDAP user as well.



Download the sample code from
the Packt website.



User roles

In the previous section, we discussed the different Authentication mechanisms for users in Alfresco. Once the user is part of Alfresco, either created in Alfresco or imported from LDAP or any other system, you can assign different roles to the user for providing specific permissions. The users and default roles to these users are controlled globally in Alfresco, which also affects the WCM.

By default, only the admin user has permission to create a web project in the **Company Home | Web Projects** folder. All the users of the system will have Consumer role by default for this folder. That means those users can have just "read" permission for the **Web Projects** folder; they really cannot create a web project. If you want to allow any other user to create a web project other than admin, then an admin can assign a Contributor or higher role to that user and then that user can get rights to create a new web project.

There are different roles available in Alfresco, which users can be assigned to. Following are the five basic roles:

- **Consumer:** Read only permission
- **Contributor:** Consumer + Permission to add/create new content
- **Editor:** Consumer + Permission to edit the existing content
- **Collaborator:** Contributor + Editor
- **Coordinator:** Collaborator + Permission to delete the content

As mentioned here, to create a web project, a user at least needs the Contributor role.

Another scenario is if you remove the default consumer roles available to all the users for the **Company Home | Web Projects** space, then even if the user is a member of some web project, he will not be able to see the Web Project folder. Therefore, the user will not have access to that web project of which he is also a member.

So, in this way you are controlling the users from outside the Web Content Management.

Common repository

As a Content Management Product, Alfresco has two different modules, DM and WCM.

In this section, we will see how we can use Alfresco with the combination of DM and WCM both as a single repository. Consider a use case of a company where they are using Alfresco WCM for managing their website but at the same time they have an Intranet application, which is more for managing documents and other assets. For this Intranet, they are using the Alfresco DM. Now suppose some of the documents/assets need to be managed in DM but are basically part of the website and need to be deployed. In this scenario, we can get the advantage of both the DM and WCM in the same repository.

Let's take the example of the Cignex company. As we have already seen in the earlier chapter, the website for Cignex is managed by Alfresco WCM. But in this website, some of the images, videos, and so on are being used, which are managed in DM. There are some departments such as Marketing, Sales, HR, IT, and many more available that are using this Intranet application. Now if a Marketing person is uploading a video, then he or she may have some requirements such as it should be transformed into different formats like 3GP (for cell phones), MPEG4 (for iPod), and Flash (the default format for the website). Alfresco can be integrated with FFMPEG, using which we can easily transform the videos to other formats.

Integrating Alfresco with the FFMPEG Video Transcoder

FFMPEG is a very popular high performance video and audio transcoder. It has various widely used commercial tools to convert audio and video files from one format to another. It is basically a command-line interface. We can easily integrate any such command-line applications with Alfresco.

You need to download the FFMPEG binary version for Microsoft Windows and put it into the <alfresco_install>/bin directory. The command used for transformation is:

```
ffmpeg.exe -i [input_file.extension] [options] [output_file.extension]
```

Various options for video transcoding

Following are the various options for video transcoding:

- **-b bitrate**
Set the video bitrate in bit/s (default = 200 kb/s).
=>512 k
- **-r fps**
Set frame rate (Hz value, fraction, or abbreviation), (default = 25).
=>15.02
- **-s size**
Set frame size. The format is 'wxh' (ffserver default = 160x128, ffmpeg default = same as source).
=>320x240
- **-aspect aspect**
Set aspect ratio (4:3, 16:9 or 1.3333, 1.7777).
=>4:3

Various options for audio transcoding

Following are the various options for audio transcoding:

- **-ar freq**
Set the audio sampling frequency (default = 44100 Hz).
=> 44100
- **-ab bitrate**
Set the audio bitrate in bit/s (default = 64 k).
=> 64 k
- **-ac channels**
Set the number of audio channels (default = 1 Mono; 2 = stereo).
=> 2

For further details and all options, you can refer to <http://www.ffmpeg.org/ffmpeg-doc.html>.

Integrating transformation as an Action in Alfresco

In this section, we will see how we can use FFMPEG transformations as a custom action in Alfresco. From there we can trigger some business rules on the spaces, which will execute this custom action to perform transformation of all video files. Follow these steps to execute the transformation:

1. We need to configure the extra mime types that we want to use. Here in our case, we are using 3GP, MP2, MP4, AVI, MOV, and many more. All others are already supported by Alfresco and are defined in the `mimetype-map.xml` file in Alfresco (except 3GP). So we need to add 3GP as a mimetype supported by Alfresco in the `mimetype-map` file, as follows:

```
<config evaluator="string-compare" condition="Mimetype Map">
    <mimetypes>
        <mimetype mimetype="video/3gp" display="Mobile video">
            <extension>3gp</extension>
        </mimetype>
    </mimetypes>
</config>
```

2. The next step is, as FFMPEG is a command-line interface, we can use Alfresco's `org.alfresco.util.exec.RuntimeExec` class to execute this command. For that we need to configure it as a bean, which will be referred by our custom action executer. We can make these bean entries in the `custom-services-context.xml` file.

```
<bean id="transformer.ffmpegVideo"
    class="com.cignex.alfresco.repo.content.transform.
        FfmpegVideoContentTransformer"
    parent="baseContentTransformer"
    init-method=>init</>
<property name=>executer<>
    <bean name=>transformer.ffmpegVideo.Command>
        class=>org.alfresco.util.exec.RuntimeExec</>
    <property name=>commandMap<>
        <map>
            <entry key=>Windows.*</>
                <value>ffmpeg -i <${source}> ${options}
                    <${target}></value>
            </entry>
        </map>
    </property>
    <property name=>defaultProperties<>
        <props>
            <prop key="options"></prop>
        </props>
    </property>
</bean>
```

```
        </property>
    </bean>
</property>
</bean>

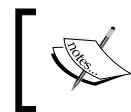
<bean id="transform-video"
class="com.cignex.alfresco.repo.action.executer.
    FfmpegVideoTransformActionExecuter"
    parent=>transform>>
    <property name=>videoContentTransformer>>
        <ref bean=>transformer.ffmpegVideo> />
    </property>
</bean>
```

3. In the web-client-config-custom.xml file, add the following:

```
<config evaluator="string-compare" condition="Action Wizards">
<video-transformers>
    <transformer name="video/x-msvideo"/>
    <transformer name="video/mp4"/>
    <transformer name="video/3gp"/>
    <transformer name="video/x-ms-wmv"/>
</video-transformers>
<action-handlers>
    <handler name="transform-video"
        class="com.cignex.alfresco.web.bean.actions.handlers.
            FfmpegTransformVideoHandler" />
</action-handlers>
</config>
```

4. Here you can specify all transformer names, and the formats you want to support in the transform action from the Alfresco UI as:

```
<transformer name="video/mp4"/>
```
5. You can add more if you want; here we have configured four different formats.
6. Once this is done, you need to create custom Action Executer classes and do the configurations for this.



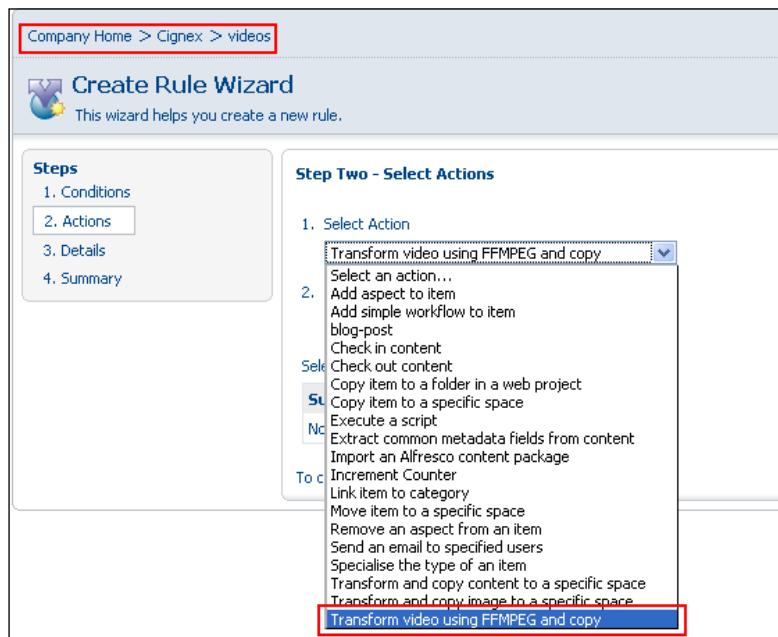
You can also refer to the following Wikis for more detail on these:
http://wiki.alfresco.com/wiki/Custom_Actions
http://wiki.alfresco.com/wiki/Wizard_Framework

7. After finishing creation and configuration of the Action Executer, you are ready with the example code. Now start the Alfresco server and we will see how you can configure the rule for transformation.
8. We will create the folder structure such as **Company Home | Cignex | videos**. Inside the videos space, create spaces for all different types of formats (for video) that you want to support (that is, MPEG4, 3GP, and so on).

Configuring FFMPEG transformation as a business rule

Alfresco DM provides the business rule; you can configure a business rule for a particular type of content or for all the content in a particular space. Here we will configure a business rule on the **videos** space to transform the videos we upload in this space to convert into other formats.

1. Browse to **Company Home | Cignex | videos** and click on **More Actions | Manage Content Rules**. On the next screen, click on the **Create Rule** link. In the first step of the **Create Rule Wizard**, select **All Items** and click on **Add to List**. Click on **Next**.
2. In the next step, you can see the action created by us in the **Select Action** combo box. Select the action **Transform video using FFMPEG and copy**, as shown in the following screenshot. Click on **Next**:



3. In the next step, you need to select the proper value for the required format (the target format for transformation). As we have configured 3GP, MP4, AVI, and WMV in the configuration file, we will have all these four options available to us for transforming the video. After that, provide the proper options for a particular video transformation selection based on source- and target-type of video file. Also, you need to provide the destination space to copy the transformed video. Here we have created the **3GP** space inside the videos folder, so provide it as a destination folder. Click on **OK**.

Company Home > Cignex > videos

Create Rule Wizard
This wizard helps you create a new rule.

Set action values

Required format: 3gp

Options

Destination: [3GP]

OK Cancel

4. In the next step, choose **Type** as **Inbound**; also provide **Title** and **Description**. Check the option **Apply rule to sub spaces** as shown in the following screenshot. Click on **Finish**:

Company Home > Cignex > videos

Create Rule Wizard
This wizard helps you create a new rule.

Steps
1. Conditions
2. Actions
3. Details
4. Summary

Step Three - Enter Details

Type: Inbound

Title: Transform video *

Description: Transforms video in 3GP Format

Other Options

Apply rule to sub spaces

Run rule in background

If this option is selected the rule will execute in the background so the results may not appear immediately.

Disable rule

Next Back Finish Cancel

Now whenever you upload any video in **videos** space, it will be transformed into the 3GP video format. It will also be copied to the **3GP** space inside the videos space as shown in the next screenshot:

You can create as many rules and transformations as you want to perform for different video types.

[ Download the sample code files from the Packt website.]

Copying videos from DM to WCM

In this section, we will discuss the different ways to copy the content—here videos from DM to WCM. One is with the help of business rule and the other one is using JavaScript. Both of these are described in the following sections.

DM to WCM using business rule

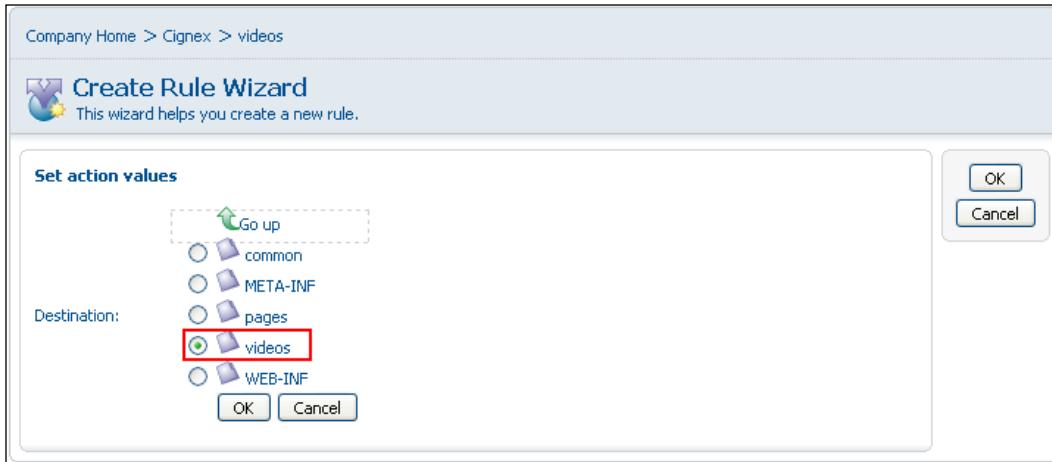
We have videos available in all required formats under **Company Home | Cignex | videos** and the respective folders for the video formats. Now we need to move these videos in WCM in the proper web project and to the proper folder.

