



Portal Administrator's Guide

Liferay Portal Administrator's Guide
by Richard L. Sezov, Jr. and Stephen Kostas
Copyright © 2010 by Liferay, Inc.
Put ISBN Number Here if this is a published work

This work is offered under the Creative Commons Attribution-Share Alike Unported license.



You are free:

- to share—to copy, distribute, and transmit the work
- to remix—to adapt the work

Under the following conditions:

- **Attribution.** You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- **Share Alike.** If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.

The full version of this license appears in the appendix of this book, or you may view it online here:

<http://creativecommons.org/licenses/by-sa/3.0>

Contributors:

Ray Auge, Jian Cao (Steven), Brian Chan, Alice Cheng, Bryan Cheung, Ivan Cheung, Shepherd Ching, Alexander Chow, Bruno Farache, Jorge Ferrer, Mike Han, Jeffrey Handa, JR Houn, Scott Lee, Wei Hong Ma (Sai), Charles May, James Min, Alberto Montero, Jerry Niu, Michael Saechang, Li Ji Shan (Dale), Ed Shin, Joseph Shum, Stephen Wilburn, Michael Young

Table of Contents

1. Introduction.....	15
ROBUST FUNCTIONALITY.....	16
AWARD-WINNING USER INTERFACE	17
MULTI-TENANCY VIA COMMUNITIES AND ORGANIZATIONS	17
FLEXIBLE ARCHITECTURE	17
ENTERPRISE APPLICATION INTEGRATION	17
SECURITY	17
HIGH AVAILABILITY AND HIGH PERFORMANCE	18
THEMING AND EASE OF BRANDING	18
READY INTEGRATION	19
CUSTOMIZATION AND EASE OF DEVELOPMENT.....	19
2. Initial Setup.....	21
EDITIONS OF LIFERAY.....	21
OBTAINING LIFERAY.....	22
INSTALLING A BUNDLE.....	23
INSTALLING LIFERAY FOR AN ENTERPRISE.....	25
SAMPLE DATA.....	25
LIFERAY HOME.....	25
DATABASE SETUP.....	26
DEFAULT METHOD: AUTOMATIC.....	26
MANUAL METHOD.....	26
TURNING A BUNDLE INTO AN ENTERPRISE PORTAL.....	27
THE PORTAL-EXT.PROPERTIES FILE.....	29
INSTALLING LIFERAY ON AN EXISTING APPLICATION SERVER.....	30
INSTALLING LIFERAY IN 10 EASY STEPS.....	30
GLASSFISH 3.X.....	32
JETTY 6.....	35
JBoss 5.X.....	37
RESIN 3.1.X.....	40
RESIN 3.2.X.....	42
TOMCAT 6.0.X.....	43
WEBLOGIC 10.....	45
ORACLE WEBLOGIC 10.3.....	49
WEBSPHERE 6.1.....	53
WEBSPHERE 7.0.....	57
MAKING LIFERAY COEXIST WITH OTHER JAVA EE APPLICATIONS.....	60
SUMMARY.....	61
3. Configuration.....	63
LIFERAY'S USER INTERFACE.....	63
NAVIGATING LIFERAY.....	64
NAVIGATING THE CONTROL PANEL.....	67
PORTAL ARCHITECTURE.....	69
USERS.....	70
USER GROUPS.....	70
ROLES.....	71
ORGANIZATIONS.....	71
COMMUNITIES.....	72
TEAMS.....	72
USING THE CONTROL PANEL.....	72
ADDING USERS.....	73
USER MANAGEMENT.....	75
ORGANIZATIONS.....	76

COMMUNITIES.....	78
SITE TEMPLATES.....	80
USER GROUPS.....	82
USER GROUPS AND PAGE TEMPLATES.....	82
ROLES.....	87
DEFINING PERMISSIONS ON A ROLE.....	88
SPECIAL NOTE ABOUT THE POWER USERS ROLE.....	91
TEAMS.....	91
GLOBAL SERVER SETTINGS.....	92
PASSWORD POLICIES.....	92
SETTINGS.....	93
GENERAL.....	94
AUTHENTICATION: GENERAL SETTINGS.....	94
AUTHENTICATION: LDAP.....	95
EXPORT.....	99
SINGLE SIGN-ON.....	102
AUTHENTICATION: CENTRAL AUTHENTICATION SERVICE (CAS).....	102
AUTHENTICATION: FACEBOOK.....	104
AUTHENTICATION: NTLM.....	104
AUTHENTICATION: OPENID.....	104
AUTHENTICATION: OPENSSO.....	105
AUTHENTICATION: SITEMINDER.....	106
USERS.....	106
MAIL HOST NAMES.....	107
EMAIL NOTIFICATIONS.....	107
IDENTIFICATION.....	107
MISCELLANEOUS: DISPLAY SETTINGS.....	107
CUSTOM FIELDS.....	107
MONITORING.....	108
PLUGINS CONFIGURATION.....	108
PAGE TEMPLATES.....	109
SITE TEMPLATES.....	110
SERVER ADMINISTRATION.....	111
RESOURCES.....	111
LOG LEVELS.....	111
PROPERTIES.....	112
CAPTCHA.....	112
DATA MIGRATION.....	112
FILE UPLOADS.....	113
MAIL.....	113
OPENOFFICE.....	113
SCRIPT.....	114
SHUTDOWN.....	114
PORTAL INSTANCES.....	114
PLUGINS INSTALLATION.....	115
SUMMARY.....	115
4. Web Content Management.....	117
PAGE CREATION AND MANAGEMENT.....	118
MANAGING PAGES.....	119
UNDERSTANDING PUBLIC AND PRIVATE PAGES.....	120
MANAGE PAGES INTERFACE.....	120
LOOK AND FEEL.....	121
EXPORT / IMPORT.....	122
SETTINGS.....	122
VIRTUAL HOST.....	123
LOGO.....	123
SITEMAP.....	123
MONITORING.....	124

STAGING	124
EDITING A PAGE.....	131
WHAT IS WEB CONTENT MANAGEMENT?.....	135
HOW CAN LIFERAY'S WCM HELP YOU?	135
WHAT FEATURES DOES LIFERAY WCM HAVE?	135
BUILDING A SITE WITH LIFERAY'S WCM.....	136
SIMPLE CONTENT CREATION.....	136
WEB CONTENT SECTION OF THE CONTROL PANEL.....	137
PUBLISHING CONTENT WITH THE WEB CONTENT DISPLAY PORTLET.....	140
ADVANCED CONTENT CREATION.....	143
STRUCTURES.....	144
TEMPLATES	148
ASSIGNING TEMPLATE PERMISSION.....	153
ADVANCED PUBLISHING OPTIONS.....	153
SCHEDULING WEB CONTENT	154
TAGS AND CATEGORIES.....	155
USING LIFERAY'S INTEGRATED WORKFLOW WITH CONTENT MANAGEMENT.....	157
DEFINING WORKFLOWS FOR WEB CONTENT	158
USING THE ASSET PUBLISHER PORTLET.....	159
QUERYING FOR CONTENT.....	160
ORDERING AND GROUPING.....	162
DISPLAY SETTINGS.....	163
SUMMARY.....	166
5. Liferay Collaboration Suite.....	167
SCOPES.....	168
ARCHIVED SETUPS.....	169
PERMISSIONS.....	170
SHARING.....	170
ANY WEB SITE.....	171
FACEBOOK.....	171
GOOGLE GADGET.....	172
NETVIBES.....	173
FRIENDS.....	173
BLOGS.....	173
THE BLOGS PORTLET.....	174
CONFIGURING THE BLOGS PORTLET.....	175
PERMISSIONS.....	176
ADDING BLOG ENTRIES.....	176
AGGREGATING BLOG ENTRIES.....	179
CALENDAR.....	180
CONFIGURING THE CALENDAR PORTLET.....	181
EMAIL FROM.....	181
EVENT REMINDER EMAIL.....	181
DISPLAY SETTINGS.....	182
USING THE CALENDAR PORTLET.....	182
CHAT	183
MAIL.....	184
MESSAGE BOARDS.....	186
GENERAL.....	186
EMAIL FROM.....	187
MESSAGE ADDED EMAIL.....	187
MESSAGE UPDATED EMAIL.....	187
THREAD PRIORITIES.....	187
USER RANKS.....	188
RSS.....	188
PERMISSIONS.....	188
ADDING CATEGORIES AND MAILING LISTS.....	189
USING THE MESSAGE BOARDS.....	190

POSTING NEW THREADS.....	191
MESSAGE BOARD ADMINISTRATIVE FUNCTIONS.....	192
MOVING THREADS.....	193
DELETING THREADS.....	193
BANNING USERS.....	194
SPLITTING THREADS.....	194
EDITING POSTS.....	194
PERMISSIONS.....	194
WIKIS.....	195
GETTING STARTED WITH THE LIFERAY WIKI.....	195
MANAGING WIKIS.....	197
ADDING AND EDITING WIKI PAGES.....	198
PAGE DETAILS.....	200
DETAILS.....	200
HISTORY.....	200
INCOMING / OUTGOING LINKS.....	201
ATTACHMENTS.....	201
NAVIGATING IN THE WIKI PORTLET.....	201
TAGS.....	201
CATEGORIES.....	202
SOCIAL EQUITY.....	204
SUMMARY.....	205
6. Advanced Liferay Configuration.....	207
THE PORTAL-EXT.PROPERTIES FILE.....	208
PROPERTIES OVERRIDE.....	208
LIFERAY HOME.....	209
PORTAL CONTEXT.....	209
RESOURCE REPOSITORIES ROOT.....	209
TECHNOLOGY COMPATIBILITY KIT.....	209
SCHEMA.....	209
UPGRADE.....	210
VERIFY.....	210
AUTO DEPLOY.....	210
HOT DEPLOY.....	213
HOT UNDEPLOY.....	213
SANDBOX DEPLOY.....	214
PLUGIN.....	214
PORTLET.....	214
PERSISTENCE.....	215
JPA.....	215
TRANSACTION MANAGER.....	216
PORTLET COORDINATION.....	217
THEME.....	218
RESOURCE ACTIONS.....	218
MODEL HINTS.....	219
SERVICE BUILDER.....	219
SPRING.....	219
HIBERNATE.....	220
JDBC.....	222
CUSTOM SQL.....	224
DATABASE.....	224
EHCAHCHE.....	225
JAVA SCRIPT.....	225
COMBO.....	227
SQL DATA.....	228
COMPANY.....	228
USERS.....	229
FACEBOOK CONNECTION.....	233

NTLM.....	233
REQUEST HEADER AUTHENTICATION.....	233
AUTHENTICATION TOKEN.....	233
GROUPS AND ROLES.....	234
ORGANIZATIONS.....	237
SECURITY MANAGER.....	238
BASIC AUTHENTICATION.....	238
LANGUAGES AND TIME ZONES.....	238
LOOK AND FEEL.....	240
AYOUTS.....	240
EDITORS.....	240
FIELDS.....	241
REQUEST.....	241
SESSION.....	242
HTTP.....	244
JAAS.....	244
LDAP.....	245
CAS.....	247
NTLM.....	248
OPENID.....	248
OPENSSO.....	248
SITEMINDER.....	249
AUTHENTICATION PIPELINE.....	249
AUTO LOGIN.....	252
SSO WITH MAC (MESSAGE AUTHENTICATION CODE).....	253
PASSWORDS.....	253
PERMISSIONS.....	255
CAPTCHA.....	257
STARTUP EVENTS.....	258
SHUTDOWN EVENTS.....	259
PORTAL EVENTS.....	259
LOGIN EVENT.....	260
LOGOUT EVENT.....	260
DEFAULT LANDING PAGE.....	260
DEFAULT LOGOUT PAGE.....	260
DEFAULT GUEST PUBLIC LAYOUTS.....	260
DEFAULT USER PRIVATE LAYOUTS.....	261
DEFAULT USER PUBLIC LAYOUTS.....	262
SANITIZER.....	263
SOCIAL EQUITY.....	263
VAADIN.....	264
DEFAULT ADMIN.....	264
AYOUTS.....	264
DEFAULT SETTINGS LAYOUTS.....	265
PORTLET URL.....	270
PREFERENCES.....	270
STRUTS.....	270
REDIRECT.....	270
IMAGES.....	271
FILESYSTEMHOOK.....	272
EDITORS.....	272
FIELDS.....	272
MIME TYPES.....	273
AMAZON.....	273
BROWSER LAUNCHER.....	273
CONTROL PANEL.....	274
INSTANT MESSENGER.....	274
LUCENE SEARCH.....	275

SOURCEFORGE.....	278
VALUE OBJECT.....	278
COMMUNICATION LINK.....	280
CLUSTER LINK.....	280
CLUSTER EXECUTOR.....	281
MINIFIER.....	281
MONITORING.....	282
MULTICAST.....	282
CONTENT DELIVERY NETWORK.....	283
COUNTER.....	283
LOCK.....	284
JBI.....	284
JCR.....	284
LIVE USERS.....	284
LOCK.....	285
MAIL.....	285
OPENOFFICE.....	287
POLLER.....	287
POP.....	287
QUARTZ.....	288
SCHEDULER.....	288
SEARCH CONTAINER.....	288
SHAREPOINT.....	288
SOCIAL BOOKMARKS.....	289
VELOCITY ENGINE.....	289
VIRTUAL HOSTS.....	290
HTTP.....	291
SERVLET FILTERS.....	291
UPLOAD SERVLET REQUEST.....	294
WEB SERVER.....	295
WEBDAV.....	295
MAIN SERVLET.....	295
AXIS SERVLET.....	295
GOOGLE GADGET SERVLET.....	296
JSON TUNNEL SERVLET.....	296
LIFERAY TUNNEL SERVLET.....	296
NETVIBES SERVLET.....	296
SPRING REMOTING SERVLET.....	296
WEBDAV SERVLET.....	296
WIDGET SERVLET.....	296
ADMIN PORTLET.....	296
ANNOUNCEMENTS PORTLET.....	297
ASSET PUBLISHER PORTLET.....	298
ASSET.....	298
BLOGS PORTLET.....	299
BREADCRUMB PORTLET.....	300
CALENDAR PORTLET.....	300
COMMUNITIES PORTLET.....	300
DISCUSSION TAG LIBRARY.....	301
DOCUMENT LIBRARY PORTLET.....	301
DOCKBAR PORTLET.....	302
FLAGS PORTLET.....	302
EMAIL NOTIFICATION SETTINGS.....	302
IFRAME PORTLET.....	302
IMAGE GALLERY PORTLET.....	303
LOGIN PORTLET.....	303
INVITATION PORTLET.....	303
JOURNAL PORTLET.....	303