There is one more rule available in DM, which will copy the DM content to WCM. For this, you need to configure that rule. Use the following steps:

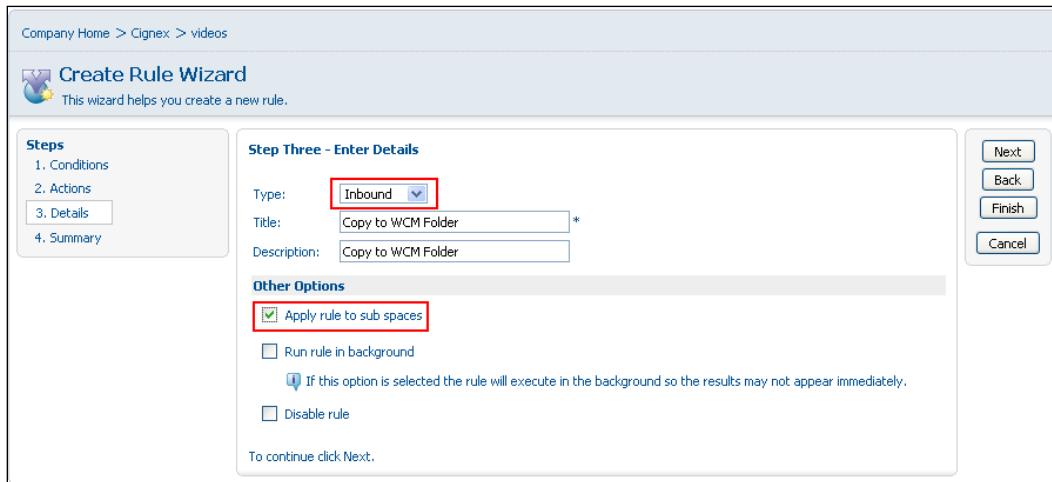
1. Browse to **Company Home | Cignex | videos** and click on **More Actions | Manage Content Rules**. In the next screen, click on the **Create Rule** link. In the first step of the Create Rule Wizard, select **All Items** and click on **Add to List**. Click on **Next**.
2. In the next step, you can see the action created by us in the **Select Action** combo box. Select the **Copy item to a folder in a web project** action, as shown in the following screenshot. Click on **Next**:

The screenshot shows the 'Create Rule Wizard' interface. The title bar says 'Company Home > Cignex > videos'. The main area is titled 'Create Rule Wizard' with the sub-instruction 'This wizard helps you create a new rule.' Below this is a 'Steps' sidebar with four options: 1. Conditions, 2. Actions (which is selected and highlighted in blue), 3. Details, and 4. Summary. The main content area is titled 'Step Two - Select Actions'. It contains a section '1. Select Action' with a dropdown menu showing 'Select an action...'. A list of actions follows, with 'Copy item to a folder in a web project' highlighted with a red rectangle. Other actions listed include: Add aspect to item, Add simple workflow to item, blog-post, Check in content, Copy item to a specific space, Execute a script, Extract common metadata fields from content, Import an Alfresco content package, Increment Counter, Link item to category, Move item to a specific space, Remove an aspect from an item, Send an email to specified users, Specialise the type of an item, Transform and copy content to a specific space, Transform and copy image to a specific space, and Transform video using FFMPEG and copy.

- In the next screen, select the web project and the specific folder in that web project as **Destination** folder and click on **OK** as shown in the following screenshot:



- Then click on **Next**. In the next screen, choose **Type** as **Inbound** and provide the **Title** and the **Description**. Also select **Apply rule to sub spaces** as shown in the next screenshot. Click on **Finish**:



DM to WCM using JavaScript

There is one API available in Alfresco JavaScript, which allows you to copy content from one repository to another. Using this API, you can copy the content from DM to WCM. Here we will copy all the videos, which we transformed previously to a specific folder in the web project.

Alfresco JavaScript has `crossRepoCopy` as one of the root objects, using which we can copy nodes or content between the different repositories (Document Management and Web Content Management).

`copy(ScriptNode source, ScriptNode destination, String name)`: This will copy a source node to the specified destination node and return the new node. In the parameter name, you can specify the new name for that content, which is being copied to some other location.

Using this API, we can copy all of the transformed video from the DM repository to the WCM repository, as follows:

1. Create new JavaScript file, `copy_wcm.js`, with the following content:

```
logger.log("Document :: " + document.name);

var l_cignex_root=avm.lookupStore("wwwcignex--admin");

l_video_path="/ROOT/videos";

if(l_cignex_root != null) {
    l_videoNode = l_cignex_root.lookupNode(l_video_path + "/");
    if(l_videoNode != null){
        crossRepoCopy.copy(document,l_videoNode,document.name);
        logger.log(document.name + " is copied");
    }
}
```

2. Upload this JavaScript file to the **Company Home | Data Dictionary | Scripts** space in Alfresco.

The screenshot shows the 'Scripts' view in the Alfresco Data Dictionary. At the top, there's a breadcrumb navigation: Company Home > Data Dictionary > Scripts. Below the header, there's a brief description: 'This view allows you to browse the items in this space. JavaScript files'. On the right side of the header, there are buttons for 'Add Content', 'Create', 'More Actions', and 'Icon View'. Under the 'Browse Spaces' section, it says 'No items to display. Click the 'Create Space' action to create a space.' There are buttons for 'Page' and 'Items Per Page' (set to 9). The main area is titled 'Content Items' with another 'Items Per Page' button (set to 4). It lists several scripts: 'command-processor.js', 'command-search.js', 'command-utils.js', and 'copy_wcm.js'. The 'copy_wcm.js' item is highlighted with a red box. Below each script listing are edit, delete, and other action icons. At the bottom, there's another 'Page' and 'Items Per Page' section.

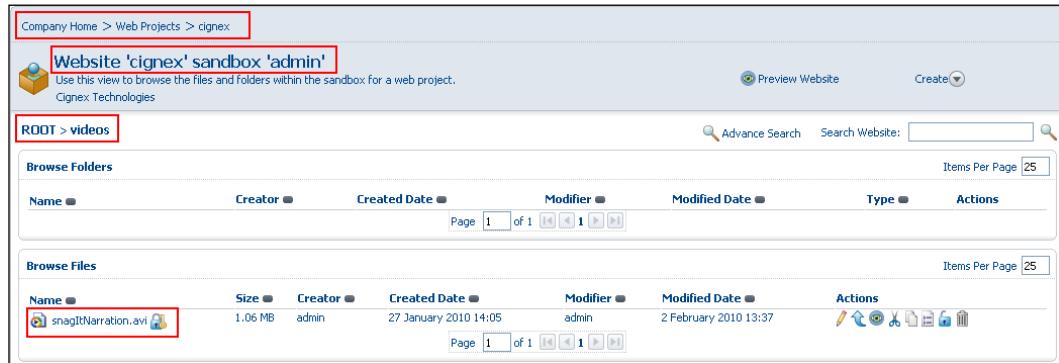
- Now, with the help of **Run Action**, you can execute this script. Go to the **View Details** page of the video content, which you want to copy to the WCM. Click on **Run Action** from the available actions, as shown in the following screenshot:

The screenshot shows the 'View Details' page for a video file named 'snagItNarration.avi'. The left side has sections for 'Custom View', 'Links' (with options like 'View In Browser', 'Download Content', 'View in WebDAV', 'Details Page URL', and 'Alfresco Node Reference'), and 'Properties' (listing details such as Name, Content Type, Encoding, Title, Description, Author, Size, Creator, Created Date, Modifier, Modified Date, Email ID, and Tags). The right side has a 'Actions' panel with various options: Edit Offline, Check Out, Download, Update, Cut, Copy, Delete, Take Ownership, Manage Content Users, Create Shortcut, Preview in Template, Run Action (highlighted with a red box), Make Multilingual, Start Discussion, and Start Advanced Workflow.

4. Choose **Execute a script** from the available list of actions and click on the **Set Values and Add** button. Select `copy_wcm.js` from the list of available JavaScript files to execute and click on **OK**.



5. Click on **Finish**. This will execute the `copy_wcm.js` JavaScript and copy the content to the WCM in a specified web project and a specified folder, as shown in the following screenshot:



You can also configure this 'Execute Script' as a rule on the videos folder. For configuring it as a business rule, you can follow the same steps as mentioned in the *DM to WCM Using Business Rule* section. Just choose the **Execute a Script** action during configuration and `copy_wcm.js` as a value.

If you configure this rule for Inbound Documents, it will automatically copy the videos file to the corresponding destination folder in WCM.

Image transformation in WCM

Alfresco leverages the power of ImageMagick for image transformations. Refer to *Chapter 2, Installation and Configuration* for installing ImageMagick. Alfresco JavaScript provides some APIs to perform image transformations. The prerequisite for this is to have ImageMagick installed. The Script node has the APIs mentioned in the following section for image transformation.

Image transformation APIs

Alfresco provides the following APIs for image transformations. `transformImage` is overloaded with different parameters. You can use any of these based on your requirement:

- `transformImage(string mimetype)`: This will transform an image to a new type specified in `mimetype` and will return the transformed image node if successful or null if failed.
- `transformImage(string mimetype, string options)`: This will transform an image to a new type specified in `mimetype` by applying the supplied ImageMagick options and will return the transformed image node if successful or null if failed.
- `transformImage(string mimetype, ScriptNode destination)`: This will transform an image to a new type specified in `mimetype` and will return the transformed image node if successful or null if failed. A new image document will be created in specified destination folder.
- `transformImage(string mimetype, string options, ScriptNode destination)`: This will transform an image to a new type specified in `mimetype` by applying the supplied ImageMagick options and will return the transformed image node if successful or null if failed. A new image document will be created in the specified destination folder.

Configuring new action for image transformation in WCM

Following are the steps to configure image transformation as an action in Alfresco.

1. Configure action for transform image in `web-client-config-custom.xml` file as:

```
<config>
<actions>
    <!-- Transform Image Action -->
    <action id="transform_image">
        <permissions>
            <permission allow="true">Write</permission>
        </permissions>
        <evaluator>org.alfresco.web.action.evaluator.
            WCMWorkflowEvaluator</evaluator>
        <label-id>title_action_transform_image</label-id>
        <image>/images/icons/action.gif</image>
        <action>browseSandbox</action>
```

```
<script>/Company Home/
    Data Dictionary/Scripts/transform_image.js</script>
<params>
    <param name="id">#{actionContext.id}</param>
</params>
<target>new</target>
</action>
</actions>
</config>
```

2. Add this action in the action group for `avm_file_browse` in the `web-client-config-custom.xml` file, as follows:

```
<config>
<actions>
    <action-group id="avm_file_browse" replace="true">
        <show-link>false</show-link>
        <action idref="edit_file" />
        <action idref="update_file" />
        <action idref="preview_file" />
        <action idref="cut_avm_node" />
        <action idref="copy_avm_node" />
        <action idref="file_details" />
        <action idref="unlock_file" />
        <action idref="delete_file_browse" />
        <action idref="tranform_image" />
    </action-group>
</actions>
</config>
```

3. Now we need to create JavaScript file, which will be executed when this action is performed. This JavaScript will be responsible for transforming the image. The JavaScript `transform_image.js` will look as follows:

```
logger.log("called :: " + args["id"]);
var path = args["id"];
var node = avm.lookupNode(path)
logger.log("node : " + node);
var thumbImage = node.transformImage("image/png", "-resize
120", node.parent);
```

4. Here, you can specify any image type for transformation instead of `image/png` and also can specify the required options as the second argument in `transformImage` API.