JOURNAL ARTICLES PORTLET.....	306
JOURNAL CONTENT SEARCH PORTLET.....	306
MESSAGE BOARDS PORTLET.....	306
MY PLACES PORTLET.....	307
NAVIGATION PORTLET.....	308
NESTED PORTLETS PORTLET.....	308
PORTLET CSS PORTLET.....	308
SEARCH PORTLET.....	308
SHOPPING PORTLET.....	309
SOFTWARE CATALOG PORTLET.....	310
TAGS COMPILER PORTLET.....	310
TAGS PORTLET.....	310
TASKS PORTLET.....	310
TRANSLATOR PORTLET.....	311
WIKI PORTLET.....	311
PLUGIN MANAGEMENT.....	313
PORTLETS.....	313
THEMES.....	314
LAYOUT TEMPLATES.....	315
HOOK PLUGINS.....	315
WEB PLUGINS.....	315
INSTALLING PLUGINS FROM REPOSITORIES.....	316
INSTALLING PLUGINS MANUALLY.....	318
PLUGIN TROUBLESHOOTING.....	320
LIFERAY CONFIGURATION ISSUES.....	321
DEPLOY ISSUES FOR SPECIFIC CONTAINERS.....	323
CHANGING THE CONFIGURATION OPTIONS IN MULTIPLE PLACES.....	325
CREATING YOUR OWN PLUGIN REPOSITORY.....	325
THE SOFTWARE CATALOG.....	326
MANUALLY CREATING A SOFTWARE CATALOG.....	334
CONNECTING TO A SOFTWARE CATALOG.....	334
LIFERAY SERVICES ORIENTED ARCHITECTURE.....	334
ACCESSING LIFERAY'S WSDL.....	337
SUMMARY.....	338
7. Enterprise Configuration.....	341
LIFERAY CLUSTERING.....	342
ALL NODES SHOULD BE POINTING TO THE SAME LIFERAY DATABASE.....	344
DOCUMENT LIBRARY CONFIGURATION.....	344
DEFAULT FILE SYSTEM HOOK.....	344
JACKRABBIT SHARING.....	344
OTHER STORAGE OPTIONS.....	345
SEARCH CONFIGURATION.....	347
PLUGGABLE ENTERPRISE SEARCH.....	347
LUCENE CONFIGURATION.....	349
HOT DEPLOY.....	351
DISTRIBUTED CACHING.....	351
HIBERNATE CACHE SETTINGS.....	353
CLUSTERING JACKRABBIT.....	354
WORKFLOW WITH KALEO.....	355
INSTALLATION.....	355
KALEO WORKFLOW IN A NUTSHELL.....	356
PROCESS DEFINITIONS.....	356
WORKFLOW IN THE CONTROL PANEL	364
INTEGRATING WITH USERS, COMMUNITIES, ORGANIZATIONS AND ROLES.....	366
USING KALEO WORKFLOW PROCESSES IN LIFERAY PORTAL.....	367
PERFORMANCE TUNING.....	368
MEMORY.....	368
GARBAGE COLLECTION.....	369

PROPERTIES FILE CHANGES.....	372
SERVLET FILTERS.....	372
PORTLETS.....	372
READ-WRITER DATABASE CONFIGURATION.....	373
DATABASE SHARDING.....	374
SUMMARY.....	376
8. Maintaining A Liferay Portal.....	377
LIFERAY MONITORING USING GOOGLE ANALYTICS.....	377
BACKING UP A LIFERAY INSTALLATION.....	378
SOURCE CODE.....	379
LIFERAY'S FILE SYSTEM	379
DATABASE.....	379
LIFERAY'S LOGGING SYSTEM.....	380
UPGRADING LIFERAY.....	382
LIFERAY UPGRADE PROCEDURE.....	383
UPGRADE STEPS.....	383
UPGRADING FROM LIFERAY 5.1 TO LIFERAY 5.2.....	384
PREREQUISITE	384
CHANGES IN CONFIGURATION PROPERTIES	384
THEME UPGRADE	385
API CHANGES	386
UPGRADING FROM LIFERAY 5.2 TO LIFERAY 6.0.....	386
PREREQUISITE.....	387
UPGRADING YOUR PERMISSIONS ALGORITHM.....	387
UPGRADING EXT TO EXT PLUGINS.....	388
SUMMARY.....	388
9. Appendix: Documentation License.....	391
CREATIVE COMMONS LICENSE.....	391
LICENSE.....	391
CREATIVE COMMONS NOTICE.....	399
10. Colophon.....	401

PREFACE

Liferay Portal is the leading open source portal in the marketplace today. It has received awards from multiple leading industry publications, and has an impressive download rate (over 60,000 downloads a month and over a million downloads total). Why is it so popular? Because Liferay Portal has *out of the box* all of the features you need to run a successful web site, whether that site is a public Internet site, a corporate Intranet, a social network, or anything in between.

This book was written for anyone who has any part in setting up, using, or maintaining a web site built on Liferay Portal. It will guide you step-by-step through the installation, configuration, and use of Liferay Portal. Use this book as a handbook to getting your Liferay Portal installation running smoothly, and then keep it by your side as you configure and maintain your Liferay-powered web site.

The information contained herein has been organized in a way that makes it easy to locate information. We start at the beginning: downloading and configuring the Liferay bundles. From there, we work all the way through the multiple ways of installing Liferay manually on an application server, to portal administration. After this, we cover the many features of Liferay Portal, including its web content management system, its collaboration suite, and the Kaleo workflow engine. From there we go into advanced administration topics and enterprise configuration, including clustering and integrating Liferay with other services, such as search. We round things out by showing you how to optimize Liferay's performance, how to manage a

Liferay installation, how to back it up, and how to upgrade Liferay if you are moving from a previous version.

What's New in the Fourth Edition

Liferay Portal version 6 is a significant upgrade from the last release. This edition of the book has been exhaustively overhauled so that it covers Liferay Portal version 6 as well as contains material that hasn't been covered before. Of course all of the new portal properties are covered, upgrading and installing the new version is covered, and the rest of the book has been exhaustively gone through and updated.

You'll be treated to an expanded control panel in Liferay Portal 6, and every option is detailed and explained here. You'll gain an understanding of all of the constructs within Liferay, including users, roles, communities, organizations, user groups, and now teams.

You'll be introduced to Liferay portal's collaboration suite which includes like blogs, calendar, chat, mail, message boards, and wikis. Your users will be able to collaborate like never before using this suite of integrated applications.

But more than that, for the first time we've been able to provide documentation on the use of Liferay portal's web content management suite. Now you'll have a ready reference showing you how to load and maintain your site's content without having to guess.

With all of this new stuff, the book is a bit thicker. But we've tried to keep it to a manageable size by excising material which applied to Liferay 4.x and below. The result is what you have in your hands: a lean, mean guide chock full of information that is relevant to you as you build your site on Liferay portal.

Conventions

Sections are broken up into multiple levels of headings, and these are designed to make it easy to find information.



Tip: This is a tip. Tips are used to indicate a suggestion or a piece of information that affects whatever is being talked about in the surrounding text. They are always accompanied by this gray box and the icon to the left.

Source code and configuration file directives are presented monospaced and in gray boxes, as below.

```
If source code goes multi-line, the lines will be \
separated by a backslash character like this.
```

Italics are used to represent links or buttons to be clicked on in a user interface.

Monospaced type is used to denote Java classes, code, or properties within the text.

Bold is used to describe field labels and portlets.

Page headers denote the chapters, and footers denote the particular section within the chapter.

Publisher Notes

It is our hope that this book will be valuable to you, and that it will be an indispensable resource as you begin to administer a Liferay portal server. If you need any assistance beyond what is covered in this book, Liferay, Inc. offers training, consulting, and support services to fill any need that you might have. Please see <http://www.liferay.com/services> for further information about the services we can provide.

It is entirely possible that some errors or mistakes made it into the final version of this book. Any issues that we find or that are reported to us by the community are documented on the Official Liferay Wiki. You can view them or contribute information that you've found about any issues here:

<http://www.liferay.com/community/wiki/-/wiki/Main/Liferay+Administrator's+Guide+Errata>

As always, we welcome any feedback. If there is any way you think we could make this book better, please feel free to mention it on our forums. You can also use any of the email addresses on our *Contact Us* page (<http://www.liferay.com/contact-us>). We are here to serve you, our users and customers, and to help make your experience using Liferay Portal the best it can be.

Author Notes

RICHARD SEZOV:

The first edition of this book was outlined in a small notebook (paper, not a computer) on a plane flying from Philadelphia to Los Angeles. Covering Liferay 4.3, it was the first actual, *physical* book Liferay produced. The book has come a long way since then, and somehow we're able to get it to be more comprehensive with each edition.

For this fourth edition, for the first time we have multiple authors. We did that in order to cover the material we needed to cover in such a short time, and also because in parallel, I am working on another book for developers called *Liferay in Action*. I very much appreciate the help from my co-author, and am glad for his help. This book could not have come together without him. I also want to especially thank Stephen Wilburn for his contributions.

I have endeavored to give credit to everyone who made a contribution (it's on the copyright page), but if I missed somebody—which would not be surprising—please let me know so your name is not left out of the next edition! I cannot express enough how wonderful it is to be surrounded by so

many talented people who do everything they can to make this product the best it can be—even when a particular task is not their primary job.

The engineering team at Liferay is a fantastic group of people, and the documentation team is highly indebted to them because we pepper them with so many questions all the time. So special thanks are due to Ray Auge, Julio Camarero, Nate Cavanaugh, Brian Chan, Alex Chow, Bruno Farache, Jorge Ferrer, and Mike Young. I also have to especially thank Cynthia Wilburn, who keeps us all on track and who somehow juggles a huge incoherent pile of stuff that needs to get done into an organized plan that we can all follow.

I'd also like to thank my daughter Julia for the brightness of her countenance and her marvelous ability to tell me when work is done and it's time to play. And of course, I want to thank my wife, Deborah, who continually has to put up with long hours as a computer widow, for her understanding and support—especially for this edition. I couldn't do any of this without her.

Rich Sezov

<http://www.liferay.com/web/richard.sezov/blog>

STEPHEN KOSTAS:

As this book comes to completion, I'm approaching my one year anniversary as a member of the Liferay team. Being part of Liferay has meant working with a great group of people from all over the world, each doing their part to bring us to the release of Liferay 6. It has also meant a lot of hard work, working with Rich, the other Steve, and the engineering team to make sure that this 4th edition of the Administrator's Guide provides everything that you need to get Liferay working for you.

I suppose that this is where I should thank people, so thanks to Rich, for all of his toil and late nights spent putting this together, my wife Dana, for putting up with me paying attention to my laptop instead of her, and the engineering team, for answering our questions, providing us with all sorts of useful information, and of course, building Liferay.

Stephen Kostas

<http://www.liferay.com/web/stephen.kostas/blog>

1. INTRODUCTION

Liferay Portal is the world's leading open source enterprise portal solution using the latest in Java and Web 2.0 technologies. Now in its 11th year of development, the award-winning Liferay Portal is one of the most widely deployed portal technologies on the market, with an estimated 250,000 deployments worldwide.

More than a portal, Liferay is a platform for creating effective business applications and solutions. It offers a robust feature set, impressive scalability, time-saving development tools, support for over 30 languages, and a flexible, scalable architecture that is open source developed and enterprise refined. Notable differentiators include:

- Compatible with all major databases, operating systems, and app servers
- Hierarchical system of communities and organizations
- Granular, delegable permissioning system
- Highly scalable, supporting more than 5,000 concurrent transactions (33,000 simultaneous users) per server
- Real-world performance of millions of page views and 1.3 million users
- Supports 22 languages, out-of-the-box
- Award-winning user interface

Robust Functionality

The product is also known for robust functionality with over 60 out-of-the-box portlets that can be categorized into three main categories:

1. Content Management & Web Publishing

Liferay Portal's built-in content management system is a structured system with support for roles-based workflow, separation of presentation from content creation, and editorial approval and versioning processes. Its document library features versioning, document locking, and automatic file-type conversions for document-based collaboration, as well as WebDAV, and Microsoft Office® integration for dynamic document sharing. Content may also be stored on third-party systems such as Jackrabbit, Magnolia, and Alfresco. What's more, users can seamlessly integrate content with native collaboration and social networking features, as well as third party applications, to publish full-bodied enterprise solutions to the web.

2. Collaboration

Liferay's collaboration suite includes Message Boards, Blogs, Wikis—featuring RSS capabilities, tagging, common meta-data, and social bookmarking—that work within Liferay Portal's flexible system of communities and organizations. As a whole, they enable productive discussion around your collective knowledge; not only can users manage their own work experiences with our built-in webmail client and personal calendars, they can dynamically share their work and thoughts as part of a team.

3. Social Networking

Tying our content and collaboration features together is an enterprise-ready social networking suite with presence-enabled features like instant messaging and activity tracking that facilitate fluid, real-time communication within your organization. Moreover, Liferay's Social API gives users a platform for building their own social applications. Users can plug into Liferay's social capabilities and allow third party applications to take part in Liferay's activity feeds, member and "friend" lists, and other social assets, and customize these assets per their specific needs. In essence, Liferay provides you with the tools and framework for building a fully functional social network that can be customized to meet your unique specifications.

Award-winning User Interface

Liferay offers a rich, easy-to-use “Web 2.0” interface using AJAX and other presentation layer technologies. It features effortless GUI-based personalization, drag-and-drop portlets, dynamic navigation and breadcrumb features, and an instant-add portlet library. The portal platform also integrates with YUI3, jQuery or your JavaScript library of choice.

Multi-tenancy via Communities and Organizations

Liferay Portal gives enterprises the ability to organize users by business-defined categories such as departments, geographies, or offices, as well as by cross-departmental teams and workgroups. Each community and organization can inherit and define its own permissions and administer user, group, and role management for the various portlets it chooses to include.

Virtual hosting allows them each to apply their own individual friendly URLs, themes, and configurations, independent of the others.

Flexible Architecture

Organizations choose Liferay Portal for the flexibility of its architecture and the ease of integration. Thanks to our use of Service Oriented Architecture, users get accelerated development cycles, re-usable services, and composite application development.

Enterprise Application Integration

Moreover, Liferay Portal is an open framework with a completely exposed API supporting web services (SOAP), JSON, RMI and our own proprietary tunneling classes. As long as your existing applications are open and provide support through some service layer or API, Liferay can integrate with those applications.

There is a range of options for application integration depending on your needs, from web services and a Liferay iFrame portlet for lighter integration scenarios, to a web proxy or WSRP for more complex integration scenarios.

Security

Liferay Portal uses industry standard, government-grade encryption technologies, including advanced algorithms such as DES, MD5, and RSA, and was benchmarked as among the most secure portal platforms using LogicLibrary’s Logicscan suite. It offers a customizable single sign-on (SSO) that integrates with Yale CAS, JAAS, LDAP, Netegrity, Microsoft Exchange, and more.

What's more, Liferay Portal ships with robust user management and security features including password policies, user reminder settings, and complete log-in security procedures. Liferay also abides by OWASP guidelines to reduce the risk of security vulnerabilities. Other security features include:

- Pluggable Authentication
- Email Verification
- Session Management

High Availability and High Performance

Liferay Portal has been tested to support more than 3,000 concurrent transactions (33,000 simultaneous users) on a single 8-core application server, with mean login times under one second and maximum % throughput of 79+ logins per second. Our Web Content Management (WCM) system scales to beyond 150,000 concurrent users on a single Liferay Portal server with average transaction times under 50ms and 35% CPU utilization; In high-traffic web-publishing scenarios, Liferay Portal has proven to handle millions of page views and over 1.3 million users. Additionally, in collaboration and social networking scenarios, each physical server supports over 1300 concurrent users at average transaction times of under 800ms.

Liferay Portal is also deployable to the Cloud and virtual server environments and ensures high availability and performance with:

- Hardware/Software Load Balancing, HTTP Failover, Session Replication
- Distributed Cache using Lightweight Multicast Protocol
- Terracotta, Oracle RAC, and other scalability solutions

Theming and Ease of Branding

Theming in Liferay Portal allows you to create dynamic sites of any kind, from traditional portals to heavily branded solutions that may not resemble a portal at all. Via the creation and dynamic inheritance of CSS and Javascript templates, you have full control over the look-and-feel of your site without actually having to modify any code within the portal or portlets. Since all components of the Liferay SDK (Themes, Hooks, Layout Templates and Portlets) are hot deployable to your Liferay environment, you can install and change these customizations while the portal is still running. This makes for impressive flexibility in customizing Liferay Portal, extremely painless updates, as well as notable savings in development time.

Ready Integration

As the only independent enterprise portal on the market, Liferay is fully committed to keeping its platform compatible with all major databases, operating systems, and application servers. Liferay Portal is built with standard integration technologies including JSR 168/286, JSR 170, WebDav, iCal and WSRP. The core source is comprised of open source frameworks such as Struts, Spring and Hibernate and the product runs PHP, Ruby, Python, Grails and other lightweight scripting technologies within a robust Java framework.

Thanks to its adherence to open standards and this flexible architecture, Liferay Portal integrates easily with both open source and mainstay proprietary products such as Documentum, Microsoft Office®, Alfresco, Intalio, JasperSoft, Magnolia, MuleSource, Pentaho, Terracotta, and more.

Customization and Ease of Development

Created, designed, and honed by developers, Liferay Portal places high value on the ease of development and offers its own SDK. It includes tools such as Liferay Service Builder that automatically generates data tiers (SQL and Hibernate logic / classes), Spring Dependency Injection wiring, and Web Service access. This frees your development team to focus on higher priority matters like business logic. Meanwhile, Liferay Hooks give you access to modify core Liferay functionality without modifying Liferay's core source code. Liferay's extensible development environment permits an easy upgrade path for your customizations because it creates a clean separation between the platform and your modifications. The modular nature of our hooks results in less code to maintain and fewer scenarios to test. Other features include:

- Industry-standard Struts/Tiles MVC framework
- Integrated Kaleo workflow engine allowing developers to define dynamic business processes (i.e., publishing and editorial approval, user registration, e-commerce transactions)
- Spring framework for easy transaction management
- Support for other frameworks including JSF and IceFaces, Wicket, Spring MVC, and others

2. INITIAL SETUP

Liferay Portal is one of the most flexible applications with regard to application server environment on the market today. You can install Liferay Portal on everything from a shared Tomcat installation to a multi-node cluster running a commercial application server, and on everything in between. In fact, Liferay is used successfully in all of these scenarios every day.

You will find that because Liferay is extremely flexible in its deployment options, it is easy to install as well. If you already have an application server, you can simply use the tools for deployment that came with your application server. If you do not have an application server, Liferay provides several application server bundles from which to choose. These are very easy to install and with a small amount of configuration can be made into production-ready systems.

Editions of Liferay

Liferay ships in two different editions: Liferay Portal Community Edition (CE) and Liferay Portal Enterprise Edition (EE). CE is the same Liferay Portal that has been available for years: frequently updated and bursting with the latest features, the Community Edition of Liferay Portal is offered for free under the Lesser GNU public license, an open source license. This license gives you the flexibility to link Liferay with your own code in your portlet, theme, hook, layout, Ext, or web plugins, no matter what license you use for your code. If, however, you modify Liferay directly, those modifications need to be contributed back to the open source product. This is really the best of both worlds: you have the freedom to do what you want with your code, and Liferay receives the benefits of any enhancements that are made directly.

Liferay Portal EE is a supported version of Liferay Portal for the enterprise. Hardened for security and designed to be rock solid stable, EE is offered with a subscription and support package, allowing organizations to build their portals on a stable version of the product that is offered over an extended period of time.

Because the release cycle for EE is longer than that for CE, each enterprise release is supported for 4 years. All bug fixes in Liferay Portal are backported to your version of Liferay for the duration of your subscription. This gives organizations the peace of mind that comes from knowing that their Liferay-powered web sites are stable and will run for years to come, enabling them to build their sites on a proven, stable platform. Additionally, Liferay's professional services team offers training and consulting on the Enterprise Edition to ensure long-term support and stability for our clients.

Obtaining Liferay

The CE version of Liferay is freely downloadable from our web site at <http://www.liferay.com>. Click on the *Downloads* link at the top of the page, and you will be presented with multiple options for getting a copy of Liferay, including our convenient bundles or a .war package for installation on your application server of choice.

The EE version of Liferay is provided to you as a result of your support subscription. You will receive download links which will allow you to obtain a copy of a Liferay bundle or a .war package for installation on your application server of choice.

So what is a bundle anyway? A bundle is simply an open source application server with Liferay preinstalled. If you want to install a bundle, there is a list of bundles available. If you do not currently have an application server, it is best to download the Tomcat bundle, as Tomcat is one of the smallest and most straightforward bundles to configure. If you have an application server preference, you can also choose the server you prefer from the available Liferay Portal bundles. All of the bundles ship with a Java Runtime Environment for Windows; if you are using a different operating system, you will need to have a JDK (Java Development Kit) installed prior to launching Liferay.

Please note that Liferay is not able to provide application server bundles for proprietary application servers such as WebLogic or WebSphere, because the licenses for these servers do not allow for redistribution. Liferay Portal, however, runs just as well on these application servers as it does on open source application servers. You will need to use the .war package to install Liferay on these application servers.

For a manual install, you will need the Liferay .war file as well as Liferay's dependency .jars. Later in this chapter are instructions for installing Liferay on many of the major application servers available today.

Installing a Bundle

Liferay bundles contain the same directory structure regardless of application server. The top-level folder is named for the release of Liferay. This folder is also sometimes called *Liferay Home*.

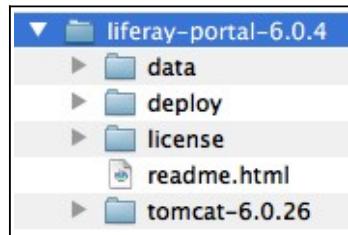


Illustration 1: Bundle directory structure

Inside this folder, you will find folders for various uses:

Data: This folder is used to store the embedded HSQL database which the bundles use, as well as the configuration and data for the Jackrabbit JSR-170 content repository and the Lucene search index.

Deploy: Plugins which you wish to deploy to Liferay can be copied into this folder. It is also used by Liferay's graphical plugin installer utility, which is available from the Control Panel.

License: Contains both Liferay's license and a file which describes the licenses for many of the other open source projects that are used internally by Liferay.

[Application Server]: There will also be an application server folder which is different depending on which bundle you have downloaded. This folder contains the application server in which Liferay has been installed.

In most cases, installing a bundle is as easy as uncompressed the archive and then starting the application server. For example, if you were to install Liferay Portal on Tomcat, you would simply unzip the bundle to a location of your choice.

Now you would start Tomcat in the same way as you would if you had downloaded it manually. Tomcat is launched by way of a script which is found in its *bin* folder. If you drop to a command prompt and go to this folder, you can launch Tomcat via the following command on Windows:

```
startup
```

or the following command on Linux / Mac / Unix:

```
./startup.sh
```

The Liferay / Tomcat bundle will then launch. If you are on Windows, you will see another command prompt window appear with Tomcat's console

Initial Setup

in it. If you are on Linux, you can see the Tomcat console by issuing the following command:

```
tail -f ../logs/catalina.out
```

Once Tomcat has completed its start up, it should automatically launch a web browser so you can see the home page. If it does not, launch your web browser and then go to the following address: <http://localhost:8080>. The default Liferay home page will then appear in your web browser. It will be using an embedded database for its configuration, but it is fully functional. You can now begin exploring the various features of Liferay.

Liferay ships by default with a sample web site included, called 7 Cogs. You can access this site and log in as the various users to get familiar with Liferay and what it can do.

Installing a different bundle is done in exactly the same way: unzip the bundle into the folder of your choice, launch the application server, and then view the portal in your web browser.

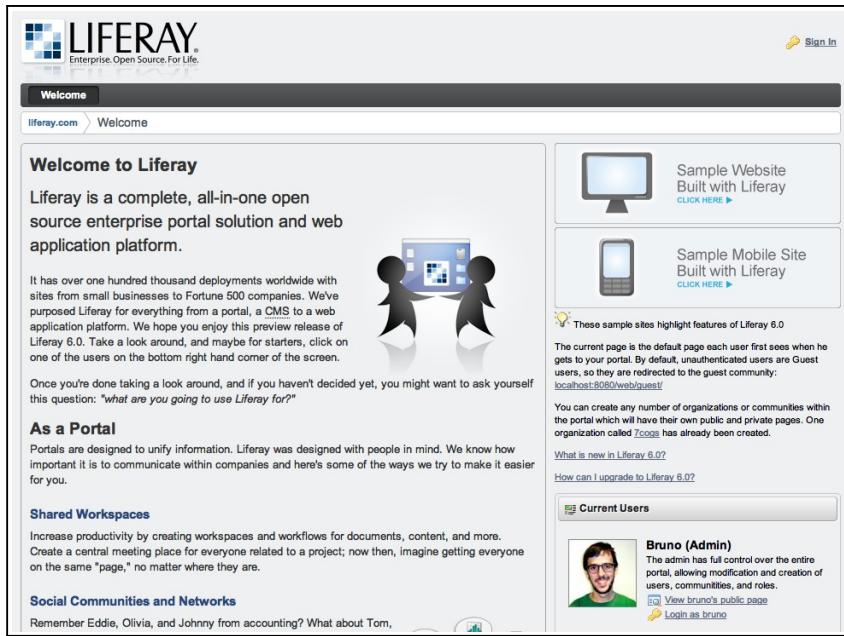


Illustration 2: Default Page in the Liferay Bundles

As you can see, bundles are the easiest way to get started with Liferay. They come pre-configured with a running Liferay that can be used immediately to explore all of the things that Liferay can do. And with minimal extra configuration (which we will see later), bundles can be converted into full production-ready systems.

Installing Liferay for an Enterprise

When it comes time to install Liferay Portal on your server, you'll find it is easiest to do this by starting with a bundle and then reconfiguring that bundle so that it is enterprise-ready. Because this is by far the quickest and easiest method to get a production Liferay system running, we will look at this first. Often, however, enterprises will have an established Java EE infrastructure upon which they would like to install Liferay. In this situation, a bundle will not suffice. Most of the rest of this chapter, therefore, will focus on installing Liferay onto an already-established application server.

Sample Data

Liferay CE ships with some sample data to help you see some of the things Liferay Portal can do. While the sample 7 Cogs data is a good example of how Liferay might be used, when you are ready to build your own site, you won't want that data cluttering up your database. So before you connect Liferay to your production database, you will want to make sure you have removed the sample 7 Cogs data from your Liferay installation. This is as simple as undeploying the application that installs the 7 Cogs data.

There is only one application included in the bundle that you will need to remove. It's a hook that copies the 7 Cogs data into the database when Liferay is started. Because we want to revert Liferay's behavior back to its defaults for a clean install, you will want to remove the *sevencogs-hook* application. The other two applications related to 7 Cogs are both themes, which you can leave installed if you wish.

If you forget to undeploy the *sevencogs-hook* application before you connect Liferay to your real database, the sample data will be created in your database and may cause issues, especially if you already have data in your database. So you want to make sure that you get *sevencogs-hook* undeployed before setting up your server. Use your application server's method for uninstalling applications in order to remove them.

Liferay Home

Liferay Portal uses a special folder defined as *Liferay Home*. This folder is one folder higher than the location of the application server itself. This is why the bundles place the application server one folder in from the bundle's root folder.

If Liferay is unable to create the resources it needs in this folder, or if it finds itself running on certain application servers, it will fall back to defining the home folder in the home folder of the user ID that is running Liferay.

As described above in the *Bundles* section, the home folder is very important to the operation of Liferay. The aforementioned folders (*data*, *deploy*, and *license*) will be created there, and you can also put a special configuration file called *portal-ext.properties* there.

This file is fully documented in Chapter 6: *Advanced Liferay Configuration*, but we will use it in this chapter for some basic configuration, including setting up Liferay to talk to our database.

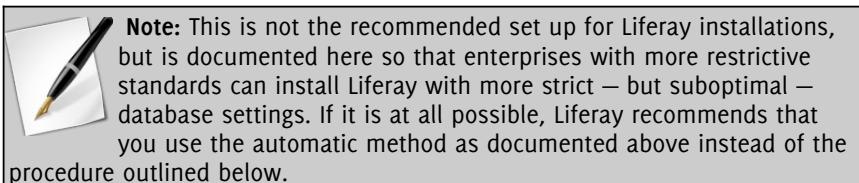
Database Setup

Default Method: Automatic

If you create your database and grant a user ID full access to it, Liferay can use that user ID to create its indexes and tables automatically. This is the recommended way to set up Liferay, as it allows you to take advantage of Liferay's ability to automatically maintain its database during upgrades or through various plugin installs which may create tables of their own. It is by far the best way to set up your Liferay installation.

If you will be setting up Liferay's database with the recommended permissions, you can skip the next section.

Manual Method



Even though Liferay can create its database automatically, some enterprises prefer *not* to allow the user ID configured in an application server to have the permissions over the database necessary for Liferay and its plugins to maintain their tables. For these organizations, Select, Insert, Update, and Delete are generally all the permissions that are granted, and so we will go over how to set up the database manually. If your organization is willing to grant the Liferay user ID permissions to create and drop tables in the database—and this is the recommended configuration—you can skip this section.

One other caveat is this: Liferay has an automatic database upgrade function which runs when the version of Liferay is upgraded to a new release. If the user ID that accesses the database does not have enough rights to create /

modify / drop tables in the database, you will need to grant those rights to the ID before you start your upgraded Liferay for the first time. Once the upgrade is complete, you can remove those rights until the next upgrade. Additionally, many plugins provided by Liferay require that new tables be added to Liferay's database. These plugins cannot be installed if Liferay does not have permission to create these tables automatically. If you wish to install these plugins, you will need to grant rights to create tables in the database before you attempt to install them.

Liferay provides an SQL script archive download on the web site. For the CE version, it is in the *Additional Files* section of the Downloads page. For the EE version, you will be provided a link to this archive. Download this file and unzip it. You will find that it contains a folder structure that is broken down by the type of script (full, minimal, or upgrade), and then further by database vendor type.