- Upload this JavaScript in the **Company Home | Data Dictionary | Scripts** folder as shown in the following screenshot:

The screenshot shows a web interface for managing scripts. At the top, there's a breadcrumb navigation: 'Company Home > Data Dictionary > Scripts'. Below it is a header with a 'Scripts' icon and a note: 'This view allows you to browse the items in this space. JavaScript files'. On the right, there are buttons for 'Add Content', 'Create', 'More Actions', and 'Icon View'. A dropdown menu 'Items Per Page' is set to 9. The main area has sections for 'Browse Spaces' (empty) and 'Content Items'. Under 'Content Items', there are two entries: 'test return value.js' and 'transform_image.js'. The 'transform_image.js' entry is highlighted with a red box. It includes details like '0.31 KB', '4 February 2010 12:31', and a set of small icons for edit, delete, etc. Navigation buttons at the bottom show 'Page 1 of 1' and 'Page 2 of 2'.

- Now the new action for Image Transformation will be available for any content in WCM.

Using image transformation action in WCM

In this section, we will see how we can use this action, which we configured in the previous section.

- Go to the Cignex web project and click on **Browse Website**. Browse to the **ROOT | images** folder. You can see the images available there as shown in the following screenshot:

The screenshot shows a 'Browse Website' interface for the 'ROOT > images' folder. At the top, there's a breadcrumb navigation: 'ROOT > images'. Below it is a header with 'Advance Search', 'Search Website', and a dropdown for 'Items Per Page' set to 25. The main area has sections for 'Browse Folders' and 'Browse Files'. In the 'Browse Files' section, there is one item: 'Expand-snap.JPG'. The 'Actions' column for this file is highlighted with a red box and annotated with a callout bubble 'Transform Image Action'. Navigation buttons at the bottom show 'Page 1 of 1'.

2. Currently, we have the Expand-snap.JPG image file available. We will apply the **Transform Image** action on this and we will have the same image transformed into PNG as Expand-snap.png file with resizing.

Name	Creator	Created Date	Modifier	Modified Date	Type	Actions
Expand-snap.JPG	admin	2 February 2010 13:46	admin	2 February 2010 13:46	JPG	
Expand-snap.png	admin	4 February 2010 13:47	admin	4 February 2010 13:47	PNG	



Download the sample code files from
the Packt website.



Advanced search in WCM

The success of content management systems depends on its ability to locate the required content with fewer clicks. You will realize the benefits of having a powerful search engine when you have a large amount of content in your content management system.

Unlike many commercial content management systems, Alfresco includes a free and very powerful search engine called Lucene as part of installation. Hence you don't have to buy and install a third-party search engine. And moreover, you don't have to deal with integration issues and upgrades.

In Alfresco DM, you will be able to search both content and properties. You can do a full-text search on any word in content, regardless of the format. You can search for content in a particular space. You can also search for content belonging to certain categories or of a specific type. We will leverage this searching capability of Alfresco DM in WCM as well with the help of the Lucene search engine. In this section, we will see how we can use Lucene to search the content stored in Alfresco WCM and easily do a full-text search on any word in content, regardless of the format.

Searching in WCM is similar to the searching in DM, but in WCM, Lucene-based search is only possible in the Staging Sandbox. It is not possible for any User's Sandbox or also the Workflow Sandbox. XPath-based searching is possible for all the WCM stores including User's Sandbox and Workflow Sandbox. However, the drawback of XPath search is that the performance might slow down depending on the query and the store structure, as the implementation walks the object model.

WCM search can be performed via Java, JavaScript, FreeMarker template, or the Node Browser.

For more information on search, refer to <http://wiki.alfresco.com/wiki/Search>.

Using JavaScript

Using JavaScript API for search, you can search the content against WCM stores. Alfresco provides search as a root object for JavaScript, which provides access to Lucene search. The available APIs to perform search are:

- `luceneSearch(string query)`: This will perform a full-text Lucene search with the provided query and return the search result as an array of Script node objects.
- `luceneSearch(string store, string query)`: This will perform a full-text Lucene search in the specified store with the provided query and return the search result as an array of Script node objects.
- `luceneSearch(string query, string sortColumn, boolean asc)`: This will perform a full-text Lucene search with the provided query and return the sorted search result based on the column specified and the sorting order as an array of Script node objects.
- `luceneSearch(string store, string query, string sortColumn, boolean asc)`: This will perform a full-text Lucene search in the specified store with the provided query and return the sorted search result based on the column specified and the sorting order as an array of Script node objects.
- `xpathSearch(string xpath)`: This will perform an Alfresco XPath search and return the search result as an array of Script node objects.
- `xpathSearch(string store, string xpath)`: This will perform an Alfresco XPath search in the specified store and return the search result as an array of Script node objects.
- `ScriptNode findNode(NodeRef noderef)`: This will return a Script node for the specified noderef.
- `ScriptNode findNode(string noderef)`: This will return a Script node for the specified noderef in the form of String.

Here for the Lucene query, you can pass any valid Lucene query as an argument. While building the Lucene query, you can use multiple query criteria with combination of relational operator. For example, to search text "cignex", you can have the query as TEXT:cignex, as follows:

```
var results = search.luceneSearch("avm://wwwcignex", "TEXT:cignex");
```

This will search for the text "cignex" in wwwcignex store (Staging Sandbox of the web project named wwwcignex).

Now if you want to restrict the search to some specific folder within the store, you can have path criteria in the query as PATH: "www/avm_webapps/ROOT/common/*", as follows:

```
var results = search.luceneSearch("avm://wwwcignex", "TEXT:cignex" AND  
PATH:\\"www/avm_webapps/ROOT/common/*\"");
```

AVM API to search in WCM store

Alfresco provides search APIs to perform search in the Alfresco Repository. The following is the API used to perform Lucene search in WCM.

- `store.luceneSearch(query)`: This will perform a Lucene search with the provided query against the store and return the search result as an array of AVM nodes.

To perform the search on a particular store, you need to get the store first and then you can execute search against that store, that is:

```
var store = avm.lookupStore("wwwcignex");  
var results = store.luceneSearch("wca\\:parentformname:news");
```

Using FreeMarker template

Similar to JavaScript, FreeMarker template language also provides the API to perform the search against the WCM stores; the API is:

- `store.luceneSearch(query)`: This will perform a Lucene search with the provided query against the store and return the search result as an array of AVM nodes.

For example, to search all the folders in the web project, you can have the query as follows:

```
<#assign store = avm.lookupStore("wwwcignex")>  
<list store.luceneSearch(TYPE:"cm:folder") as l_search_result>  
  <li> ${l_search_result.properties.name} </li>  
</list>
```

Using the Node browser

You can use the Node browser available in Alfresco Explorer to search the content in the WCM stores. But only Admin users can have access to Node Browser. In order to use the Node browser:

1. Click on the **Administration Console** button () on the top menu bar.
2. Click on the **Node Browser** option on the Administration console screen.
3. It will have a list of all the stores of repository. Select the appropriate WCM store in which you want to perform the search.
4. In the combo box for **Search**, select the Search Language, that is, **lucene** or **xpath**, and so on.
5. In the **Search** text area, provide the search query for the specific language.
6. Click on **Search**. It will perform the search and return the result nodes.



Using Java

Java also provides Search Service to perform the search operations. The APIs available in search service to execute search are as follows:

- `query(Storeref store, String language, String query)`: This will search against the specified store in WCM with the given query and language.
- `query(Storeref store, String language, String query, QueryParameterDefinition[] queryParameterDefinition)`: This will search against the specified store in WCM with the given query language and Query Parameters. Here you can have a parameterized query.

- `query(Storeref store, QName queryId, QueryParameter[] queryParameters)`: This will execute a canned query against the specified store with the given query identifier and query parameters.
- `query(SearchParameters searchParameters)`: This will perform search using the specified Search Parameters.

All of the previous APIs for search using search service available in Alfresco will return Search results as a ResultSet. You can iterate this result set and perform any further processing.

Case study: User Interface for Advanced Search in WCM

In this section, we will discuss the case study of UI for Advance Search in WCM. Here we will provide different criteria for search, such as choosing the web form of which we want to search the content and the facility to choose the path in the ROOT folder to search the content. For this we will use Lucene search so that it will search only in a Staging Sandbox.

This search example is for free text-based search and does not provide the capability to search the content based on its property. We will see the example of searching content based on the content property in the next section.

To implement this functionality:

1. Provide a link for Advance Search in the `browse-sandbox.jsp` page as:

```
<div style="float:right;padding-right:20px;padding-top:3px">
<a:actionLink id="advance-search-image" value="Advance
Search" image="/images/icons/search_icon.gif" showLink="false"
showLink="false" action="advanceSearch" />
<a:actionLink id="advance-search-apply" value="Advance Search"
showLink="false" action="advanceSearch" />
</div>
```

2. Create a new JSP file in the `jsp/extension/wcm/search` folder for rendering the Search UI Page.

3. Add a managed bean definition in the `faces-config-custom.xml` file. This bean is going to handle search parameter retrieval process for Advanced Search JSP.

```
<managed-bean>
    <description>
        The bean that holds a state for the Advanced Search screen.
    </description>
    <managed-bean-name>CustomAdvancedSearchBean</managed-bean-name>
    <managed-bean-class>com.cignex.web.bean.wcm.CustomAdvancedSearchBean
    </managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
    <managed-property>
        <property-name>nodeService</property-name>
        <value>#{NodeService}</value>
    </managed-property>

    <managed-property>
        <property-name>searchService</property-name>
        <value>#{SearchService}</value>
    </managed-property>
    <managed-property>
        <property-name>serviceRegistry</property-name>
        <value>#{ServiceRegistry}</value>
    </managed-property>
</managed-bean>
```

4. Implement class `com.cignex.web.bean.wcm.CustomAdvancedSearchBean` and implement a method [`public Search ()`] that will perform the search with the specified parameters. This class will also have corresponding fields for all search components.
5. Create a Java class, `CustomAdvanceSearchBean`, with the specified package structure, which will actually perform the search in the repository.



Download the sample code files from
the Packt website.

Search text in ROOT folder and for all web forms

This will search in all of the sub-folders of **ROOT** in the current web project and will also search for all the web forms. The **Reset All** button will reset all search criteria. Search will return the search results with the filename and corresponding path. The following screenshot shows a snap of the search screen, which searches for the text **Alfresco** in the current web project's Staging Sandbox.

The screenshot shows the 'Advanced Search' interface with the following configuration:

- Text to search for:** alfresco
- Select WebForm:** -Select-
- Select Folder:** + ROOT
- Selected Folder:** (empty)

The search results table has two columns: **FileName** and **FilePath**. The results are:

FileName	FilePath
alfresco_3_book_news.html	/ROOT/common/news/alfresco_3_book_news.html
web.xml	/ROOT/WEB-INF/web.xml
alfresco_3_book_news.xml	/ROOT/common/news/alfresco_3_book_news.xml
blog1.xml	/ROOT/common/news/blog/blog1.xml
training_news.xml	/ROOT/common/news/training_news.xml
blog1.html	/ROOT/common/news/blog/blog1.html
training_news.html	/ROOT/common/news/training_news.html
log4j.properties	/ROOT/WEB-INF/lib/log4j.properties
cignex_news.xml	/ROOT/common/news/cignex_news.xml
alfresco_training.xml	/ROOT/pages/technology/alfresco_training.xml
training.xml	/ROOT/common/news/training.xml
alfresco_training.html	/ROOT/pages/technology/alfresco_training.html
training.html	/ROOT/common/news/training.html
left_navigation.jsp	/ROOT/common/inc/left_navigation.jsp
training.jsp	/ROOT/pages/services/training.jsp
index.jsp	/ROOT/index.jsp
index.jsp	/ROOT/pages/technology/index.jsp
imenu0.jsp	/ROOT/common/inc/imenu0.jsp

Search text in particular web form content

This will search the text in all the sub-folders of **ROOT** in the current web project and will also search in a specific, selected web form as shown in the following screenshot:

The screenshot shows the 'Advanced Search' interface with the following configuration:

- Text to search for:** alfresco
- Select WebForm:** news
- Select Folder:** + ROOT
- Selected Folder:** (empty)

The search results table has two columns: **FileName** and **FilePath**. The results are:

FileName	FilePath
alfresco_3_book_news.html	/ROOT/common/news/alfresco_3_book_news.html
alfresco_3_book_news.xml	/ROOT/common/news/alfresco_3_book_news.xml
training_news.xml	/ROOT/common/news/training_news.xml
training_news.html	/ROOT/common/news/training_news.html
cignex_news.xml	/ROOT/common/news/cignex_news.xml

Search text in particular folder of web project

This will search the specified text in the particular selected folder within the web project. The Folder Tree component allows you to choose the folder in which you want to search the text. The following screenshot shows the Folder Tree component and the search result:

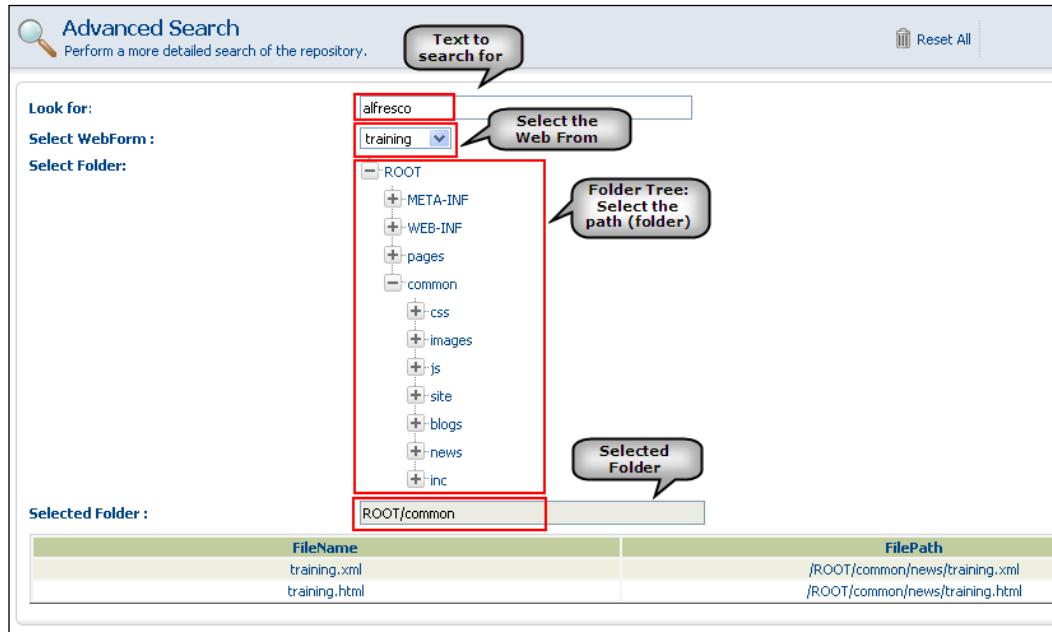
The screenshot shows the 'Advanced Search' interface with the following components and annotations:

- Text to search for:** A text input field containing "alfresco".
- Select WebForm :** A dropdown menu showing "--Select--".
- Select Folder:** A folder tree component with a red border around it. It shows the structure: ROOT > META-INF > WEB-INF > pages > common > css > images > js > site > blogs > news > inc. A callout bubble says "Folder Tree: Select the path (folder) to search".
- Selected Folder :** A text input field containing "ROOT/common" with a red border around it. A callout bubble says "Selected Folder".
- Table Results:** A table with two columns: "FileName" and "FilePath". The table lists the following entries:

FileName	FilePath
alfresco_3_book_news.html	/ROOT/common/news/alfresco_3_book_news.html
alfresco_3_book_news.xml	/ROOT/common/news/alfresco_3_book_news.xml
blog1.xml	/ROOT/common/blogs/blog1.xml
training_news.xml	/ROOT/common/news/training_news.xml
blog1.html	/ROOT/common/blogs/blog1.html
training_news.html	/ROOT/common/news/training_news.html
cignex_news.xml	/ROOT/common/news/cignex_news.xml
training.xml	/ROOT/common/news/training.xml
training.html	/ROOT/common/news/training.html
left_navigation.jsp	/ROOT/common/inc/left_navigation.jsp
imenus0.jsp	/ROOT/common/inc/imenus0.jsp

Search text in particular web form and in particular folder of web project

This will search the specified text in a particular folder selected with the help of the Folder Tree component and also search the content created using the specified web form. The **Select WebForm** combo box will fetch all web forms associated with the current web project:



Metadata extraction for WCM

The case study we discussed in the previous section was for the free text search with advanced search criteria in the WCM. In this section, we will discuss how we can perform the search based on the metadata of particular content. Alfresco provides some of the metadata extractors, which are responsible for extracting the metadata from the added or updated content at the server side. Alfresco has different extractors for handling different types of mimetypes for DM.

In this section, we will see how we can configure XML Metadata Extractor for WCM. We will take the example of news content, which we are creating with the help of a news web form.

1. To Activate metadata extraction for WCM, rename the `wcm-xml-metadata-extracter-context.xml.sample` file available in the `<alf_home>/tomcat/shared/classes/alfresco/extension` folder to `wcm-xml-metadata-extracter-context.xml`. This file will have bean declaration for WCM Metadata extraction, as follows:

```
<bean id="avmMetadataExtractorRegistry"
      class="org.alfresco.repo.content.metadata.
      MetadataExtractorRegistry" />
      ...
      <property name="invokePolicies">
          <value>true</value>
      </property>
      ...
      <property name="metadataExtractorRegistry">
          <ref bean="avmMetadataExtractorRegistry" />
      </property>
      ...
</bean>
```

2. The AVMNodeService fires policies for content creation and update that are listened to by `avmMetadataExtractors`, which is responsible for initiating the metadata extraction for content newly created or updated.
3. To configure an extractor to handle XML content, first we need to configure `XpathMetadataExtractor`, as follows:

```
<property name="xpathMappingProperties">
    <bean class="org.springframework.beans.factory.config.
    PropertiesFactoryBean">
        <property name="location">
            <value>
                classpath:alfresco/extension/News-Xpath-Mappings.
                properties
            </value>
        </property>
    </bean>
</property>
```

Here, News-Xpath-Mappings.properties stored in the alfresco/extension folder will have mappings of XPath as follows:

```
# Name Space  
namespace.prefix.trn=http://www.alfrescobook.com/webforms  
  
# XPATH Mappings  
short_title=news/shortTitle/text()  
content_header=news/contentHeader/text()  
content_sub_header=news/contentSubHeader/text()  
news_date=news/newsDate/text()
```

4. Now we need to map this property with the properties of a content model; for that again, we need to configure property as follows:

```
<property name="mappingProperties">  
    <bean class="org.springframework.beans.factory.config.  
        PropertiesFactoryBean">  
        <property name="location">  
            <value>classpath:alfresco/extension/Content-Model-Mappings.  
                properties  
            </value>  
        </property>  
    </bean>  
</property>
```

5. Here Content-Model-Mappings.properties stored in the alfresco/extension folder will have a mapping of properties of the content model as:

```
namespace.prefix.cignex=http://www.cignex.com
```

```
# Mappings  
short_title=shortTitle  
content_header=contentHeader  
content_sub_header=contentSubHeader  
news_date=newsDate
```

6. Now we need to select the extractor as XMLMetadataExtractor and configure bean for this as:

```
<bean id="extracter.xml.sample.XMLMetadataExtractor"  
    class="org.alfresco.repo.content.metadata.xml.  
XMLMetadataExtractor"  
    parent="baseMetadataExtractor">  
    <property name="registry">  
        <ref bean="avmMetadataExtractorRegistry" />  
    </property>  
    <property name="overwritePolicy">  
        <value>EAGER</value>  
    </property>  
    <property name="selectors">  
        <list>
```

```

        <ref bean="extracter.xml.sample.selector.XPathSelector" />
    </list>
</property>
</bean>

```

7. We also need to create a custom content model to define our custom properties. In this example, we are considering news as one content type, so we need to create a custom aspect with the properties of news as:

```

<aspect name="cignex:news">
    <title>Cignex News</title>
    <properties>
        <property name="cignex:shortTitle">
            <title>Short Title</title>
            <type>d:text</type>
            <index enabled="true"/>
        </property>

        <property name="cignex:contentHeader">
            <title>Content Header</title>
            <type>d:text</type>
            <index enabled="true"/>
        </property>
        <property name="cignex:contentSubHeader">
            <title>Content Sub Header</title>
            <type>d:text</type>
            <index enabled="true"/>
        </property>
        <property name="cignex:newsDate">
            <title>News Date</title>
            <type>d:date</type>
            <index enabled="true"/>
        </property>
    </properties>
</aspect>

```

8. Now we can search based on any custom property (metadata). You can add search criteria as:

`@cignex\ :shortTitle:"CignexNews",@cignex\ :contentHeader:"Cignex*"`

9. You can use any of the previously mentioned approaches using Java or JavaScript API, among others, to perform the search by providing this criteria.



Download the sample code files for the WCM metadata extractor from the Packt website.



Summary

As Alfresco itself provides many services, using WCM as part of Alfresco we can leverage all these services and capabilities of Alfresco DM in WCM and can use WCM as an enhanced repository. In this chapter, we learned the following points:

- Leveraging Membership and Security Mechanism along with the LDAP example
- FFmpeg Integration with Alfresco and using DM Business Rules
- Copying content from DM to WCM
- Using image transformation in WCM
- Leveraging Advance Search with custom UI
- Using WCM Metadata Extractor to allow metadata search in WCM

In the next chapter, we will discuss the Administrative tasks to maintain and upgrade the system.

12

WCM Administration

Maintaining and upgrading the system is as equally important as implementing it. A well-maintained system will give the highest return on investments. This chapter provides a high-level overview of administering and maintaining your Alfresco implementation. It includes information about backing up your valuable content, upgrading your system to newer versions, enabling full audit trail, and setting up a multi-tenant system configuration. You will also find general maintenance tips such as maintaining log files and periodically updating your admin password.

By the end of this chapter you will have learned how to:

- Back up your data on a regular basis for storage and retrieval
- Perform general maintenance tasks such as examining log files
- Upgrade your Alfresco application to newer versions
- Provide Administrator rights
- Clean up deployment history

Data backup

This is one of the most important, yet also one of the most neglected, areas of computing. Backing up your data should be at the top of your computer maintenance list—right next to virus protection. Without data backup, you are running the risk of losing your data.

Data loss can happen in many ways. One of the most common causes is the physical failure of the media that the data is stored on. In some situations, users of the system might have deleted the content due to some error. No matter what, your data is your intellectual property and you need to protect it by doing proper backups regularly.

List of items to back up

Alfresco stores the content information in both the database and the filesystem. You need to back up both the filesystem and relational database. As a part of the implementation, you might have customized Alfresco, and hence you need to back up customization files. If you have used an external membership system such as Active Directory or OpenLDAP, then you might have to back up the user and group data as well.

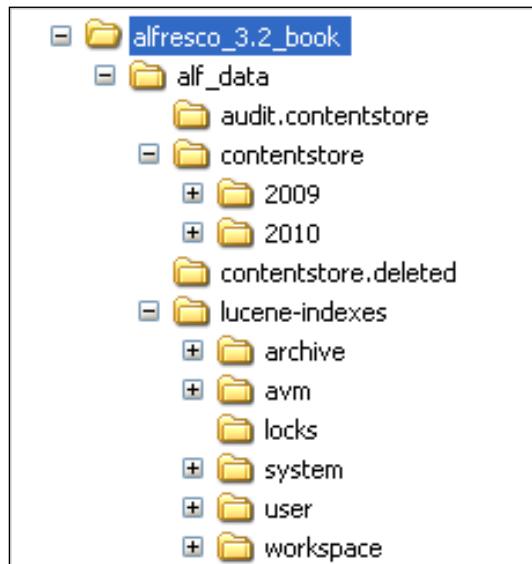
You can set up automated processes to back up data periodically. On Linux operating systems, you can write a cron job to run a back-up script on a regular basis. Similarly, all other operating systems support back-up utilities.

Most often, people tend to store the back-up data on the same server. This might create issues when the server crashes. Hence, it is recommended to move the back-up data on to some other external server to store it.

Now, let's examine various types of data which need to be backed up.

Content stored in filesystem

Typically, the content in the filesystem is stored in your `<install_folder>/alf_data` folder as shown in the following screenshot. The folder `contentstore` contains the binary content with all of the versions. The folders `lucene-indexes` and `backup-lucene-indexes` contain search information. The folder `audit.contentstore` contains audit trail details:



You need to back up the entire folder `alf_data` and the contents. Most of the operating systems provide back-up utilities.

If you are a Windows user, you may use the back-up utility that comes with Microsoft Windows XP (it is installed by default with the XP Home version). You will find it from the **Start** menu under **All Programs | Accessories | System Tools | Backup**. When you start it, you are presented with the back-up wizard.

Metadata stored in a relational database

A relational database contains a bunch of tables defined as per the Alfresco schema. These tables hold information about users, security, audit, spaces, metadata, rules, scripts, and various business processes (jBPM).