It is best to use the `create-minimal` script if you are installing a fresh version of Liferay on a development, QA, or production server. This script creates the necessary Liferay tables in the database, with a minimum configuration. This is most appropriate for a new installation of Liferay.

The `create` script, by contrast, configures a Liferay database with a portion of the content from <http://www.liferay.com> embedded in it. This can be useful from a development perspective, as it contains working examples of the use of many of Liferay's features, including the Content Management System.

Inside the `create` or `create-minimal` folders are scripts for every database that Liferay supports. A DBA can use the script provided to create the Liferay database, complete with the indexes necessary for optimal performance. Once this is done, be sure that the ID that the portal will use to connect to the database has at least Select, Insert, Update, and Delete permissions. Preferably, however, the ID should also have rights to create, modify, and drop tables and indexes, as this makes upgrading easier. This, however, is not necessary for the daily operation of Liferay.

Once your DBA has created the database and provided the credentials for accessing it, you are ready to begin 1) making a bundle enterprise-ready or 2) manually installing Liferay on your application server.

Turning a Bundle into an Enterprise Portal

Liferay Portal is distributed with the following bundle options for servlet containers and full Java EE application servers:

- Geronimo+Tomcat
- Glassfish 3

- JBoss
- Jetty
- JOnAS
- Resin
- Tomcat 6.0

Choose your preferred bundle and download it from the downloads page on Liferay's web site or via the EE links that were provided to you. A prerequisite for running any of the bundles is that you have the proper version of the Java Development Kit (1.5 or higher) installed on the machine to which you are installing Liferay. Make sure that you have also created the `JAVA_HOME` environment variable and have pointed it to your Java installation.

Unzip the bundle to the location from which you are going to run it. For example, you might use `D:\apps` in Windows or `/opt` in Linux or UNIX variants. The default bundle installation of Liferay Portal uses an embedded database. While this is a good method to have it up and running fast for evaluation or development, it has several drawbacks:

- Only one user can access it at a time. This is because the data is stored on a file on disk and HSQL locks it when doing changes.
- The data is stored inside the bundle and might be lost on re-deployment.
- This configuration does not scale well and will have performance problems when multiple users are accessing the system.

Obviously, you do not want to be running Liferay against the embedded database. Fortunately, Liferay has great support for a good number of production-ready databases, and it is easy to configure Liferay to use them. The exact instructions will depend on the application server and database, but can be summarized as:

1. Create the database in your DBMS of choice (see the above section labeled *Database Setup* for further information).
2. [Optional] Create a Data Source called `jdbc/LiferayPool` in your application server which points to your database and has the proper credentials to access it.
3. [Optional] Create a mail session called `mail/MailSession` in your application server which points to your mail server, so Liferay can send mail.

4. Create a `portal-ext.properties` file in the Liferay Home folder which either points directly to the database and mail session or points to the application server's Data Source and mail session.
5. Start Liferay. Liferay will create the tables automatically and start. Otherwise, you will have had to prepare the database first by running the appropriate create script.

Refer to the manual installation instructions below for further details on configuring the various application servers. There is no difference between the Liferay bundles and the regular distribution archives of the application servers as they are available from their own sites, with the exception that Liferay is pre-installed in them, and the JVM settings may have been optimized for use with Liferay.

The `portal-ext.properties` File

To point your Liferay bundle to your database, create a file called `portal-ext.properties` in your Liferay Home folder. This file overrides default properties that come with Liferay. You are going to override the default configuration which points Liferay to the embedded HSQL database.

There are two ways to set up the connection:

- Use your application server's connection pool.
- Use the built-in connection pool.

If you want to use your application server's connection pool, you will have to create one in your application server that points to your database. It should be called `jdbc/LiferayPool`. To cause Liferay to use this connection pool, add the following directive to your `portal-ext.properties` file:

```
jdbc.default.jndi.name=jdbc/LiferayPool
```

To use the built-in connection pool—based on C3P0—add the template which is provided in Chapter 6 for your particular database. The template for MySQL is provided as an example below.

```
#  
# MySQL  
#  
jdbc.default.driverClassName=com.mysql.jdbc.Driver  
jdbc.default.url=jdbc:mysql://localhost/lportal?  
useUnicode=true&characterEncoding=UTF-8&useFastDateParsing=false  
jdbc.default.username=  
jdbc.default.password=
```

You would provide the user name and password to the database as values for the `username` and `password` directives.

For mail, there is a similar procedure. Again, you have two ways to configure your server:

- Use your application server's mail session.
- Use the built-in mail session.

To use your application server's mail session, you will have to create one in your application server that points to your mail server. Once you have done that, add the following directive to your portal-ext.properties file:

```
mail.session.jndi.name=mail/MailSession
```

To use the built-in mail session, add the following directives to your portal-ext.properties file, substituting your mail server information:

```
mail.session.mail.pop3.host=localhost  
mail.session.mail.pop3.password=  
mail.session.mail.pop3.port=110  
mail.session.mail.pop3.user=  
mail.session.mail.smtp.auth=false  
mail.session.mail.smtp.host=localhost  
mail.session.mail.smtp.password=  
mail.session.mail.smtp.port=25  
mail.session.mail.smtp.user=  
mail.session.mail.store.protocol=pop3  
mail.session.mail.transport.protocol=smt
```

Save the file. You can now start your application server.

Installing Liferay on an Existing Application Server

This section contains detailed instructions for installing Liferay Portal using its WAR distribution. This allows system administrators to deploy Liferay in existing application server installations. It is recommended that you have a good understanding of how to deploy Java EE applications in your application server of choice.

Installing Liferay in 10 Easy Steps

There are 10 generic steps to installing Liferay on an existing application server:

1. Obtain the Liferay .war file and the dependencies archive.
2. Make sure you do not have an application listening at the root (/) of your server. If you do, move it to a different context or undeploy it.
3. Decide whether you want to use your application server's data sources or if you want to use the one included with Liferay. If you

want to use your application server's data source, create a data source for Liferay called `jdbc/LiferayPool`.

4. Decide whether you want to use your application server's mail session or if you want to use the one included with Liferay. If you want to use your application server's mail session, create one called `mail/MailSession`.
5. Shut your application server down.
6. Extract the dependencies to a location on your server's global class path. This allows both Liferay and plugins to access these dependencies.
7. Create a `portal-ext.properties` file and place it in the Liferay Home folder. See the notes on specific application servers below for the location of this folder for your application server.
8. Add either the JNDI name of your data source or the JDBC parameters above to connect Liferay to your database.
9. Add either the JNDI name of your mail session or the mail parameters above to connect Liferay to your mail server.
10. Start your application server, deploy the Liferay .war file, and start it.

The instructions below are specific for each application server that Liferay supports. Liferay supports a wide combination of application servers and databases. Because of this, for brevity this section assumes MySQL as the database, that the database has already been created, and that you are using your application server's mail session and data source. To use other databases, substitute the JDBC driver and URL construct for your database in place of the MySQL ones shown here.

We also assume your application server is already installed and running successfully. If you still need to install your application server, please follow your vendor's instructions first.

The following instructions assume an installation on a local machine. When installing to a remote server, substitute localhost with the host name or IP of the server.



Tip: Note that Liferay 5.x and above *requires* JDK 1.5 or greater. Do not attempt to install Liferay 6.x on an application server that runs under Java 1.4 or lower; it will not work. If you are running an application server that ships with a JDK and that JDK is 1.4 or lower, you will need to upgrade your application server in order to run current versions of Liferay Portal. Liferay 4.x, however, will run fine on these application servers.

Remember, for all of these application servers, create your portal-ext.properties file in the Liferay Home folder and make sure it points to your database connection pool and mail session.

GlassFish 3.x

Liferay Home is in the Glassfish root folder. We will assume for these instructions that you are using the default domain stored in [GlassFish Root]/glassfish/domains/domain1.

1. Before starting GlassFish, you will need to modify some settings in the domain you will be using to increase the default amount of memory available. In your domain folder is a config folder. Open the file called domain.xml in this folder.
2. At approximately line 166 of this file, you will find the following JVM option being set:

```
<jvm-options>-Xmx512m</jvm-options>
```

Change this to:

```
<jvm-options>-Xmx1024m</jvm-options>
```

3. Add another line after this line with the following JVM option:

```
<jvm-options>-XX:MaxPermSize=256m</jvm-options>
```

Save and close the file.

4. In your domain folder is a folder called docroot. This folder contains a default page for the domain in a file called index.html. Delete or move this file to another location.
5. Extract the Liferay dependencies archive into your domain's lib folder. Extract your database's JDBC driver here as well.
6. On Glassfish 3.0.1, you will need to extract commons-codec.jar from the Liferay WAR file, rename it to commons-codec-repackaged.jar, and copy it to [Glassfish Root]/glassfish/modules/, overwriting Glassfish's version of the file

Database Configuration

If you want GlassFish to manage the data source, use the following instructions. If you want to use the built-in Liferay data source, you can skip this section.

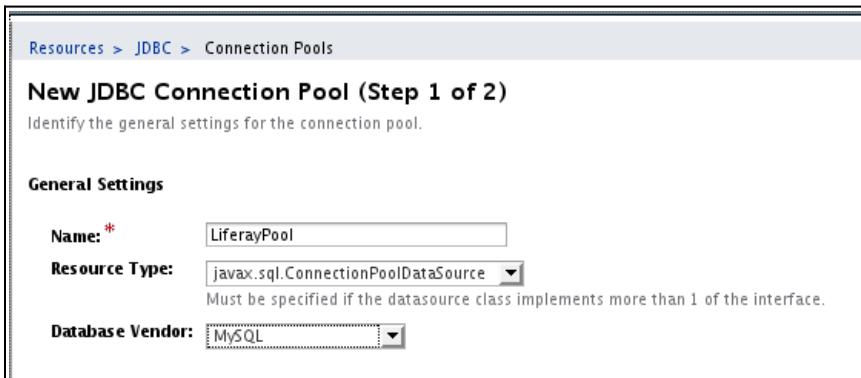


Illustration 3: Glassfish JDBC Connection Pool

1. Go to the GlassFish console URL: <http://localhost:4848>.
2. Under *Other Tasks*, select *Create New JDBC Connection Pool*.
3. In the first screen, give it a name of LiferayPool, a Resource Type of javax.sql.ConnectionPoolDataSource, and select MySQL as the Database Vendor. Click *Next*.
4. On the next page, scroll down to the *Additional Properties* section. Find the property called *URL*, and set its value to:

```
jdbc:mysql://localhost/lportal?
useUnicode=true&characterEncoding=UTF-
8&emulateLocators=true
```

5. If your database is not on the same server as GlassFish, substitute your database server's host name for localhost above.
6. Click *Add Property*, and add a property called *user* with a value of the user name to connect to the database.
7. Click *Add Property* again, and add a property called *password* with a value of the password to connect to the database.
8. Click *Finish*.
9. You will now see a list of Connection Pools. To test your connection, click the LiferayPool and click the *Ping* button. If you get a **Ping Succeeded** message, everything has been set up correctly.
10. Click *JDBC Resources*. You will see a list of JDBC Resources by JNDI Name.
11. Click *New*.
12. Make the JNDI Name *jdbc/LiferayPool* and select the LiferayPool you created earlier.

13. Click *OK*.

Mail Configuration

If you want GlassFish to manage your mail session, follow the instructions below. If you want Liferay to manage your mail session, you can skip this section.

1. Under *Resources*, click *JavaMail Sessions*.
2. Click *New*.
3. Give the JavaMail Session a JNDI name of `mail/MailSession`, and fill out the rest of the form with the appropriate information for your mail server.

Click *OK*.

Deploy Liferay

1. Create a file called `portal-ext.properties`. Add the following directives to the file:

```
jdbc.default.driverClassName=com.mysql.jdbc.Driver  
jdbc.default.url=jdbc:mysql://localhost/lportal?  
useUnicode=true&characterEncoding=UTF-8&useFastDateParsing=false  
jdbc.default.username=root  
jdbc.default.password=root
```

If you are using GlassFish's data source, add the JNDI name instead:

```
jdbc.default.jndi.name=jdbc/LiferayPool
```

Do the same thing for the Mail Session. If you are using the built-in configuration, set the following properties for your system:

```
mail.session.mail.pop3.host=localhost  
mail.session.mail.pop3.password=  
mail.session.mail.pop3.port=110  
mail.session.mail.pop3.user=  
mail.session.mail.smtp.auth=false  
mail.session.mail.smtp.host=localhost  
mail.session.mail.smtp.password=  
mail.session.mail.smtp.port=25  
mail.session.mail.smtp.user=  
mail.session.mail.store.protocol=pop3  
mail.session.mail.transport.protocol=smtp
```

If you are using GlassFish's mail session, add the JNDI name instead:

```
mail.session.jndi.name=mail/MailSession
```

Save and close the file.

2. Go to the GlassFish console URL: `http://localhost:4848`
3. Click *Applications* in the tree on the left.
4. Click the *Deploy* button.
5. Under *Packaged File to Be Uploaded to the Server* click *Choose File*, and browse to the location of the Liferay .war file.
6. Leave the rest of the defaults and click *OK*.

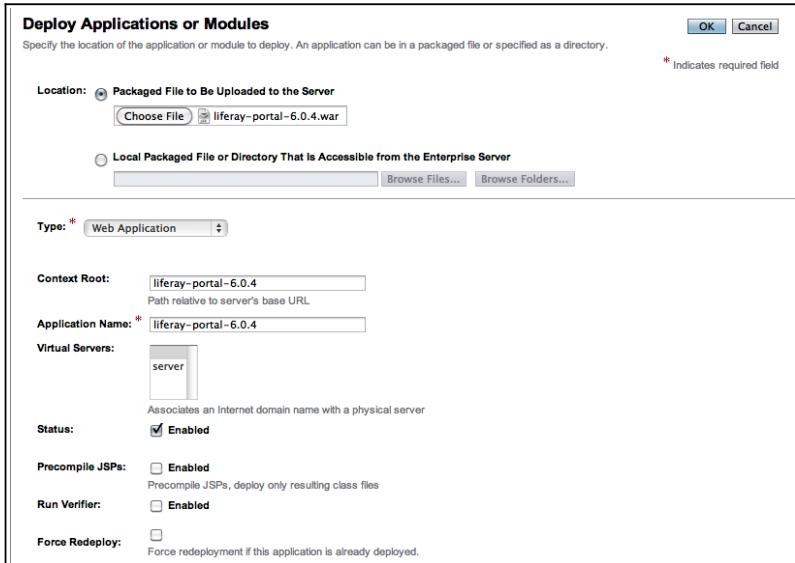


Illustration 4: Deploying Liferay in GlassFish 3

Liferay will be deployed and started automatically.

Jetty 6

Liferay Home is one folder above Jetty's install location.