Most of the database vendors (commercial or open source) provide utilities to take the database dump. Based on the database you have selected during installation (MySQL, Oracle, or MS SQL Server), you can use an appropriate utility to take the database dump.

The MySQL database provides a utility called `mysqldump` to back up both the database table definitions and contents. It can be used to dump a database or a collection of databases for backup or for transferring the data to another SQL server (not necessarily a MySQL server). The dump contains SQL statements to create the table or populate it, or both.

The following is the command to take database backup in MySQL:

Syntax: `mysqldump [options] db_name [tables] [> output_file_name]`

An example command is:

```
> mysqldump alfresco > alfresco_outfile.sql
```

Customization files

You might be customizing your Alfresco application over a period of time. Typically, you might have added or updated the following files:

- Logos, images, and stylesheets
- JSP files (dashboard)
- Presentation templates
- Configuration files, property files
- Files in the extension folder
- Custom application code (WAR file, source Java files, and so on)

The process you follow to maintain and back up your customization files depends upon the development process you follow within your organization. It is useful to maintain your customization files in some configuration management system such as CVS or SVN, which helps you to easily maintain and back up.

Membership data

If you have used the Alfresco out-of-the-box membership system, then the data is stored in the relational database. You don't have to do any special task to back up the data as you are already backing up the relational database tables.

If you have used an external membership system such as Active Directory or OpenLDAP to have a single sign-on or centralized identity management system, then you must consider backup of your membership data as well.

You will have access to back-up tools based on the membership system you have used. Ensure that the data in the external membership system is backed up.

Log files

The location of the log files depend upon the application server. For the Tomcat installation, the log files are located at <install_folder> itself. Tomcat application server creates a log file per day. The current log file is named alfresco.log and at the end of the day, the log file will be backed up as alfresco.log.YYYY-MM-DD (for example, alfresco.log.2010-03-25).

Based on the usage of the system and on the logging level, the size of these log files might be pretty big. Hence, it is a good practice to back up the older log files and remove them from the current location to save hard disk space.

Backup frequency

The frequency at which you take backup really depends upon the nature of the application, your high availability requirements, and the Alfresco deployment option you have chosen.

For example, you can consider only a one-time backup of customization files. You can back up the files whenever you enhance the application or upgrade the application to newer versions.

Since the content, metadata, and tasks change very frequently, regular backup of the Alfresco filesystem and relational database is required. You have to consider the business risk and system resources availability while deciding on the back-up frequency.

Backup is based on Alfresco deployment

If your application is highly accessed by thousands of users, then it is important for you to deploy Alfresco in a clustered environment. If it is a critical application such as finance or insurance, then you should consider deploying Alfresco in hot back-up mode with master-slave configuration. The data backup policy and process might be different based on the way you have deployed Alfresco.

The typical process to back up Alfresco repository is as follows:

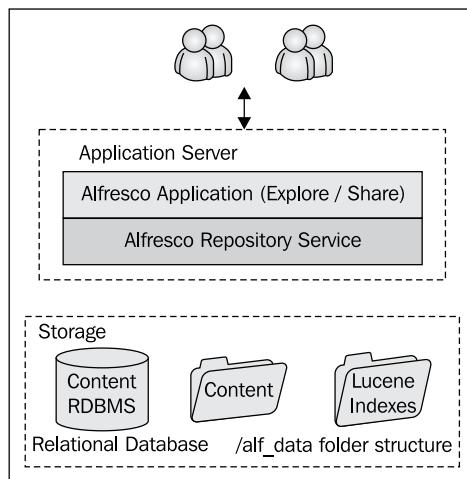
1. Stop Alfresco to ensure that no one can make changes during the backup.
2. Export the MySQL (or other) database.
3. Back up the Alfresco `alf_data` directory.
4. Start Alfresco.

To restore Alfresco repository:

1. Stop Alfresco.
2. Delete the `alf_data` folder and restore the `alf_data` folder that you backed up earlier.
3. Drop the database and import the database that you have exported.
4. Start Alfresco.

Alfresco deployed as a Repository Application Server

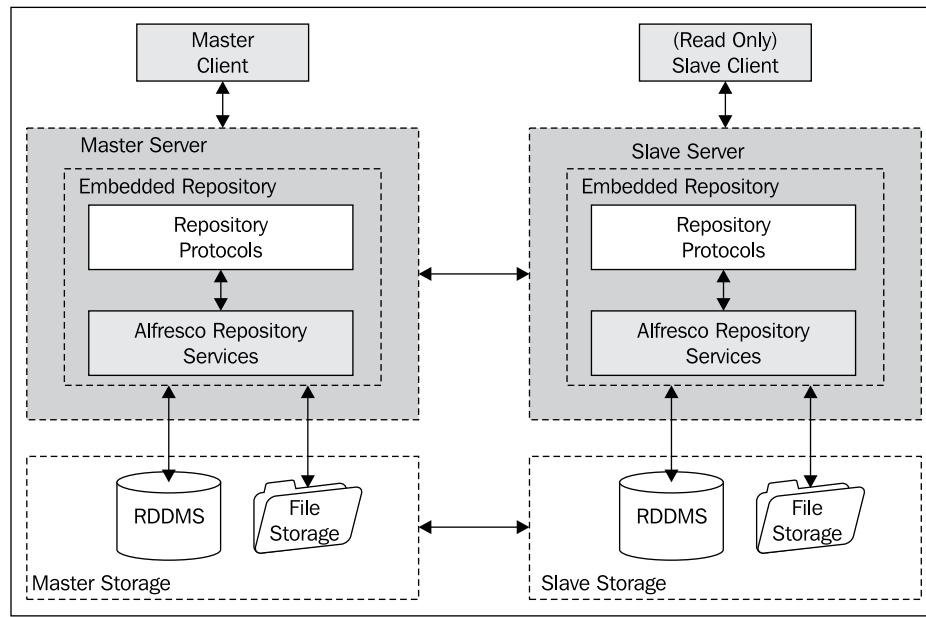
In this deployment (shown in the following figure), the web application becomes the host for an embedded repository and remote access is via the application, that is, HTTP. This is the default deployment option chosen by the Alfresco installer. This means the repository automatically benefits from any enhanced features provided by higher-end web application servers. For example, the repository can be embedded inside Apache Tomcat for the lightest weight deployment, but also embedded inside J2EE-compliant application servers from JBoss, Oracle, or IBM to take advantage of distributed transactions, and much more:



In this deployment option, you need to take the backup of the `alf_data` folder and the database. There will be one copy of customization files.

Alfresco deployed as a hot backup

In this deployment, as shown in the following figure, one Repository server is designated the master and another completely separate Repository server is designated the slave. The live application is hosted on the master, and as it is used, synchronous and asynchronous replication updates are made to the slave, that is, the backup. The backup remains in read-only mode. If for some reason, the master breaks down, it is a relatively simple task to swap over to the slave to continue operation.



In this deployment option, you don't need to take a regular backup as the data is getting backed up automatically.

Upgrading to new versions of Alfresco

You can consider upgrading to a new version of Alfresco if you are expecting one of the following benefits:

- Security patches
- Bug fixes
- New features
- Compatibility with other systems

Even if you are not getting the benefits listed earlier, sometimes you might consider upgrading to a newer version, as you do not want to maintain a big gap between the Alfresco version on which your application is currently running and the latest Alfresco version. If this gap is too big, it might be very expensive for you to upgrade later on. This is the scenario with most enterprise software.

Alfresco has upgrade scripts, which helps you to upgrade to newer versions automatically. However, it is essential to follow certain best practices while upgrading your system. Always try upgrading the test or staging server first before trying on production server. It is essential that you back up your existing data before attempting an upgrade. Follow the information and instructions given in the *Data backup* section.

Upgrading to a minor release

An Alfresco minor (or dot) release typically contains bug fixes and minor enhancements. There will not be any new features. An example is upgrading from Alfresco 3.2 to 3.2.1 release.

Since there are no new features, typically, the database schema remains the same. In this situation, you can replace only the web application (WAR) file to upgrade.

The WAR file (`alfresco.war`) for Tomcat installation is located in the `<install_folder>/tomcat/webapps` folder.

Follow these steps for minor upgrades:

1. Download the latest `alfresco.war` file from the Alfresco website.
2. Stop Alfresco.
3. Back up all of the data including customization files (as explained in the earlier sections).
4. Delete the web application folder `<install_folder>/tomcat/webapps/alfresco`.
5. Replace the `alfresco.war` file in the `<install_folder>/tomcat/webapps` folder with the latest one.
6. Restore the customization files.
7. Start Alfresco.

Test your application after upgrading it to ensure that the upgrade is successful.

Upgrading to a major release

An Alfresco major release typically contains new features, performance enhancements, and bug fixes. An example is upgrading from Alfresco 2.x to 3.x releases.

The upgrade scripts will be executed automatically by the server when starting up against an existing database. Scripts that support the various Hibernate dialects can be found in the `<configRoot>/alfresco/dbscripts/upgrade/*` folders. This means that you don't have to do manual upgrades any more.

For example, let us assume that you are using the Tomcat bundle of Alfresco 2.1 (installed in C:\alfresco2.1 folder) on a Microsoft Windows operating system, and you want to upgrade to Alfresco 3.3 release.

Follow these steps for major upgrades:

1. Stop Alfresco in your current installation folder C:\alfresco2.1.
2. Back up all of the data, including customization files (as explained in the earlier sections).
3. Download the complete Alfresco package, Tomcat bundle for the Microsoft Windows operating system.
4. Perform a new installation in a different folder (say C:\alfresco3.3).
5. Copy the older Alfresco file content folder to the newer installation (copy the C:\alfresco2.1\alf_data folder to C:\alfresco3.3\alf_data).
6. Create a new database table and restore the relational database content from the older database. Update the Alfresco configuration file in the new installation to point to this new database.
7. Restore the customization files in the new installation.
8. Start Alfresco in the new installation.

Though most of the upgrade happens automatically, you might have to perform some manual steps to restore your customization files in a new installation.

There are some configuration files and a properties file in Alfresco's config folder (/tomcat/webapps/alfresco/WEB-INF/classes/alfresco/), which you might have updated, which require manual updates.

While I was writing this book, I upgraded the example application from Alfresco 2.1 version to Alfresco 3.3 version. I used the following script to restore some of the customization files. I manually updated some of the configuration files. Refer to the batch file that I used to restore the customization files on the Windows platform.

```
rem -----
rem Replaces/Adds Alfresco Custom Files to new Alfresco installation
rem -----
set L_LOCALDIR=%CD%
set L_SRCDIR=C:\alfresco_book_21
set L_DESTDIR=C:\alfresco_book_30
rem ----- Replace Logos -----
CD %L_DESTDIR%\tomcat\webapps\alfresco\images\logo
move AlfrescoLogo32.png AlfrescoLogo32.png-ORIGINAL
move AlfrescoLogo200.png AlfrescoLogo200.png-ORIGINAL
move AlfrescoFadedBG.png AlfrescoFadedBG.png-ORIGINAL
```

```
copy %L_SRCDIR%\tomcat\webapps\alfresco\images\logo\AlfrescoLogo32.png  
.copy %L_SRCDIR%\tomcat\webapps\alfresco\images\logo\AlfrescoLogo200.png.  
copy %L_SRCDIR%\tomcat\webapps\alfresco\images\logo\AlfrescoFadedBG.png.  
rem ----- Copy files in extension folder -----  
CD %L_DESTDIR%\tomcat\shared\classes\alfresco\extension  
copy %L_SRCDIR%\tomcat\shared\classes\alfresco\extension\custom-model-context.xml.  
copy %L_SRCDIR%\tomcat\shared\classes\alfresco\extension\customModel.xml.  
copy %L_SRCDIR%\tomcat\shared\classes\alfresco\extension\web-client-config-custom.xml.  
CD %L_LOCALDIR%  
echo I am done...  
pause
```

You can create your own batch scripts to automatically restore your customization files. Typically, most of the developers use tools such as Eclipse to build and deploy the customization files to the newer installations.

Test your application after upgrading it to ensure that the upgrade is successful.

Cleaning up deployment history

There are two ways to clean out old deployment reports in WCM:

- Using Alfresco Explorer
- Using scheduler

Using Alfresco Explorer

Administrator and Content Managers of the web project have the right to delete the Deployment Reports. Follow these steps:

1. Go to the Staging Sandbox area of the web project.
2. Click on the **Actions** menu.

3. Click on the **Delete All Deploy Reports** submenu.

The screenshot shows a web browser window with the URL 'Company Home > Web Projects > cignex'. The main content area displays a 'cignex' web project with a 'Staging Sandbox' section. On the right, there is a context menu titled 'Actions' with several options: 'View Details', 'Create Webapp Folder', 'Edit Web Project Settings', 'Invite Web Project Users', 'Delete All Deploy Reports' (which is highlighted with a red box), and 'Delete'. Below the 'Actions' menu, there are links for 'Browse Website', 'Preview Website', and 'Refresh'.