1. Download and install Jetty 6.
2. Download the Liferay Portal .war file.
3. Download Liferay Portal Dependencies.
4. Copy the dependencies to `$JETTY_HOME/lib/ext`.
5. Edit `$JETTY_HOME/extra/etc/start-plus.config`.

```
$(jetty.home)/lib/ext/
```

Initial Setup

```
$(jetty.home)/lib/ext/*
```

6. Create a data source bound to `jdbc/LiferayPool` by editing `$JETTY_HOME/etc/jetty.xml`.

```
<Call name="addService">
  <Arg>
    <New class="org.mortbay.jetty.plus.JotmService">
      <Set name="Name">TransactionMgr</Set>
      <Call name="addDataSource">
        <Arg>jdbc/LiferayPool</Arg>
        <Arg>
          <New
            class="org.enhydra.jdbc.standard.StandardXADataSource">
            <Set name="DriverName">com.mysql.jdbc.Driver</Set>
            <Set name="Url">jdbc:mysql://localhost/lportal?
useUnicode=true&characterEncoding=UTF-8</Set>
            <Set name="User"></Set>
            <Set name="Password"></Set>
          </New>
        </Arg>
        <Arg>
          <New
            class="org.enhydra.jdbc.pool.StandardXAPoolDataSource">
            <Arg type="Integer">4</Arg>
            <Set name="MinSize">4</Set>
            <Set name="MaxSize">15</Set>
          </New>
        </Arg>
      </Call>
    </New>
  </Arg>
</Call>
```

7. Download `mysql-connector-java-{$version}-bin.jar` and copy to `$JETTY_HOME/lib/ext`. This is the JDBC driver for MySQL. If you are using a different database, copy the appropriate driver.
8. Create a mail session bound to `mail/MailSession` by editing `$JETTY_HOME/etc/jetty.xml`:

```
<Call name="addService">
  <Arg>
    <New class="org.mortbay.jetty.plus.MailService">
      <Set name="Name">MailService</Set>
      <Set name="JNDI">mail/MailSession</Set>
      <Put name="mail.smtp.host">localhost</Put>
    </New>
  </Arg>
</Call>
```

9. Create `$JETTY_HOME/etc/jaas.config`.

```
PortalRealm {
    com.liferay.portal.kernel.security.jaas.PortalLoginModule required;
};
```

10. Create directory `$JETTY_HOME/webapps/root` and unpack the Liferay .war file into it.
11. Go to `$JETTY_HOME/webapps/root/WEB-INF/lib` and delete `xerces-Impl.jar` and `xml-apis.jar`.
15. Copy `$JETTY_HOME/webapps/root/WEB-INF/lib/commons-logging.jar` to `$JETTY_HOME/ext` (overwriting the existing one).
16. Create batch file: `run.bat`.

```
@echo off

if "" == "%JAVA_HOME%" goto errorJavaHome

%JAVA_HOME%/bin/java -Xmx512m -Dfile.encoding=UTF8 -Duser.timezone=GMT
-Djava.security.auth.login.config=../etc/jaas.config
-DSTART=../extra/etc/start-plus.config -jar ../start.jar ../etc/jetty.xml

goto end

:errorJavaHome
echo JAVA_HOME not defined.

goto end

:end
```

Note: If you get a `java.lang.OutOfMemoryError` exception while starting up Jetty, give your JVM more memory by setting `-Xmx1024m`.

Start Liferay by running `run.bat`. Open your browser to `http://localhost:8080`. You should see the default Liferay home page.

JBoss 5.x

Liferay Home is one folder above JBoss's install location.

1. Download and install JBoss AS 5.0.1 GA into your preferred directory. This directory will be referred to below as `$JBoss_HOME`.
2. Download the latest version of the Liferay Portal .war file.
3. Remove `hibernate-validator.jar` from `$JBoss_HOME/common/lib`.
4. Go to `$JBoss_HOME/server/default/lib/`. Download `mysql-connector-java-{$version}-bin.jar` and copy to this directory. (This

is the JDBC connector for MySQL. Use the appropriate driver for your database.)

5. Download Liferay's Portal Dependencies. Unzip to \$JBOSS_HOME/server/default/lib.
6. Configure JAAS. Edit \$JBOSS_HOME/server/default/conf/login-config.xml and comment out the entire XML for policy *other* in lines 115-131.

```
<!--<application-policy name = "other">-->
...
<!--<authentication>
<login-module code = "org.jboss.security.
auth.spi.UsersRolesLoginModule"
flag = "required" />
</authentication>
</application-policy>-->
```

Database Configuration

If you want JBoss to manage the data source, use the following instructions. If you want to use the built-in Liferay data source, you can skip this section.

Create \$JBOSS_HOME/server/default/deploy/liferay-ds.xml with the following content:

```
<datasources>
    <local-tx-datasource>
        <jndi-name>jdbc/LiferayPool</jndi-name>
        <connection-url>
            jdbc:mysql://localhost/lportal?
useUnicode=true&characterEncoding=UTF-8
        </connection-url>
        <driver-class>com.mysql.jdbc.Driver</driver-class>
        <user-name></user-name>
        <password></password>
        <min-pool-size>0</min-pool-size>
    </local-tx-datasource>
</datasources>
```

Mail Configuration

If you want JBoss to manage the mail configuration, use the following instructions. If you want to use the built-in Liferay mail session, you can skip this section.

Set mail properties by replacing the contents of \$JBOSS_HOME/server/default/deploy/mail-service.xml with:

```
<?xml version="1.0"?>
<server>
<mbean code="org.jboss.mail.MailService" name="jboss:service=MailSession">
<attribute name="JNDIName">mail/MailSession</attribute>
<attribute name="User">nobody</attribute>
<attribute name="Password">password</attribute>
<attribute name="Configuration">
<configuration>
<property name="mail.store.protocol" value="imap" />
<property name="mail.transport.protocol" value="smtp" />
<property name="mail imap host" value="localhost" />
<property name="mail.pop3.host" value="localhost" />
<property name="mail.smtp.host" value="localhost" />
</configuration>
</attribute>
</mbean>
</server>
```

Deploy Liferay

1. Delete all the files and folders in \$JBoss_HOME/server/default/deploy/ROOT.war
2. Unzip the Liferay .war file to the ROOT.war directory.
3. Remove jaxrpc.jar, stax.jar, xercesImpl.jar, xml-apis.jar from:
\$JBoss_HOME/server/default/deploy/ROOT.war/WEB-INF/lib
4. Navigate to the Liferay Home folder, which is one folder above JBoss's install location.
5. Create a file called portal-ext.properties. Add the following directives to the file:

```
jdbc.default.driverClassName=com.mysql.jdbc.Driver
jdbc.default.url=jdbc:mysql://localhost/lportal?
useUnicode=true&characterEncoding=UTF-8&useFastDateParsing=false
jdbc.default.username=root
jdbc.default.password=root
```

If you are using JBoss's data source, add the JNDI name instead:

```
jdbc.default.jndi.name=jdbc/LiferayPool
```

Do the same thing for the Mail Session. If you are using the built-in configuration, set the following properties for your system:

```
mail.session.mail.pop3.host=localhost
mail.session.mail.pop3.password=
mail.session.mail.pop3.port=110
```

```
mail.session.mail.pop3.user=
mail.session.mail.smtp.auth=false
mail.session.mail.smtp.host=localhost
mail.session.mail.smtp.password=
mail.session.mail.smtp.port=25
mail.session.mail.smtp.user=
mail.session.mail.store.protocol=pop3
mail.session.mail.transport.protocol=smtp
```

If you are using JBoss's mail session, add the JNDI name instead:

```
mail.session.jndi.name=mail/MailSession
```

Save and close the file.

Start JBoss. Open your browser to <http://localhost:8080>. You should see the default Liferay home page.

Resin 3.1.x

Liferay Home is one folder above Resin's install location.

1. Download and install Resin into your preferred directory. From now on, the directory where you installed Resin will be referred to as \$RESIN_HOME.
2. Edit \$RESIN_HOME/conf/resin.conf. Replace lines 8-13 with:

```
<class-loader>
    <tree-loader path="${resin.home}/lib"/>
    <tree-loader path="${server.root}/lib"/>
    <compiling-loader path="$
{server.rootDir}/common/classes"/>
    <library-loader path="${server.rootDir}/common/lib"/>
</class-loader>
```

And add the following:

```
<database>
<jndi-name>jdbc/LiferayPool</jndi-name>
<driver type="com.mysql.jdbc.Driver">
<url>jdbc:mysql://localhost/lportal?
useUnicode=true&characterEncoding=UTF-8</url>
<user></user>
<password></password>
</driver>
<prepared-statement-cache-size>8</prepared-statement-cache-size>
<max-connections>20</max-connections>
<max-idle-time>30s</max-idle-time>
</database>
<resource jndi-name="mail/MailSession" type="javax.mail.Session">
<init>
```

```
<mail.store.protocol>imap</mail.store.protocol>
<mail.transport.protocol>smtp</mail.transport.protocol>
<mail imap.host>localhost</mail imap.host>
<mail.pop3.host>localhost</mail.pop3.host>
<mail.smtp.host>localhost</mail.smtp.host>
</init>
</resource>
<system-property
    javax.xml.parsers.DocumentBuilderFactory="org.apache.xerces.jaxp.DocumentBuilderFactoryImpl"
    />
<system-property
    javax.xml.parsers.SAXParserFactory="org.apache.xerces.jaxp.SAXParserFactoryImpl" />
<system-property
    javax.xml.transform.TransformerFactory="org.apache.xalan.processor.TransformerFactoryImpl" />
<system-property
    org.xml.sax.driver="org.apache.xerces.parsers.SAXParser" />
```

3. Go to \$RESIN_HOME and create a new directory called common/lib. Download mysql-connector-java-{\$version}-bin.jar and copy to this directory. This is the JDBC connector for MySQL. If you are using another database, substitute this with the appropriate driver.
4. Download the Liferay Portal Dependencies and unzip into \$RESIN_HOME/common/lib.
5. Delete contents of \$RESIN_HOME/webapps/ROOT.
6. Unzip liferay-portal-x.x.x.war to \$RESIN_HOME/webapps/ROOT.
7. If you are using Resin 3.1.9 or higher, remove \$RESIN_HOME/lib/portlet-01.jar. This contains the old Portlet 1.0 classes. The Portlet 2.0 classes are backwards compatible, so this should not affect anything.
8. Next, you will need several .jar files which are included as part of the Liferay source distribution. Many application servers ship with these already on the class path, but Resin does not. The best way to get the appropriate versions of these files is to download the Liferay source code and get them from there. Once you have downloaded the Liferay source, unzip it to a temporary folder.
 1. Go to \$LIFERAY_SOURCE/lib/development/ and copy activation.jar and mail.jar to \$RESIN_HOME/common/lib. Copy sax-path.jar and xalan.jar to \$RESIN_HOME/lib.
 2. Go to \$LIFERAY_SOURCE/lib/portal and copy xercesImpl.jar and xml-apis.jar to \$RESIN_HOME/lib.

9. To start the server, open a command prompt, navigate to the \$RESIN_HOME and type:

```
java -jar lib/resin.jar start
```

10. Open your browser to <http://localhost:8080>. You should see the default Liferay home page.

Resin 3.2.x

Liferay Home is one folder up from Resin's install location.

1. Download and install Resin 3.2.1 into your preferred directory. From now on, the directory where you installed Resin will be referred to as \$RESIN_HOME.
2. Edit \$RESIN_HOME/conf/resin.conf. Replace lines line 9-13 with:

```
<tree-loader path="${resin.home}/ext-lib"/>
<tree-loader path="${resin.root}/ext-lib"/>

<tree-loader path="${resin.home}/lib"/>
<tree-loader path="${resin.root}/lib"/>

<compiling-loader path="${server.rootDir}/common/classes"/>
<library-loader path="${server.rootDir}/common/lib"/>
```

3. Search <jvm-arg> tag in resin.conf and replace what is there with the following:

```
<jvm-arg>-Xmx256m</jvm-arg>
<jvm-arg>-Xss1m</jvm-arg>
<jvm-arg>-Dcom.sun.management.jmxremote</jvm-arg>

<jvm-arg>-Xmx1024m</jvm-arg>
<jvm-arg>-XX:MaxPermSize=256m</jvm-arg>
<jvm-arg>-Dfile.encoding=UTF-8</jvm-arg>
<jvm-arg>-Duser.timezone=GMT</jvm-arg>
```

4. Go to \$RESIN_HOME and create a new directory called common/lib. Download mysqlconnector-java-{\$version}-bin.jar and copy to this directory. This is the JDBC connector for MySQL. If you are using another database, substitute this with the appropriate driver.
5. Download the Liferay Portal Dependencies and unzip into \$RESIN_HOME/common/lib.
6. Delete the contents of \$RESIN_HOME/webapps/ROOT.
7. Unzip liferay-portal-x.x.x.war to \$RESIN_HOME/webapps/ROOT.

8. Next, you will need several jar files which are included as part of the Liferay source distribution. Many application servers ship with these already on the class path, but Resin does not. The best way to get the appropriate versions of these files is to download the Liferay source code and get them from there. Once you have downloaded the Liferay source, unzip it to a temporary folder.
 - o Go to \$LIFERAY_SOURCE/lib/development Copy saxpath.jar to \$RESIN_HOME/common/lib
9. To start the server, open a command prompt, navigate to the \$RESIN_HOME and type:

```
java -jar lib/resin.jar start
```

Open your browser to <http://localhost:8080>. You should see the default Liferay home page.

Tomcat 6.0.x

Liferay Home is one folder above Tomcat's install location.

1. Download and install Tomcat 6.0.X into your preferred directory. From now on, the directory where you installed Tomcat will be referred to as \$TOMCAT_HOME.
Note: For JDK 5 users: move \$TOMCAT_HOME/webapps/ROOT/WEB-INF/lib/xercesImpl.jar to \$TOMCAT_HOME/common/endorsed. JDK 1.4 is no longer supported in Liferay 5.x and above.
2. Create and edit \$TOMCAT_HOME/conf/Catalina/localhost/ROOT.xml to set up the portal web application.

```
<Context path="">
</Context>
```

3. Download liferay-portal-x.x.x.war.
4. Download Liferay's Portal Dependencies. Create a \$TOMCAT_HOME/lib/ext directory and unzip the dependencies ZIP in there. If the files do not extract to this directory, make sure they are in the correct directory by moving them there.
5. Edit \$TOMCAT_HOME/conf/catalina.properties:

```
common.loader=
    ${catalina.home}/classes,\n    ...\\
    ${catalina.home}/lib/ext/*.jar
```

6. Make sure your database server is installed and is working. If it's installed in a different machine, make sure that it's accessible from the one where Liferay is being installed.
7. Configure data sources for your database. Make sure the JDBC driver for your database is accessible by Tomcat. Obtain the JDBC driver for your version of the database server. In the case of MySQL use mysql-connector-java-{\$version}-bin.jar. Next, copy the JAR file to \$TOMCAT_HOME/common/lib/ext.
8. Edit \$TOMCAT_HOME/conf/Catalina/localhost/ROOT.xml.

```
<Context...>
    <Resource
        name="jdbc/LiferayPool"
        auth="Container"
        type="javax.sql.DataSource"
        driverClassName="com.mysql.jdbc.Driver"
        url="jdbc:mysql://localhost/lportal?
useUnicode=true&characterEncoding=UTF-8"
        username=""
        password=""
        maxActive="100"
        maxIdle="30"
        maxWait="10000"
    />
</Context>
```

9. Be sure to enter the user name and password to your database in the appropriate fields above.
10. Create a mail session bound to mail/MailSession. Edit \$TOMCAT_HOME/conf/Catalina/localhost/ROOT.xml and configure a mail session.

```
<Context...>
    <Resource
        name="mail/MailSession"
        auth="Container"
        type="javax.mail.Session"
        mail.transport.protocol="smtp"
        mail.smtp.host="localhost"
        mail.store.protocol="imap"
        mail.imap.host="localhost"
    />
</Context>
```

11. Configure JAAS.

Edit \$TOMCAT_HOME/conf/Catalina/localhost/ROOT.xml and configure a security realm.

```
<Context...>
<Realm
  className="org.apache.catalina.realm.JAASRealm"
  appName="PortalRealm"
  userClassNames="com.liferay.portal.security.jaas.PortalPrincipal"
  roleClassNames="com.liferay.portal.security.jaas.PortalRole"
  debug="99"
  useContextClassLoader="false"
 />
</Context>
```

14. Create \$TOMCAT_HOME/conf/jaas.config.

```
PortalRealm {
  com.liferay.portal.kernel.security.jaas.PortalLoginModule required;
};
```

15. Edit \$TOMCAT_HOME/bin/catalina.bat (on Windows) or \$TOMCAT_HOME/bin/catalina.sh (on Linux / Mac / Unix) so that Tomcat can reference the login module.

```
rem ----- Execute...
set JAVA_OPTS=-Xms128m -Xmx512m -Dfile.encoding=UTF8
-Duser.timezone=GMT -Djava.security.auth.login.config=
%CATALINA_HOME%/conf/jaas.config
```

16. Delete contents \$TOMCAT_HOME/webapps/ROOT directory. This undeploys the default Tomcat home page.

17. Unpack liferay-portal-x.x.x.war to \$TOMCAT_HOME/webapps/ROOT.

18. For supporting UTF-8 URI Encoding, edit \$TOMCAT_HOME/conf/server.xml:

```
<!-- Define a non-SSL HTTP/1.1 Connector on port 8080 -->
<Connector port="8080" maxHttpHeaderSize="8192"
           maxThreads="150" minSpareThreads="25"
           maxSpareThreads="75" enableLookups="false"
           redirectPort="8443" acceptCount="100"
           connectionTimeout="20000"
           disableUploadTimeout="true"
           URIEncoding="UTF-8"
 />
```

19. Run Tomcat, point browser to http://localhost:8080. You should see the default Liferay home page.

WebLogic 10

Liferay Home is one folder above the home folder of the domain in which Liferay is installed.

These instructions assume that you have already configured a domain and server, and that you have access to the WebLogic console.

Dependency Jars

1. Navigate to the folder which corresponds to the domain to which you will be installing Liferay. Inside this folder is a lib folder. Unzip the Liferay dependencies archive to this folder.
2. Copy the JDBC driver for your database to this folder as well.
3. You will also need the xercesImpl.jar or you will get SAX parsing errors after you deploy Liferay. You may download this from <http://xerces.apache.org>. Copy the xercesImpl.jar file into this directory.
4. Create a folder called endorsed in \$WEB-LOGIC-HOME/jrockit90_150_04/jre/lib, then copy commons-lang.jar, rhino.jar, serializer.jar, and xalan.jar to the folder that you just created.

Database Configuration

If you want WebLogic to manage your data source, use the following procedure. If you want to use Liferay's built-in data source, you can skip this section.

1. Browse to your WebLogic Console. Click the Lock & Edit button above the Domain Structure tree on the left side of the page.
2. From the Domain Structure tree on the left, select Data Sources. Then click the New button on the right side of the screen.
3. Give the Data Source a name, such as LiferayDataSource.
4. Define the JNDI name as jdbc/LiferayPool.
5. Select your Database Type and the Driver class, and then click the Next button.
6. Accept the defaults on the next screen by clicking Next.
7. On the next screen, put in your Database Name, Host Name, Database User Name, and Password. If you have been following the defaults we



Illustration 5: WebLogic: Data Sources

have been using so far, you would use *lportal*, *localhost*, *root*, and no password as the values. Click *Next*.

8. The next screen allows you to test your database configuration. Click the *Test Connection* button. If the test succeeds, you have configured your database correctly. Check off the server you want to deploy this Data Source to (AdminServer is the default). Click *Finish*.
9. Click the *Activate Changes* button on the left, above the Domain Structure tree.

Mail Configuration

If you want WebLogic to manage your mail sessions, use the following procedure. If you want to use Liferay's built-in mail sessions, you can skip this section.

1. In the Domain Structure tree, select *Mail Sessions*. Then click the *Lock & Edit* button again to enable modifying these settings.
2. Click the *New* button which is now enabled on the right side of the screen.
3. Give the Mail Session a name, such as *LiferayMail*.
4. Select your new LiferayMail session from the list by clicking on it.
5. On the screen that appears, define the JNDI name as *mail/MailSession*. Click the *Save* button.
6. Click the *Targets* tab. Check off the server you want to deploy this Data Source to (AdminServer is the default).
7. Click the *Activate Changes* button on the left side of the screen, above the Domain Structure tree.

Deploy Liferay

1. Click the *Deployments* option in the Domain Structure tree on the left side of the screen.
2. Click the *Lock & Edit* button above the Domain Structure tree.
3. Click the *Install* button on the right side of the screen.
4. Click the *Upload your file(s)* link.

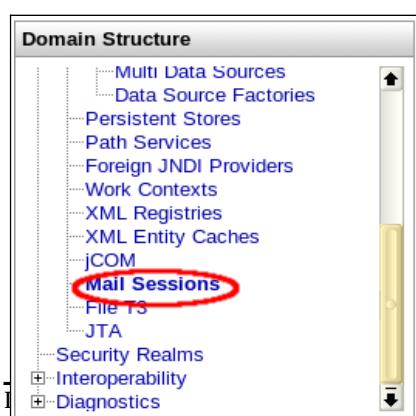


Illustration 6: WebLogic: Mail Sessions

5. Browse to where you have stored the Liferay .war file, select it, and then click *Next*.
6. Select the Liferay .war file from the list and click *Next*.
7. Leave *Install this deployment as an application* selected and click *Next*.
8. Give the application a name (the default name is fine). Leave the other defaults selected and then click *Finish*.
9. WebLogic will now deploy Liferay. When it is finished, a summary screen is displayed. Click the *Activate Changes* link on the left above the Domain Structure tree.
10. Create a portal-ext.properties file in the Liferay Home folder, which is one folder up from your domain's home folder. If you are using Liferay's built-in data source, add the database settings:

```
jdbc.default.driverClassName=com.mysql.jdbc.Driver  
jdbc.default.url=jdbc:mysql://localhost/lportal?  
useUnicode=true&characterEncoding=UTF-8&useFastDateParsing=false  
jdbc.default.username=root  
jdbc.default.password=root
```

If you are using WebLogic's data source, add the JNDI name instead:

```
jdbc.default.jndi.name=jdbc/LiferayPool
```

Do the same thing for the Mail Session. If you are using the built-in configuration, set the following properties for your system:

```
mail.session.mail.pop3.host=localhost  
mail.session.mail.pop3.password=  
mail.session.mail.pop3.port=110  
mail.session.mail.pop3.user=  
mail.session.mail.smtp.auth=false  
mail.session.mail.smtp.host=localhost  
mail.session.mail.smtp.password=  
mail.session.mail.smtp.port=25  
mail.session.mail.smtp.user=  
mail.session.mail.store.protocol=pop3  
mail.session.mail.transport.protocol=smtp
```

If you are using WebLogic's mail session, add the JNDI name instead:

```
mail.session.jndi.name=mail/MailSession
```

11. In the Deployments screen, select the Liferay application and click the *Start* button. Select *Servicing All Requests* in the pop up.
12. Click *Yes* to continue on the next screen.

Liferay will start. You will be able to get to it by browsing to `http://<server name>:7001`. If your browser is running on the same machine upon which you have installed Liferay, the URL is <http://localhost:7001>.

Oracle WebLogic 10.3

If you still have the mainWebApp module installed, you will need to remove it first.

Dependency Jars

1. Navigate to the folder which corresponds to the domain to which you will be installing Liferay. Inside this folder is a `lib` folder. Unzip the Liferay dependencies archive to this folder.
2. Copy the JDBC driver for your database to this folder as well.
3. Create a folder called `endorsed` in `$WEBLOGIC-HOME/jrockit90_150_04/jre/lib`, then copy `commons-lang.jar`, `rhino.jar`, `serializer.jar`, and `xalan.jar` to the folder that you just created.

Start Application Server

Start WebLogic.

Database Configuration

If you want WebLogic to manage your data source, use the following procedure. If you want to use Liferay's built-in data source, you can skip this section.

The screenshot shows the 'Create a New JDBC Data Source' dialog box. At the top, there are 'Back', 'Next', 'Finish', and 'Cancel' buttons. Below them is a section titled 'JDBC Data Source Properties' with the sub-instruction: 'The following properties will be used to identify your new JDBC data source.' It includes a note: '* Indicates required fields'. The 'Name' field is set to 'Liferay Data Source'. The 'JNDI Name' field contains 'jdbc/LiferayPool'. The 'Database Type' dropdown is set to 'MySQL'. The 'Database Driver' dropdown is set to 'MySQL's Driver (Type 4) Versions:using com.mysql.jdbc.Driver'. At the bottom of the dialog are the same 'Back', 'Next', 'Finish', and 'Cancel' buttons.

Illustration 7: Creating a data source in WebLogic 10.3

1. Select *JDBC → Data Sources*. Click *New*.
2. Give your data source a name, such as *Liferay Data Source*. The JNDI name should be *jdbc/LiferayPool*.
3. Choose the type of database. From the screen shot, you can see that we have chosen MySQL. The database driver class should be chosen for you automatically.
4. Click *Next* twice. You should be on the *Connection Properties* screen. Enter the database name, the host name, the port, the database user name, and the password. WebLogic will use this information to construct the appropriate JDBC URL to connect to your database. Click *Next*.

5. WebLogic will now confirm with you the information that it gathered. For MySQL, some additional parameters need to be added to the URL. Modify the JDBC URL so that it has the proper parameters:

```
jdbc:mysql://localhost/lportal?  
useUnicode=true&characterEncoding=UTF-8&\  
useFastDateParsing=false
```

6. Click *Test Configuration* to make sure WebLogic can connect to your database successfully. If it does, click *Finish*.
7. You will be back to the list of data sources. Notice that your new data source has no value in the *Target* column. Click on your data source to edit it.
8. Click the *Targets* tab and check off the server instance(s) to which you wish to deploy your data source. Then click *Save*.

Mail Configuration

1. Select *Mail Sessions* and create a new mail session which points to your mail server.
2. Give it the name Liferay Mail and give it the JNDI name of *mail/MailSession* and click *Next*.
3. Choose your server and then click *Finish*.

Deploy Liferay

1. Create a *portal-ext.properties* file in the Liferay Home folder, which is one folder up from your domain's home folder. If you are using Liferay's built-in data source, add the database settings:

```
jdbc.default.driverClassName=com.mysql.jdbc.Driver  
jdbc.default.url=jdbc:mysql://localhost/lportal?  
useUnicode=true&characterEncoding=UTF-8&useFastDateParsing=false  
jdbc.default.username=root  
jdbc.default.password=root
```

If you are using WebLogic's data source, add the JNDI name instead:

```
jdbc.default.jndi.name=jdbc/LiferayPool
```

Do the same thing for the Mail Session. If you are using the built-in configuration, set the following properties for your system:

```
mail.session.mail.pop3.host=localhost  
mail.session.mail.pop3.password=  
mail.session.mail.pop3.port=110  
mail.session.mail.pop3.user=  
mail.session.mail.smtp.auth=false
```

```
mail.session.mail.smtp.host=localhost  
mail.session.mail.smtp.password=  
mail.session.mail.smtp.port=25  
mail.session.mail.smtp.user=  
mail.session.mail.store.protocol=pop3  
mail.session.mail.transport.protocol=smtp
```

If you are using WebLogic's mail session, add the JNDI name instead:

```
mail.session.jndi.name=mail/MailSession
```

Save and close the file.

2. Add the following to server.xml

```
<library-ref>  
  <library-name>jsf</library-name>  
  <specification-version>1.2</specification-version>  
  <implementation-version>1.2</implementation-version>  
  <exact-match>false</exact-match>  
</library-ref>
```

3. Select *Deployments* and click the *Install* button. Upload *jsf-1.2.war* from WebLogic's common files directory, and select *Install this deployment as a library*.
4. After installing the JSF libraries, go back to deployments and select the Liferay .war file from the file system or click the *Upload Your File(s)* link to upload it, and then click *Next*.
5. Select *Install this deployment as an application* and click *Next*.
6. If the default name is appropriate for your installation, keep it. Otherwise, give it a name of your choosing and click *Next*.
7. Click *Finish*. After the deployment finishes, click *Save*.



Tip: After Liferay completes installing, you may see an error initializing the Web Proxy portlet. Because the XSL parser configured by default within WebLogic cannot compile a style sheet in this portlet, Liferay disables it by default. To re-enable this portlet, extract *xalan.jar* and *serializer.jar* from the Liferay .war archive and copy them to your JDK's endorsed folder for libraries. If you are using JRockit, you may find this folder in

```
[Bea Home]/jrockit_160_05/jre/lib/ext.
```

WebSphere 6.1



Tip: Throughout this installation and configuration process, WebSphere will prompt you to Click Save to apply changes to Master Configuration. Do so intermittently to save your changes.

Liferay Home is in a folder called `liferay` in the home folder of the user ID that is running WebSphere.

Installation

1. Download the Liferay Portal WAR file.
2. Download and extract these Liferay jars to `websphere/appserver/lib/ext`.
 - Dependency libraries (Liferay Portal Dependencies)
 - Your database JDBC driver .jar
 - Currently you also need to copy `portlet.jar` from the Liferay Dependencies archive into `WebSphere/AppServer/java/jre/lib/ext`, as WebSphere already contains older versions of the portlet .jar which must be overridden at the highest level of the class path. This issue may be fixed in future releases; check the Liferay Wiki for updates to this issue.

Database Configuration

1. Start WebSphere.
2. Open Administrative Console and log in.
3. Click *Resources*, click *JDBC Providers*.
4. Click *New*.
5. For name, enter the name of the JDBC provider (e.g. *MySQL JDBC Provider*).
6. For Implementation class name, enter the implementation class for your database driver's connection pool data source For MySQL, enter:

com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource
7. Click *Next*.
8. Clear any text in class path. You already copied the necessary .jars to a location on the server's class path.