4. It will ask for confirmation; click on **Yes** to delete the reports.

Using scheduler

Using scheduler you can delete the reports older than the given number of days. You can configure this scheduler to run on a periodic basis. Follow the next steps to configure this scheduler.

1. Rename the file `deployment-attempt-cleaner-context.xml.sample` to `deployment-attempt-cleaner-context.xml` specified at the location `<alfresco_home>/tomcat/shared/classes/alfresco/extension`.
2. Configure the following highlighted changes:

```

<bean id="avmDeploymentAttemptCleaner"
      class="org.alfresco.repo.avm.AVMDeploymentAttemptCleaner">
    <!-- number of days to keep deployment attempts -->
    <property name="maxAge">
        <value>1</value>
    </property>
    <property name="nodeService">
        <ref bean="NodeService" />
    </property>
    <property name="transactionService">
        <ref bean="TransactionService" />
    </property>
    <property name="searchService">
        <ref bean="SearchService" />
    </property>
    <property name="importerBootstrap">
        <ref bean="spacesBootstrap" />
    </property>

```

```
        </property>
    </bean>
<bean id="avmDeploymentAttemptCleanup"
      class="org.alfresco.util.CronTriggerBean">
    <property name="jobDetail">
        <bean id="avmDeploymentAttemptCleanerDetail"
              class="org.springframework.scheduling.quartz.JobDetailBean">
            <property name="jobClass">
                <value>
                    org.alfresco.repo.avm.AVMDeploymentAttemptCleanerJob
                </value>
            </property>
            <property name="jobDataAsMap">
                <map>
                    <entry key="deploymentAttemptCleaner">
                        <ref bean="avmDeploymentAttemptCleaner" />
                    </entry>
                </map>
            </property>
        </bean>
    </property>
    <property name="scheduler">
        <ref bean="schedulerFactory" />
    </property>
    <!-- trigger at 5:00am each day -->
    <property name="cronExpression">
        <value>0 0 5 * * ? </value>
    </property>
</bean>
```

This will purge all deployment attempt nodes older than one day at 5 a.m. every morning.

Deployment report 1 day before

Go to the Staging Sandbox of the web project. Click on the **View Deployments** link.

You will find the history of all of the deployment reports on that page as shown next:

The screenshot shows a web-based deployment report interface. At the top, it says 'Company Home > Web Projects > cignex'. Below that is the title 'Last Deployment Report' with a sub-instruction 'View deployment details for each of the servers selected in the last deployment.' A section titled 'Server 1' displays deployment details: 'Deployment Successful', 'Server: localhost:50500', 'Snapshot: 11', 'Started: 29 March 2010 16:36', 'Finished: 29 March 2010 16:36', and 'By: admin'. There is a link '► Details'. Below this is a section titled 'More Deployment Reports' with a dropdown menu showing 'Today', 'Yesterday', 'Last 7 days' (which is highlighted in blue), 'Last 30 days', and 'All'. A table lists deployment attempts:

Attempt Date	Deployed To	Snapshot
29 March 2010 16:36	Server 1	11
29 March 2010 16:31	Server 1	13

Deployment report 1 day after

Since we have configured to clean up reports that are one day older, on the next day, we can see only those reports which are deployed today. All other reports, which are shown in the previous screenshot, have been purged.

General maintenance tips

If you maintain the system regularly by cleaning up the database and fixing the system errors, your system runs faster. Some of the tips are given in this section.

Examine log files

Your log files tell you very important issues and problems about your system. The level of details logged will be based on the level of logging (INFO, ERROR, and DEBUG).

The log files are named as `alfresco.log` (current one) or `alfresco.log.YYYY-MM-DD` (older ones). Examine one of the log files and you will notice the log entries made in the following categories.

- ERROR: Error occurred (requires FIX)
- WARN: Warning messages (requires your attention)
- INFO: General information about the system

The sample messages are given next:

```
11:20:42,088 WARN [org.hibernate.cache.EhCacheProvider] Could not  
find configuration [org.jbpm.graph.def.Node]; using defaults.  
11:21:45,056 ERROR [org.alfresco.repo.action.ActionServiceImpl] An  
error was encountered whilst executing the action 'import'.  
org.alfresco.service.cmr.view.ImporterException: Failed to import  
package at line 8; column 19 due to error: A complete repository  
package cannot be imported here...  
15:03:19,308 INFO [org.alfresco.repo.admin.patch.PatchExecutor] No  
patches were required. .
```

You have to fix the errors listed in the log file and make sure there are no **ERROR** messages in the log files. There are many utilities (based on the operating system), which examine the log file for errors and send you notifications as required. Consider using or developing such a tool to be notified as soon as an **ERROR** occurs.

Reset the administrator password

Administrator has the highest powers in the Alfresco application. It is a good practice to periodically change the administrator password as a security process. You can change the password using the Alfresco Explorer's user profile option.

You also need to make the change in `alfresco-virtserver.properties` for the new password, as follows:

1. Modify value for the `alfresco.server.password` property in the above mentioned file to the new password.
2. Restart the server.

Providing administrator rights

By default, only the admin user will have administrator rights. You can provide administrator rights to any user using Alfresco Explorer as mentioned next:

1. Log into Alfresco as admin user.
2. Navigate to the Administrator Console.
3. Click on **Manage User Groups** and click on **ALFRESCO_ADMINISTRATORS** group.
4. Click on the **More Actions** menu and select the **Add User** option.
5. Search the users to whom you want to assign administrator rights and click on **Add**.

6. You can see those users added in the group as shown in the following screenshot:

Now these users will also have the Administrator rights and can perform all administrative tasks from Alfresco Explorer.

Reset complete repository data

If you are setting up an environment to test your Alfresco application, you might want to remove or reset the data once the testing is done. There might be other circumstances when you want to remove the existing users, spaces, and rules from the repository, and start fresh. Before deleting or resetting the complete repository, you might want to back it up.

Following is the process to reset the complete repository data:

1. Stop Alfresco.
2. Remove the `alf_data` folder.
3. Drop the Alfresco database and create an empty Alfresco database.
4. Start Alfresco.

When you start Alfresco, the `alf_data` folder will be created and the default database tables will be created automatically.

Migrating servers

The process of migrating an instance of Alfresco running on one server to another server follows a similar pattern to the back-up process, with additional steps of ensuring any configuration is also copied over.

Summary

In this chapter, we learned:

- Alfresco Explorer has administrative utilities to export data from the Alfresco repository and import within the repository or to another repository.
- You must back up data at regular intervals to protect your data from hardware failures.
- To consider the hot backup-deployment option of Alfresco for high availability.
- The upgrade scripts in Alfresco help you to upgrade to newer versions automatically.
- It is recommended that you try an upgrade on a test or staging server before going onto a production server.
- You can provide administrator rights to any user. You have also learned different ways to clean out old deployment reports.

Index

A

advanced configuration, description document 314-316
advanced features, FSR
about 242
payload transformations, defining 243, 244
`postCommit()` callback,
 configuring 242, 243
`prepare()` callback, configuring 242, 243
transport adapters, defining 244
advanced schema attributes, web forms
about 111
common XSD and JSP, reusing 122, 123
drop-down list, populating 122
file pickers 111-116
validation 118
WYSIWYG editor, customizing 118-121
Advanced Versioning Manager (AVM)
about 12, 85
directory version comparison 86
extended versioning operations 85
features 85
file version comparison 86
store version comparison 86
Alfresco
Alfresco Explorer, starting 73
Alfresco Share, starting 72
Alfresco virtualization server, starting 73
Alfresco virtualization server, stopping 74
configuring, as Windows service 77
data backup 389
deployment engine, starting 74
deployment engine, stopping 75
deployment history, cleaning 398
installation directory 76

maintaining 389
maintenance tips 401
running 71
search, overview 376
server, starting 72
server, stopping 73
starting, as console application 75
stopping, as console application 75
upgrading, to major release 396, 398
upgrading, to minor release 396
upgrading, to new versions 395-398
user roles 360

Alfresco 3 web scripts

features 309, 310

Alfresco application

debugging, in Eclipse environment 39, 40

Alfresco Community Lab Network 26

Alfresco components

Alfresco Module Package, installing 66-68
Alfresco WCM, installing 54, 55
Flash Player, installing 63
ImageMagick, installing 60, 61
Microsoft Office add-ins, installing 61, 62
Microsoft Office SharePoint Protocol
 Support, installing 68, 69
OpenOffice, installing 58-60
SharePoint Protocol, configuring for Online
 Editing 71
SharePoint Protocol Support AMP,
 installing 69, 70
SharePoint Protocol Support,
 configuring 70, 71
SWFTools, installing 63
SWFTools, installing on Linux 64, 65
SWFTools, installing on Windows 63
TinyMCE language packs, installing 66

WCM installation, verifying 55
WCM standalone deployment receiver,
installing 56

Alfresco content production environment
about 230
live server vs. test server 231
static vs. dynamic delivery model 231, 232

Alfresco DM
about 353
features 353
LDAP configuration 354
Membership system 354
security mechanism 354

Alfresco Document Management. *See*
Alfresco DM

Alfresco Dynamic Website 271

Alfresco Enterprise Network 26

Alfresco Explorer
starting 73
used, for cleaning up deployment
history 398, 399

Alfresco Explorer Task dialogs, custom
WCM Workflow
creating 189

Alfresco File System Receiver. *See* **FSR**

Alfresco installation
about 40
Alfresco WAR, installing on any
platform 52
Alfresco WCM, installing 54
components, installing 54
extension samples, downloading 53
manual installation 41
on Mac 50
on Red Hat Linux 48
on Windows 41
Share, deploying in separate Tomcat
instance 53
softwares required 21
Tomcat 6.x directory paths, modifying 52

Alfresco installation options. *See*
installation options, Alfresco

Alfresco, installing on Mac 50, 51

Alfresco, installing on Red Hat Linux
installation wizard method 48
Tomcat bundle installation 50

Alfresco, installing on Windows
excluding JDK 45, 46
full installation 41-45
Tomcat bundle installation 47

Alfresco Module Package(AMP)
about 57
installing 66-68

Alfresco Network 271

Alfresco repository
backing up 393
restoring 393

Alfresco server
starting 72
stopping 73

Alfresco Server Receiver. *See* **ASR**

Alfresco Share
about 271
starting 72

Alfresco Surf
about 269
features 270
platform 269

Alfresco Surf architecture
about 272
Dispatcher Servlet 272
FreeMarker 274
JSP 274
MVC pattern 272
View 273
Web Scripts 273

Alfresco Surf platform. *See* **Surf platform**

Alfresco virtualization server
starting 73
stopping 74

Alfresco WAR
installing, on any platform 52

Alfresco WCM
about 7, 24
enterprise class features 291
features 10, 17
installation options 26
installing 54
integrating, with Surf-based web
application 349
web development framework 24
web scripts, using 311, 312

Alfresco WCM model
about 11
delivery models 14
in-context preview 12
renditions templates 13
sandboxes 11
transparent layers 12, 13
virtualization 12
web forms 13
web projects 11
Web scripts 14
workflows 14

Alfresco WCM-Surf-based web application integration
about 349
response template 349
web scripts, integrating with SURF application 350

Alfresco WCM, version 3.3
Alfresco Web Editor 17
features 17
Rendition API 17
Transfer Service API 17
WCM deployment 18

Alfresco WCM web interface
about 80
logging in 80

Alfresco Web Content Management. *See* **Alfresco WCM**

Alfresco Web Editor (AWE)
about 18, 269, 295
deploying 295, 296
deploying, to Spring Surf application 297, 298
framework 302
sample web application 299, 301
tag library 298, 299
using 296

Alfresco web script framework 308, 309

Alfresco web scripts
framework 308
overview 307

Alfresco Web Studio 271

Apache Ant
about 35
features 35
integrating, with Eclipse 35

App 303

application servers
about 28
JBoss 28
Tomcat 28

applications, using Surf platform
Alfresco Dynamic Website 271
Alfresco Network 271
Alfresco Share 271
Alfresco Web Studio 271

ASR 232, 245

ASR, for dynamic delivery
about 245
AVM Deployment Target 246
WCM deployment service, configuring 245

authentication element 314

auto deployment 246

AVM API, FreeMarker methods 319

AVM APIs, JavaScript methods
lookupNode(path) 324
lookupStoreRoot(storeid) 324
lookupStore(storeid) 324
stores 324
webappsFolderPath(storeid) 324

AVM CIFS projection 97, 98

AVM Deployment Target 245, 246

AVM node APIs, FreeMarker methods
hasLockAccess 320
isDirectory 320
isFile 320
isLocked 320
isLockOwner 320
parentPath 320
path 320
rename(name) 320
version 320

AVM node APIs, JavaScript methods
hasLockAccess 325
isDirectory 324
isFile 324
isLocked 324
isLockOwner 325
parentPath 324
path 324
rename(name) 325
version 324

AVM root node APIs, FreeMarker methods

- assetUrl(path) 320
- assetUrl(storeId, path) 320
- avm.userSandboxStore(storeId, username)
319
- getModifiedItems(storeId, username,
webapp) 319
- lookupNode(path) 319
- lookupStoreRoot(storeid) 319
- lookupStore(storeid) 319
- stagingStore(storeId) 319
- stores 319
- webappsFolderPath(storeid) 319
- websiteStagingUrl(storeId) 319
- websiteUserSandboxUrl(storeId,
username) 320

AVM store APIs, FreeMarker methods

- about 320
- createdDate 320
- creator 320
- id 320
- lookupNode(path) 320
- lookupRoot 320
- luceneSearch(query) 320
- name 320

AVM store APIs, JavaScript methods

- createdDate 324
- creator 324
- id 324
- lookupNode(path) 324
- lookupRoot 324
- luceneSearch(query) 324
- name 324

B

basic elements, description document

- authentication 314
- description 314
- format 314
- shortname 314
- url 314

build process, Eclipse

- Ant target, running 38, 39
- build.properties file, creating 38
- build.xml file, creating 35-37

build tool, YUI

- about 294

C

cache root object 310

categories of scopes, Surf framework

- global 283
- page 283
- template 283

common repository

- about 361
- Alfresco, integrating with FFMPEG video
transcoder 361
- videos, copying from DM to WCM 367

components, My Alfresco Dashboard

- all active tasks 81
- getting started 80
- My Completed Tasks 81
- My Document List 81
- my pooled tasks 81
- My Spaces List 81
- My Tasks 81
- my task to do 80
- My Web Files 81
- My Web Forms 81
- OpenSearch 81

components, web scripts

- about 312
- configuration document 313
- controller script 312
- description document 312
- locale message bundle 313
- response templates 313

ConfigRegistry object 303

config root object 309

config_search_name parameter 114

configuration document 313

content, copying from DM to WCM

- business rule, used 367-369
- JavaScript, used 370-372

content delivery 230

content expiration, WCM

- about 198
- configuration 198-201

content manager 256

content reviewer 256
controller script 312
Core WEF Components, Web Editor Framework
 about 303
 App 303
 Base 303
 ConfigRegistry 303
 loader 303
 Plugin 303
 PluginRegistry 303
 WEF 303
 widget 303
Core WEF Widgets
 about 303
 Ribbon 303
 Tabbed Toolbar 303
 Toolbar 303
CSS Resources, YUI 294
custom Aspect
 advantages 184
 disadvantages 184
customizations deploying, Alfresco WCM
 code, deploying as AMP 57
 code, integrating in existing Alfresco WAR file 57
custom model Spring context file, custom WCM Workflow
 creating 189
custom WCM Workflow
 creating 179
 testing 190-198
 workflow process, defining 181

D

Daisy Chaining 358, 359
data backup
 about 389
 Alfresco deployed, as hot backup 394, 395
 Alfresco deployed, as repository application server 394
 backup frequency 392
 based on Alfresco deployment 393
 content in file system, list of items 390, 391
 file customization, list of items 391
 list of items 390

log files, list of items 392
membership data, list of items 392
metadata in relational database, list of items 391

databases
 MS SQL Server 27
 MySQL 27
 Oracle 27
 PostgreSQL 27

delivery models, Alfresco WCM
 about 14
 dynamic delivery model 15
 hybrid approach 16, 17
 overview 16
 static delivery model 15

deployment engine
 starting 74
 stopping 75

deployment, from Alfresco WCM to DM repository
 about 252
 Alfresco DM, setting up as deployment target 252
 deploying to DM 253

deployment history
 cleaning, Alfresco Explorer used 398, 399
 cleaning, scheduler used 399, 400

description document
 about 312
 advanced configuration 314, 315
 basic elements 314
 creating 313

description element 314

development tools, YUI 294

dynamic Alfresco Explorer
 about 213
 deploying 214

dynamic Alfresco Explorer customizations
 deploying 214

dynamic Alfresco Explorer customizations deployment
 first approach 214
 web client customizations, reloading 215
 workflow, testing 216

dynamic delivery model 15

dynamic deployment
 about 203, 232

advantages 204
dynamic Alfresco Explorer 213
dynamic models 204
dynamic Resource Bundle 206
dynamic workflows 208
dynamic models
about 204
model file, deploying 204
dynamic process definition
creating 208
deploying 208
dynamic process definition deployment
about 208
first approach 209
second approach 210-212
third approach 213
workflow images, displaying 213
dynamic Resource Bundles
creating 206
deploying 206
dynamic websites, using WCM
about 101
getRealPath() method, previewing 102
virtual server configuration 102
virtual server JSP support 102
WARs, previewing 102

E

Eclipse installation, Alfresco
about 30
application, building 35
build path, configuring 32
development environment,
setting up 30, 32
source code tree 33
elements, for description document
advanced configuration
cache 315
family 315
kind 315
lifecycle 316
negotiate 315
transaction 314
endTemplate tag, AWE tag library 299
Extensible Markup Language. *See XML*
Extensible Stylesheet Language
about 132

XSL-FO 132
XSLT 132
Extensible Stylesheet Language Formatting Objects. *See XSL-FO*
Extensible Stylesheet Language Transformations. *See XSLT*

F

family element 310
FFMPEG 361
FFMPEG integration with Alfresco
about 361
FFMPEG transformation,
configuring 365-367
FFMPEG transformations, using as custom
action 363, 364
options for audio transcoding 362
options for video transcoding 362
FFMPEG transformation
configuring, as business rule 365-367
file-mapping.xml file 58
file system projection 97, 98
File System Receiver (FSR) 15
filter_mimetypes parameter 116
Flash Player
installing 63
format element
about 314
values 314
FreeMarker
about 107, 127
Alfresco objects available 129
FreeMarker template engine 130
node model API 130
using, for rendition templates 127, 128
free marker directives
uses 131
FreeMarker methods, for AVM repository
about 319
AVM API 319
AVM node API 320
AVM store API 320
JavaScript controller 322
response status 321
Web script controller 321
FreeMarker template engine 128

FreeMarker templates, for renditions

- about 127, 128
- Alfresco objects available 129
- creating 132
- defining 130, 131
- directives 130
- FreeMarker template engine 128
- template-node model API 130

FSR 232, 233**FSR, for static delivery**

- about 233
- advanced features 242
- FSR, installing 233, 234
- FSR, using from Alfresco WCM
 - staging 236, 237

FSR from Alfresco WCM staging

- deployment history, viewing 239
- deployment report, viewing 239
- multiple servers, deploying to 242
- snapshot, deploying to FSR
 - manually 237, 238
- snapshots, reverting to 241
- web project, configuring 236, 237

FSR installation

- about 233
- deployment receiver, starting 235
- deployment receiver, stopping 235
- deployment targets, configuring 235
- FSR, configuring 233, 234

G**general maintenance tips, Alfresco**

- about 401
- administrator password, resetting 402
- administrator rights, providing 402
- complete repository data, resetting 403
- log files, examining 401
- servers, migrating 403

H**Hibernate 27****I****ImageMagick**

- installing 60, 61

image transformation, WCM

- about 372
- image transformation action, using 375, 376
- image transformation APIs 373
- new action, configuring 373-375

in-context preview, Alfresco WCM 12**installation**

- Alfresco 21, 40
 - JDK 22
 - MySQL 23
- installation directory, Alfresco**
- about 76
 - alf_data 76
 - alfresco 76
 - amps 76
 - bin 76
 - extras 76
 - java 76
 - licenses 76
 - openoffice 77
 - README file 77
 - tomcat 77
 - virtual-tomcat 77

installation options, Alfresco WCM

- about 26
 - Alfresco Community Lab Network 26
 - Alfresco Enterprise Network 26
 - application servers 28
 - databases 27
 - Eclipse installation 30
 - operating system s 27
 - portals 29
 - software, selecting 29
- integrate-extension target 57**

J**Java-backed controller**

- about 325
- Java Bean class, creating 325
- Java Bean, declaring 325
- web script, creating 326
- web script, listing for external access 329
- web script, registering 328, 329
- web scripts, implementing 326
- web script, storing 326
- web script, storing on Alfresco Explorer 326, 327

web script, storing on file system 326

JavaScript controller 322

JavaScript methods, for AVM repository

- about 323
- AVM API 324
- AVM node API 324
- AVM store API 324

JavaServer Pages (JSP) 18

JBoss 28

JBoss jBPM 161

JBoss portal 29

jBPM engine 161

jBPM Process Definition Language. *See* **jPDL**

JDK

- installing 22

JDK installation

- about 22
- JAVA_HOME environment variable
 - location, verifying 23

jPDL 162

L

labels, web forms

- about 117
- localizing 117, 118
- tool tips 117

layered folders

- about 261, 262
- destination file, updating 266
- files, deleting 266
- new files, adding 267
- source file, updating 264, 265
- transparent folder, creating 262-264

LDAP-based authentication

- LDAP synchronisation 357

LDAP configuration

- about 354
- Daisy Chaining 358
- with Active Directory 355-357

LDAP synchronisation 357

Liferay 29

Linux

- about 27
- advantages 27

loader, Core WEF Components 303

M

Mac

- Alfresco installation 50, 51

markContent tag, AWE tag library

- about 299
- formId attribute 299
- id attribute 299
- nestedMarker attribute 299
- title attribute 299

message bundles 313

Microsoft Office add-ins

- installing 61, 62

model file deployment

- about 204
- custom model, updating 206
- first approach 205
- second approach 206

module.properties file 57

multiple websites, managing

- about 256
- forms, reusing 256, 257
- single web project, used 259
- templates, reusing 256, 257
- web project, using as template 258

multiple websites, managing with single web project

- about 259
- FSR, setting up for target website 260
- multiple URLs, setting up on target server 259
- webapp folders, creating 260

MVC pattern, Surf architecture

- about 272
- single-tier application 274
- two-tier application 275

My Alfresco Dashboard

- about 80
- components 80
- configuring 80

MySQL

- installing 23

MySQL installation

- about 23
- verifying 23

N

negotiate element 310
news item web script
enhancing 344, 345

O

OpenOffice
installing 58-60
operating systems
Linux 27
MacOS 27
Unix 27
Windows 27