Initial Setup

9. Click *Next*.
10. Click *Finish*.
11. Click *Data Sources* under *Additional Properties*.
12. Click *New*.
13. Enter a name: *liferaydatabasesource*.
14. Enter JNDI: *jdbc/LiferayPool*.
15. Everything else should stay at the default values.
16. Click *Next*.
17. Under *Additional Properties*, click *Custom Properties*.
18. Click *New*.
19. Create three custom properties by entering *Name*, *Value* and clicking *OK* for each row in the following table.

Name	Value
1. user	root
2. serverName	localhost
3. databaseName	lportal

20. Click *Data Sources* -> *Test Connection* to test.

Mail Configuration

1. Click *Resources* -> *Mail Providers*.
2. Click *Built-in Mail Provider*.
3. Click *Mail Sessions*.
4. Click *New*.

Name: *liferaymail*

JNDI Name: *mail/MailSession*

5. Click *OK*.
6. Click *Security*.
7. Click *Secure administration, applications, and infrastructure*.
8. Select *Enable application security*.

9. Deselect *Use Java 2 security to restrict application access to local resources.*

Install Liferay

1. Click *Applications -> Install new applications.*
2. Browse for *liferay-portal-x.x.x.war*.

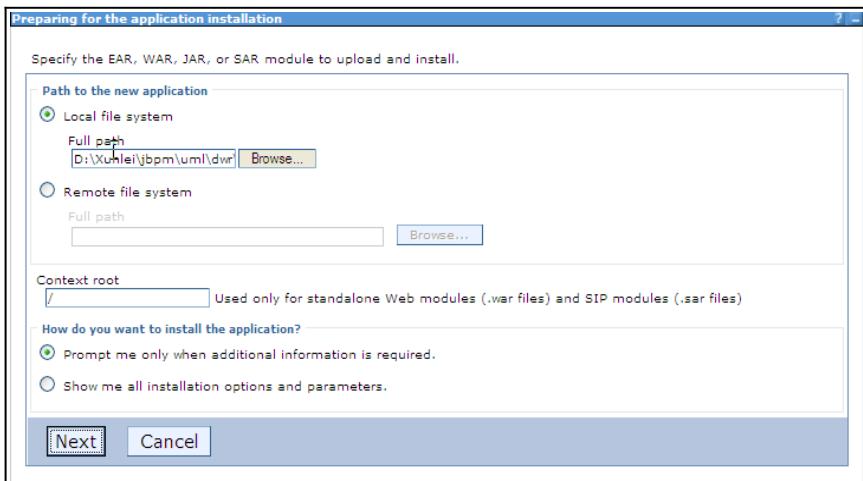


Illustration 8: Installing the Liferay .war file on WebSphere 6.1

3. Enter context root */*.
4. Click *Next*. For Steps 1 to 3, click *Next* to apply defaults.
5. Choose the *Mail Session* and *Data Source*, and then click *Next*.
6. Specify the virtual host upon which you want Liferay to run.
7. At the Summary Screen, click *Finish*.
8. Wait for the installation process to complete.
9. Save this configuration to master configuration by clicking on *System administration* and *Save Changes to Master Repository*.
10. Create a *portal-ext.properties* file in the Liferay Home folder. For WebSphere, this is a folder called *liferay* in the home folder of the user that is running WebSphere. If you are using Liferay's built-in data source, add the database settings:

```
jdbc.default.driverClassName=com.mysql.jdbc.Driver
jdbc.default.url=jdbc:mysql://localhost/lportal?
useUnicode=true&characterEncoding=UTF-8&useFastDateParsing=false
jdbc.default.username=root
jdbc.default.password=root
```

Initial Setup

If you are using WebSphere's data source per the instructions above, add the JNDI name instead:

```
jdbc.default.jndi.name=jdbc/LiferayPool
```

Do the same thing for the Mail Session. If you are using the built-in configuration, set the following properties for your system:

```
mail.session.mail.pop3.host=localhost  
mail.session.mail.pop3.password=  
mail.session.mail.pop3.port=110  
mail.session.mail.pop3.user=  
mail.session.mail.smtp.auth=false  
mail.session.mail.smtp.host=localhost  
mail.session.mail.smtp.password=  
mail.session.mail.smtp.port=25  
mail.session.mail.smtp.user=  
mail.session.mail.store.protocol=pop3  
mail.session.mail.transport.protocol=smtp
```

If you are using WebSphere's mail session, add the JNDI name instead:

```
mail.session.jndi.name=mail/MailSession
```

Save and close the file.

Start Liferay Portal

1. Applications.
2. Click *Enterprise Applications*.

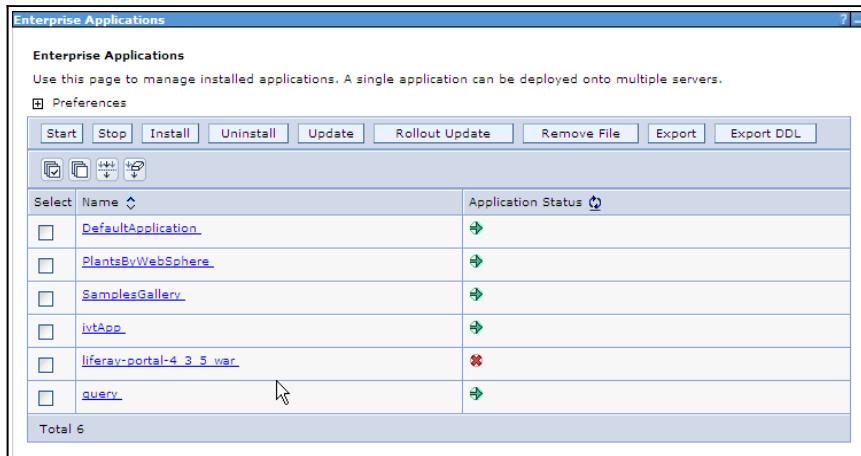


Illustration 9: Starting Liferay on WebSphere 6.1

3. Uninstall *DefaultApplication*, *PlantsByWebSphere* and *SamplesGallery*.
4. Select *liferay-portal.war*, click *Start*.
5. Open up the browser and go to <http://localhost:9080>. The default Liferay home page will be displayed.

WebSphere 7.0

Liferay Home is in a folder called `liferay` in the home folder of the user ID that is running WebSphere.

1. Download the Liferay Portal WAR file.
2. Download and extract these Liferay jars to `web-sphere/appserver/lib/ext`.
 - Dependency libraries (Liferay Portal Dependencies)
 - JDBC Driver for your database

Database Configuration

If you want WebSphere to manage the database connections, follow the instructions below.

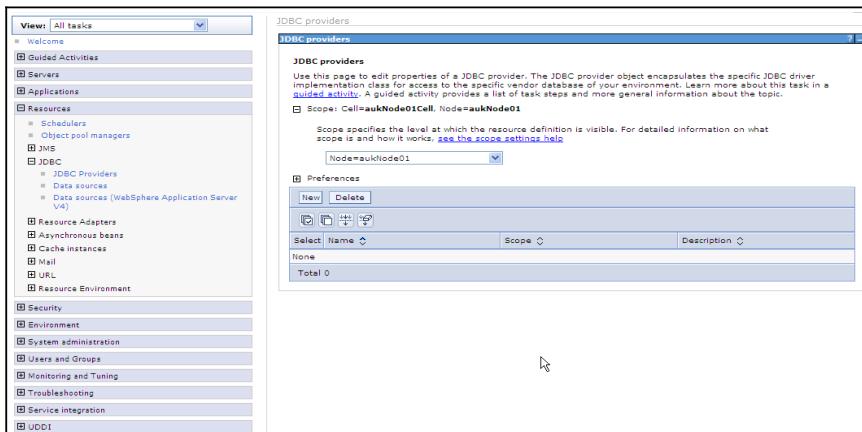


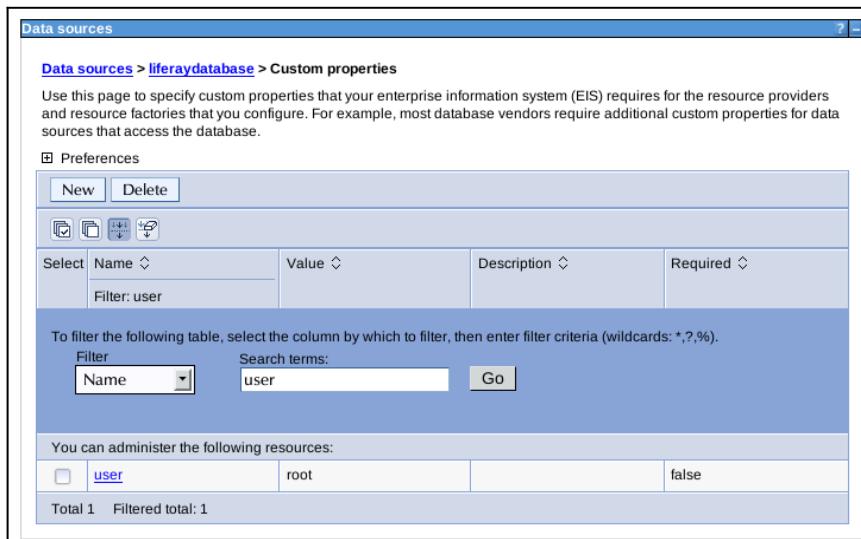
Illustration 10: WebSphere 7.0 JDBC Providers

1. Start WebSphere.
2. Open the Administrative Console and log in.
3. Click *Resources* → *JDBC Providers*.
4. Click *New*.

Initial Setup

5. For name, enter the name of JDBC provider (e.g. MySQL JDBC Provider).
6. For Implementation class name, enter:

com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource
7. Click *Next*.
8. Clear any text in class path. You already copied the necessary .jars to a location on the server's class path.
9. Click *Next*.
10. Click *Finish*.
11. Click *Data Sources* under *Additional Properties*.
12. Click *New*.
13. Enter a name: *liferaydatabasesource*.
14. Enter JNDI: *jdbc/LiferayPool*.
15. Everything else should stay at the default values. Save the data source.
16. When finished, go back into the data source and click *Custom Properties*, and then click the *Show Filter Function* button. This is the second from last of the small icons under the *New* and *Delete* buttons.
17. Type *user* into the search terms and click *Go*.



The screenshot shows the 'Data sources' interface in WebSphere. The title bar says 'Data sources'. Below it, the path 'Data sources > liferaydatabase > Custom properties' is displayed. The main content area has a heading 'Custom properties' and a sub-heading 'Use this page to specify custom properties that your enterprise information system (EIS) requires for the resource providers and resource factories that you configure. For example, most database vendors require additional custom properties for data sources that access the database.' There is a 'Preferences' checkbox followed by 'New' and 'Delete' buttons. Below these are several small icons. A table with columns 'Select', 'Name', 'Value', 'Description', and 'Required' is shown, with a 'Filter: user' input field above it. At the bottom of the table, there is a note: 'To filter the following table, select the column by which to filter, then enter filter criteria (wildcards: *, ?, %).'. Below this is a 'Filter' dropdown set to 'Name' and a 'Search terms:' input field containing 'user', with a 'Go' button next to it. At the very bottom, there is a section titled 'You can administer the following resources:' with a table showing one row: 'user' (checkbox checked), 'root' (checkbox unchecked), and 'false' (checkbox unchecked). The total count is 'Total 1 Filtered total: 1'.

Illustration 11: Modifying data source properties in WebSphere 7

18. Select the user property and give it the value of the user name to your database. Click *OK* and save to master configuration.
19. Do another filter search for the url property. Give it a value that points to your database. For example, the MySQL URL would be: `jdbc:mysql://localhost/lportal?useUnicode=true&characterEncoding=UTF-8&useFastDateParsing=false`. Click *OK* and save to master configuration.
20. Do another filter search for the password property. Enter the password for the user ID you added earlier as the value for this property. Click *OK* and save to master configuration.
21. Go back to the data source page by clicking it in the breadcrumb trail. Click the *Test Connection* button. It should connect successfully.

Mail Configuration

1. Click *Resources -> Mail -> Mail Providers*.
2. Click the Built-In Mail Provider for your node and server.
3. Click *Mail Sessions*, and then click the *New* button.
4. Give it a name of *liferaymail* and a JNDI name of *mail/MailSession*. Click *OK* and save to master configuration.
5. Click *Security -> Global Security* and deselect *Use Java 2 security to restrict application access to local resources* if it is selected. Click *Apply*.

Install Liferay

1. Click *Applications -> New Application -> New Enterprise Application*.
2. Browse to the Liferay .war file and click *Next*.
3. Leave *Fast Path* selected and click *Next*, and then click *Next* again.
4. Make sure your server is selected and click *Next*.
5. Keep the context root set to / and click *Next*.
6. Click *Finish*. When Liferay has installed, click *Save to Master Configuration*.

Start Liferay

1. Create a *portal-ext.properties* file in the Liferay Home folder. For WebSphere, this is a folder called *liferay* in the home folder of the user that is running WebSphere. If you are using Liferay's built-in data source, add the database settings:

Initial Setup

```
jdbc.default.driverClassName=com.mysql.jdbc.Driver  
jdbc.default.url=jdbc:mysql://localhost/lportal?  
useUnicode=true&characterEncoding=UTF-8&useFastDateParsing=false  
jdbc.default.username=root  
jdbc.default.password=root
```

2. If you are using WebSphere's data source per the instructions above, add the JNDI name instead:

```
jdbc.default.jndi.name=jdbc/LiferayPool
```

Do the same thing for the Mail Session. If you are using the built-in configuration, set the following properties for your system:

```
mail.session.mail.pop3.host=localhost  
mail.session.mail.pop3.password=  
mail.session.mail.pop3.port=110  
mail.session.mail.pop3.user=  
mail.session.mail.smtp.auth=false  
mail.session.mail.smtp.host=localhost  
mail.session.mail.smtp.password=  
mail.session.mail.smtp.port=25  
mail.session.mail.smtp.user=  
mail.session.mail.store.protocol=pop3  
mail.session.mail.transport.protocol=smtp
```

If you are using WebSphere's mail session, add the JNDI name instead:

```
mail.session.jndi.name=mail/MailSession
```

3. Save and close the file.
4. Click *Application Types -> WebSphere Enterprise Application*.
5. Uninstall the default application.
6. Select the Liferay application and click *Start*.

Making Liferay Coexist with Other Java EE Applications

Liferay Portal by default is configured to sit at the root (i.e., `/`) of your application server. Dedicating your application server to running only Liferay Portal is a good practice, allowing for separation between your portal environment and your web application environment. This is generally a best practice for portals, which by definition are application development platforms in and of themselves. For that reason, your instance of Liferay is likely to be hosting many applications, and even integrating several of them together on a single page. For this reason, you are going to want to make sure your portal environment has all the resources it needs to do this, and configuring it so that it is the sole consumer of any other .war files that get de-

ployed to the application server helps to make sure that your system performs optimally.