ORM (Object Relational Mapping) tool 27
out-of-the-box workflows

about 163
parallel 163
serial 163

P

parseXMLDocument, node method 130

Plugin component 303

PluginRegistry 303

postCommit() callback 242

prepare() callback 242

presentation templates

FreeMarker template engine 130

Process Definition, custom WCM Workflow

creating 185
key terms 185
process definition name, defining 186
Swimlane, defining 186
task, associating 187

processTagging method 346

R

Red Hat Linux

Alfresco installation 48
Alfresco installation, Tomcat bundle
used 50

rendering engines, Surf APIs

about 280
objects 280

renderTemplate method 345

rendition templates

about 127
Extensible Stylesheet Language 132
FreeMarker templates used 127
XSL-FO, using 134
XSLT, using 132, 133

rendition templates, Alfresco WCM 13

Representational State Transfer. See **REST**

Resource Bundles deployment

about 206
first approach 207
Resource Bundle, reloading 208
Resource Bundle, updating 208
second approach 208

response template 313

response type formats 317

REST

about 306
main principles 306

REST architecture

overview 306

Ribbon component 303

root objects, for execution script

actions 323
args 322
argsM 323
avm 323
classification 323
companyhome 322
crossrepository 323
formdata 323
guest 323
logger 323
model 323
people 323
person 322
roothome 323
search 323
server 323
session 323
url 323
userhome 322
utils 323
webprojects 323

root objects, FreeMarker

about 318
args 318

argsM 318
avm 318
classification 318
companyhome 318
date 318
guest 318
person 318
roothome 318
server 318
session 318
url 318
userhome 318
webscript 318
workflow 318

root objects, Surf APIs
content () 278
context () 277
page () 278
remote () 279
sitedata () 279
url 278
user () 277

runas attribute 309

S

sample web application, AWE 299-301
sandboxes, Alfresco WCM 11
scheduler
used, for cleaning up deployment history 399, 400
search 103
selectable_types parameter 115
SharePoint Protocol Support
configuring 70, 71
configuring, for Online Editing 71
installing 68, 69
SharePoint Protocol Support AMP
installing 69, 70
shortname element 314
simple element, web forms
default values 110
defining 109
fixed values 110
optional values 110
required values 110

single-tier application, Alfresco Surf architecture 274
snapshot 230
softwares requisites, Alfresco installation
Database 21
Flash Player Version 10.x 22
JDK 21
OpenOffice.org 22
SWFTools 22

standalone application, Surf framework used
communication, with WCM 291-293
component, using in page 283, 284
creating 280
page, designing 281, 282
page navigation, designing 285-290

startTemplate tag, AWE tag library
about 298
toolbarLocation attribute 298

static delivery model 15
static deployment 232

Surf APIs
about 276
methods 276
rendering engines 280
root objects 276

Surf framework
categories, of scopes 283

Surf Model objects
about 276
Chrome 276
Component 276
Component Type 276
Configuration object 276
Content Association 276
Page 276
Page Association 276
Template Instance 276
Template renderer 276
Template Type 276

Surf platform
about 269, 270
design 270

SWFTools
installing 63
installing, on Linux 64

installing, on Windows 63

T

Tabbed Toolbar component 303

tag library, AWE

about 298

endTemplate tag 299

markContent tag 299

startTemplate tag 298

Task Model, custom WCM Workflow

about 182

aspects, defining 183, 184

Content Model name, defining 182

Content Type, creating for each task 182

creating 182

properties, defining 183

Test Server Deployment functionality

about 247

content, previewing 250

deploying, from workflow 251

test server, deploying 249

test server pool, setting up 248, 249

test server, releasing 250, 251

TinyMCE language packs

installing 66

Toolbar component 303

transparent folder

about 262

creating 262-264

transparent layers, Alfresco WCM 12

two-tier application, Alfresco Surf

architecture 275

U

Universal Disk Formats. *See* UDFs

update-war target 58

url element 314

user roles

about 360

collaborator 360

consumer 360

contributor 360

coordinator 360

editor 360

User Sandbox interface 84

V

virtualization, Alfresco WCM 12

Virtualization server

about 99

access, to User Sandbox 100

configuration 101

configuring, for preview 99

URL format 99

vti.alfresco.alfresoHostWithPort

property 71

vti.alfresco.deployment.context property 71

vti.properties file 71

vti-server.port property 71

vti.share.shareHostWithPort property 71

W

WCM

content, expiring 198

image transformation 372

WCM installation

verifying 55

WCM integration with external applications

about 330

Alfresco WCM, integrating with

Drupal 333

Alfresco WCM, integrating with J2EE web

application 338

Alfresco WCM, integrating with

Liferay 330

portlet, creating in Liferay 333

web script, creating for fetching news

headline 330

web script, for fetching blog content 333

web script, for fetching details of a

particular news item 338

WCM solutions

about 10

physical architecture 25

Publishing style 10

recommendations 25

Wiki style 10

WCM standalone deployment receiver

installing 56

WCM systems 9

WCM web scripts
about 306
concepts 306

WCM Web Scripts
integrating, with Surf application 291, 292

WCM Workflows
about 159-161
advantages 160
configuring 163
need for 160
out-of-the-box workflow 163
workflow process 162

web content
creating 141-146

web development framework, WCM
.NET 24
Java 24
PHP 24
Python 24
Ruby 24

Web Editor Framework
about 18, 302
Core WEF Components 303
Core WEF Widgets 303

web forms
about 107
associating, with specific/ multiple projects 139-141
benefits 106
creating 107
creating, in Alfresco 123-126
editing, for renditions 147, 148
features 106

web forms, Alfresco WCM 13

web forms, creating
advanced schemas attributes 111
complex element, defining 109
schema, defining 108
simple element, defining 109
structure, identifying 108

web forms editing, for renditions
combo box, populating 153-155
update action, using 148-150
.xml file, associating to web form 151, 152

web project
about 86
content, adding 94, 95

content, submitting to staging
sandbox 95-97

creating 86, 87
creating, from existing web project 88
deployment receivers, configuring 88
e-mail notification, sending 90
sample web project information 88
site, creating 90-92
users, adding 89

User Sandboxes, listing 93, 94

Web Forms, configuring 88
workflow, configuring 89

web projects, Alfresco WCM 11

web project sandboxes 82, 83

Web publishing dashlets
about 156
configuring 156, 157

web script
about 307
advantages 308
calling, from JSP page 342
using, with Alfresco WCM 311, 312

web script controller
about 321
objectives 322

web script, for fetching blog content
about 333
calling, in Drupal 337
description document 334
execution script 334
response template 335, 336
storing/registering, in Alfresco 337

web script, for fetching details of a particular news item
about 338, 345
calling, from JSP page 348
description document 338, 345
Java-backed Bean 339, 345, 346
response template 339, 340, 346, 347
storing/registering, in Alfresco 341, 348

web script, for fetching news headlines
about 330
description document 330
execution script 331
response template 331
storing/registering, in Alfresco 332

Web Scripts, Alfresco WCM 14

web scripts implementation, for WCM
about 312
components of web scripts 312
description document, creating 313
response templates 316
WEF.Base 303
WEF object 303
widget 303
Windows
Alfresco installation 41-45
Alfresco installation, excluding JDK 45, 46
Alfresco installation, Tomcat bundle
 used 47
workflow
about 159
changing, for each snapshot
 submission 178
customizing 217
e-mail notification, adding 217, 218
removing, for specific staging
 submission 219-221
workflow configuration
about 163
content, submitting to staging box 168
edit web content wizard, using 168
submit items wizard, using 170-177
workflows, associating to web forms 164
workflows, associating to web
 project 165-167
workflow process, custom WCM Workflow
Alfresco Explorer Task dialogs,
 creating 189
custom model Spring context file,
 creating 189
defining 181
Process Definition, creating 185, 186
Task Model, creating 182
WCM project, deploying 189
workflow resource bundles, creating 188
workflow resource bundles, custom WCM Workflow
 creating 188
workflows, Alfresco WCM
about 14
association 14
definition 14
instance 14

workflow viewer 227, 228

X

XML 107
XML Schema Definition. See XSD
XSD 107
XSD data types
 xs:anyType 110
 xs:anyURI 110
 xs:boolean 110
 xs:date 110
 xs:decimal 110
 xs:double 110
 xs:enumeration 110
 xs:float 110
 xs:integer 110
 xs:normalizedString 110
 xs:string 110
 xs:time 110
XSL-FO
about 107, 134
rendition templates, associating to web
 forms 134-138
XSLT
about 107, 132
fundamentals 133
xs:schema element 108
xs:sequence element 109

Y

YUI
about 293
build tool 294
CSS Resources 294
development tools 294
features 293
YUI Core 293
YUI Utilities 293
YUI Widgets 293
YUI Core 293
YUI Utilities 293

Z

Zero workflow
implementing 222-226



Thank you for buying **Alfresco 3 Web Content Management**

About Packt Publishing

Packt, pronounced 'packed', published its first book "*Mastering phpMyAdmin for Effective MySQL Management*" in April 2004 and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: www.packtpub.com.

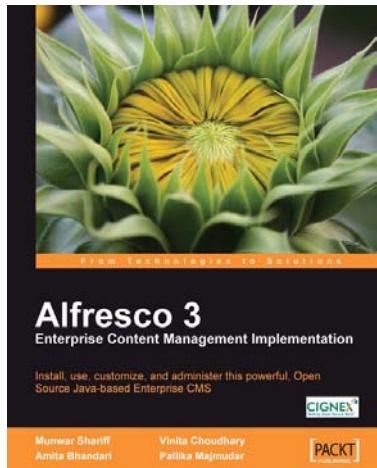
About Packt Open Source

In 2010, Packt launched two new brands, Packt Open Source and Packt Enterprise, in order to continue its focus on specialization. This book is part of the Packt Open Source brand, home to books published on software built around Open Source licences, and offering information to anybody from advanced developers to budding web designers. The Open Source brand also runs Packt's Open Source Royalty Scheme, by which Packt gives a royalty to each Open Source project about whose software a book is sold.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to author@packtpub.com. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.

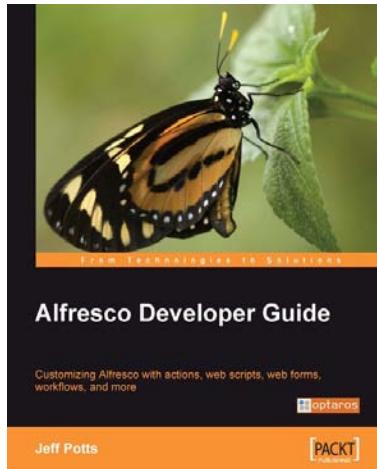


Alfresco 3 Enterprise Content Management Implementation

ISBN: 978-1-847197-36-8 Paperback: 600 pages

How to customize, use, and administer this powerful, Open Source Java-based Enterprise CMS

1. Manage your business documents with version control, library services, content organization, and advanced search
2. Create collaborative web sites using document libraries, wikis, blogs, forums, calendars, discussions, and social tagging
3. Integrate with external applications such as Liferay Portal, Adobe Flex, iPhone, iGoogle, and Facebook
4. Automate your business process with the advanced workflow concepts of Alfresco 3



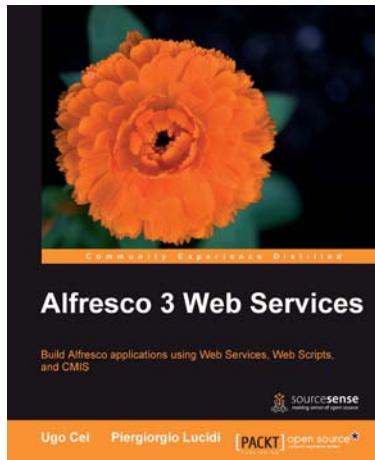
Alfresco Developer Guide

ISBN: 978-1-847193-11-7 Paperback: 556 pages

Customizing Alfresco with actions, web scripts, web forms, workflows, and more

1. Learn to customize the entire Alfresco platform, including both Document Management and Web Content Management
2. Jam-packed with real-world, step-by-step examples to jump start your development
3. Content modeling, custom actions, Java API, RESTful web scripts, advanced workflow
4. This book covers Alfresco Enterprise Edition version 2.2

Please check www.PacktPub.com for information on our titles

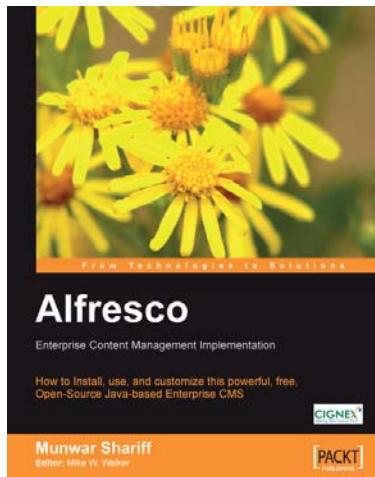


Alfresco 3 Web Services

ISBN: 978-1-849511-52-0 Paperback: 436 pages

Build Alfresco applications using Web Services, WebScripts and CMIS

1. Gain a comprehensive overview of the specifications of Web services
2. Implement the Alfresco specific Web Services
3. Get to grips with the Alfresco WebScripts and the Alfresco extensible RESTful API
4. Manipulate contents in Alfresco using different operations and APIs
5. Learn about the CMIS specification and its Alfresco implementation
6. Hands-on approach with examples built over the course of the book



Alfresco Enterprise Content Management Implementation

ISBN: 978-1-904811-11-4 Paperback: 356 pages

How to Install, use, and customize this powerful, free, Open Source Java-based Enterprise CMS

1. Manage your business documents: version control, library services, content organization, and search
2. Workflows and business rules: move and manipulate content automatically when events occur
3. Maintain, extend, and customize Alfresco: backups and other admin tasks, customizing and extending the content model, creating your own look and feel

Please check www.PacktPub.com for information on our titles