If, however, you want Liferay to share space on an application server with other applications, there is no reason why you cannot do that. In this instance, you may not want to make Liferay the default application in the root context of the server.

There are two steps to modifying this behavior:

1. Deploy Liferay in a context other than root (for example /portal).
2. Modify the portal-ext.properties file to tell Liferay the context to which it has been deployed.

To change the file, open it in a text editor. Place the portal.ctx property at the top of the file:

```
portal.ctx=/
```

This default setting defines Liferay Portal as the application that sits at the root context. If you change it to something else, say /portal, for example, you can then deploy Liferay in that context and it will live there instead of at the root context.

A full discussion of the portal-ext.properties file appears in Chapter 6.

Note for WebLogic Users: WebLogic also requires that you modify the weblogic.xml file which is included with Liferay. In this file are tags for the context root:

```
<context-root>/</context-root>
```

Change this so that it matches the path you set in your portal-ext.-properties file. You will have to modify the weblogic.xml file inside the Liferay .war itself. Extract the file from the .war file, modify it, and then put it back in the .war file. Then deploy the modified Liferay .war file to the server in the proper context.

Summary

This chapter is a guide to everything about installing Liferay. Whether you choose a Liferay bundle or an existing application server, Liferay Portal integrates seamlessly with your enterprise Java environment. It is supported on more application servers than any other portal platform, allowing you to preserve your investment in your application server of choice, or giving you the freedom to move to a different application server platform. Liferay is committed to providing you this freedom: we have 100 test servers certifying our builds with roughly 10,000 tests per version of Liferay Portal. Each of those tests must be run on all of our different supported combinations of ap-

plication servers, databases, and operating systems. So you can be sure that we are committed to supporting you on your environment of choice. Liferay Portal won't get in your way, and you can feel safe knowing that you have the freedom to use the software platform that is best for your organization, and Liferay Portal will run and perform well on it.

3. CONFIGURATION

Once Liferay is successfully installed, you can begin configuring it to fit it to your environment and your particular portal project. You can perform many of these configuration tasks through Liferay's portlet-driven user interface.

You will want to customize your portal by configuring various settings for it, such as email notifications, integration with services such as LDAP, creating users, user groups, organizations, communities, and roles, and readying your portal to have its content and applications loaded by your developers. This chapter covers these activities:

- *Liferay's User Interface*: How to navigate around Liferay and make use of the Control Panel.
- *Liferay Administration*: How to administer a Liferay portal.
- *Global Portal Settings*: Password policies, Settings, Monitoring, and more.

Liferay's User Interface

Liferay is a *portal server*. This means that it is designed to be a single environment where all of the applications a user needs can run, and these are integrated together in a consistent and systematic way. If an application lives outside of the portal, the portal should be able to consume some resource of the application (such as an RSS feed or a subset of functionality in a “dashboard” application) so that the end user can see everything he or she interacts with at a glance.

To achieve this, all of the application functionality within Liferay Portal is in fragments of the page called *portlets*. Portlets are web applications that run in a portion of a web page. Liferay's core is a portlet container, and the container's job is to aggregate the set of portlets that are to appear on any particular page and display them properly to the user. In this way, one or many applications can reside on a page, and the user can (at the administrator's discretion) arrange them in the way that works best for the user.

Portlet applications, like servlet applications, have become a Java standard which various portal server vendors have implemented. The Java standard defines the portlet specification. A JSR-168 or JSR-286 standard portlet should be deployable on any portlet container which supports those standards. Portlets are placed on the page in a certain order by the end user and are served up dynamically by the portal server.

Portal applications come generally in two flavors: 1) multiple portlets can be written to provide small amounts of functionality and then are aggregated by the portal server into a larger application, or 2) whole applications can be written to reside in only one or a few portlet windows. The choice is up to those designing the application. Developers only have to worry about what happens inside of the portlet itself; the portal server handles building out the page as it is presented to the user.

Portlets are not difficult to build, and Java standard portlets can be written by any Java developer with experience in writing web applications. Liferay provides a Plugins Software Development Kit that makes creating portlet projects easy. For further information about the Plugins SDK, please see *Liferay in Action*, published by Manning Publications, which is the official guide to Liferay development.

Additionally, Liferay supports portlets written in other programming languages, such as PHP, Ruby, Groovy, or Python. Sample portlets written in these languages can be checked out from our Subversion repository (<http://svn.liferay.com/repos/public/plugins/trunk/portlets>).

Navigating Liferay

Assuming that you've followed the instructions in Chapter 2 for removing the demo web site (7 Cogs), Liferay initially presents a very simple interface. Unauthenticated users can navigate the public pages of the portal and will see a Sign In link in the top right corner of the screen.

To sign into Liferay for the first time, you can click the *Sign In* link. You will then be presented with the **Sign In Portlet**. This portlet allows a user (or a prospective user) to do several things: sign in to Liferay, create a new account on the portal, or have a password reminder emailed if the user has forgotten his or her password. To sign in for the first time, don't create an ac-

count for yourself. We will do that later. If you were to create a new account on the portal for yourself now, it would be created using Liferay's defaults, which means the account would not have access to the administrative portlets you need in order to set up Liferay for your organization. For this reason, you will need to sign in as the default administrative user. This user's credentials are:

User Name: test@liferay.com

Password: test

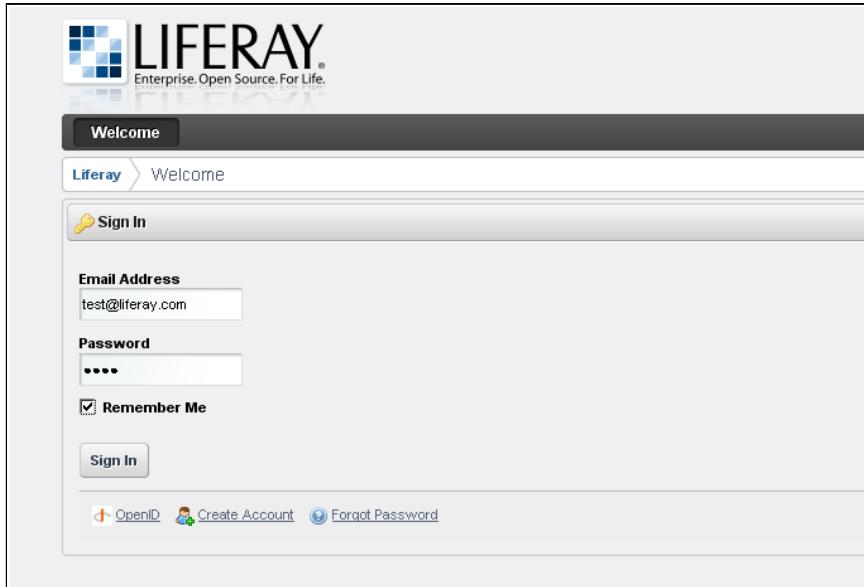


Illustration 12: Logging into Liferay Portal

Go ahead and sign into your new portal using these credentials. As you can see, Liferay by default uses one's email address as the user name. This can be changed later if you don't like this functionality, but it is generally a good practice to keep it this way. Users' email addresses are not normally things they will forget, and they are unique to each user, so they make good candidates for user IDs.

The first page that will be displayed when a user logs in for the first time is the Terms of Use page. This page is displayed to the user before he or she is allowed to access the portal. By default, your users will have to agree to your terms of service before they will be allowed to used the portal. This page can be customized to contain whatever text you want, or the feature can be disabled altogether. To continue, you will need to agree to the Terms of Service.

Once you log in as the administrative user, you will see that the Dockbar has now appeared across the top of the page. The Dockbar is the primary tool logged in users have for navigating the portal and accessing administrative functions from anywhere on the web site. Depending on the logged-in user's roles and what section of the website the user is viewing, you may see all or only some of the options available in the Dockbar.

As an administrator, the first option you will see on the Dockbar is *Add*. Mousing over *Add* will reveal a list of things that you can add. You can use this to add a new page at the current navigation level, or add portlets to the current page. When you first pull down the menu, you will see a list of common portlets that you can click on to add to the page. You will also see a *More* option, which will show you all of the currently available portlets. From the expanded *More* menu, you can add portlets to the current page. If you want to add a portlet to the current page, you can click the *Add* button next to a portlet to add it to the first column in the page, or drag the portlet from the menu to where you want it on the page.

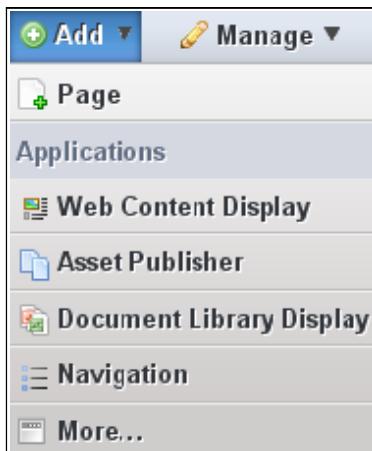


Illustration 13: Add Menu from the Dockbar

The next option you'll see is the *Manage* menu. From this menu, you can access various settings for the current page and any of its subpages. The items available are *Page*, *Page Layout Sitemap*, and *Settings*. Clicking on *Page Layout* brings up a dialog box which enables you to choose the layout template to use for the current page. The other settings are the same as their counterparts in the *Control Panel*, and are covered in detail later in this chapter. The last item in the menu is *Control Panel*; clicking on it brings you to the Control Panel.

The next thing you'll see is a check box labeled *Toggle Edit Controls*. This lets you turn on and off the edit controls in the top of the portlet windows.

This is helpful for administrators who want to look at a page they're working on and see it the way a regular user would.

If you roll your mouse over *Go to*, the Dockbar will expand, showing all of the places in the portal to which you have access. Initially, the place you are on is highlighted. You will see that you are in the *liferay.com* community, on the public pages. Liferay allows for various configurations of pages for end users: you can configure it so that some or all users have their own pages, public and private (or both), upon which they can place the portlets they need to use. The administrator account by default has its own pages. Because you are logged in with an account that has Administrator privileges, you can see everything in the portal.

One of the most important tools Liferay offers for managing your portal is the aforementioned *Control Panel*. The Control Panel is composed of administrative portlets that you can use to manage various aspects of the portal.

Navigating the Control Panel

The control panel is very easy to navigate. On the left side is a list of headings with functions underneath them. The headings are in alphabetical order, but the functions are in a logical order.

User Name: The first section is always the logged in user's personal space. Here, you can change your account information and manage your own personal pages.

Configuration

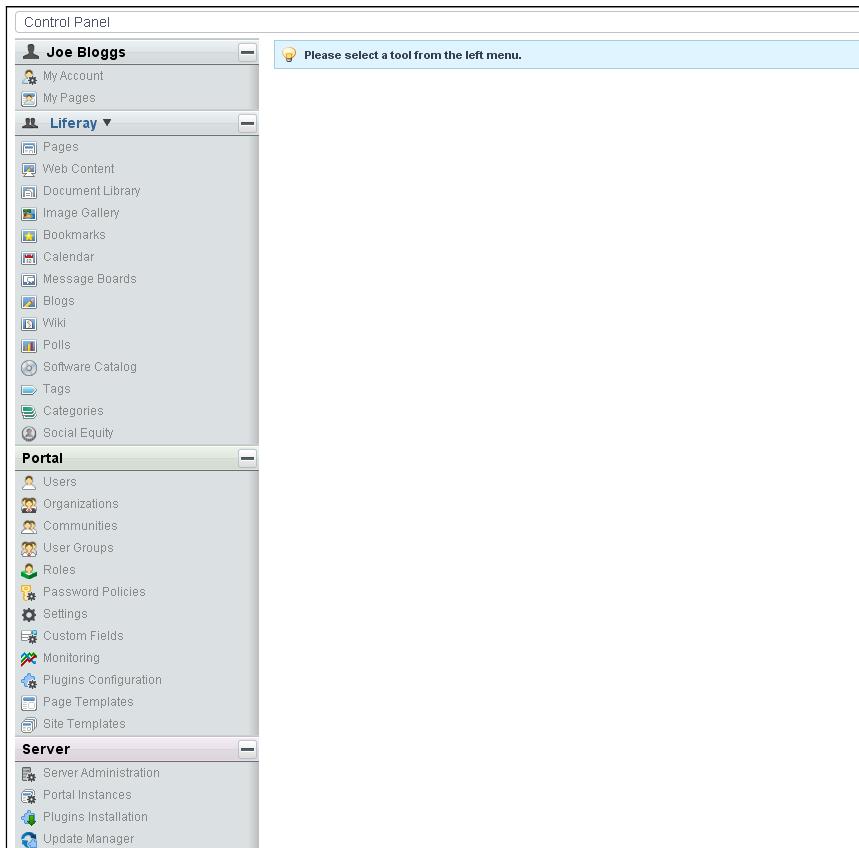


Illustration 14: Liferay's Control Panel

Content: The Content section contains links to all of Liferay's content management functions. You can maintain web content, documents, images, bookmarks, a calendar, administer a message board, configure a wiki, and more. The title of this section appears as the name of the community or organization whose content you are currently managing, and you can switch to another one at any time.

Portal: The Portal section allows portal administrators to set up and maintain the portal. This is where you can add and edit users, organizations, communities, roles, and configure the settings of the portal.

Server: The Server section contains administrative functions for configuring portal instances, plugins, and more.

All of the functions that you will need to maintain the portal or its content can be found in the control panel. Additionally, developers can write portlets which can also be added to the control panel. For further informa-

tion about this, you can take Liferay's Portal Developer course or see the official guide to Liferay development, *Liferay in Action* .

Portal Architecture

Before we dive into the user interface for adding and maintaining various portal resources, it is best to go over the concepts Liferay uses to organize a portal.

Portals are accessed by Users.

Users can be collected into User Groups.

Users can belong to Organizations.

Organizations can be grouped into hierarchies, such as Home Office → Regional Office → Satellite Office.

Users, Groups, and Organizations can belong to Communities that have a common interest.

Within Organizations and Communities, users can belong to Teams, which are groupings of users for specific functions within a community or organization.

The simplest way to think about this is that you have users and various ways those users can be grouped together. Some of these groupings follow an administratively organized hierarchy, and other groupings may be done by the users themselves (such as different users from multiple organizations starting a community called “Dog Lovers” that has a common interest in dogs). And other groupings may be done administratively via Roles for other functions that may cut across the portal (such as a Message Board Administrators role made up of users from multiple communities and organizations, allowing those users to administer any message board in the portal).

This way of organizing portal concepts may be illustrated as shown on the next page.

In the illustration below, each arrow may be read using the words “can be a member of.” So this means that Organizations can be members of Communities, Communities can be members of Roles, Users can be members of anything, and so on. Though this seems very complex, it provides a powerful mechanism for portal administrators to configure portal resources and security in a consistent and robust manner. It is important to note that the diagram illustrates only users and their collections. Permissions do not flow through all of these collections: permissions can be assigned to roles only.

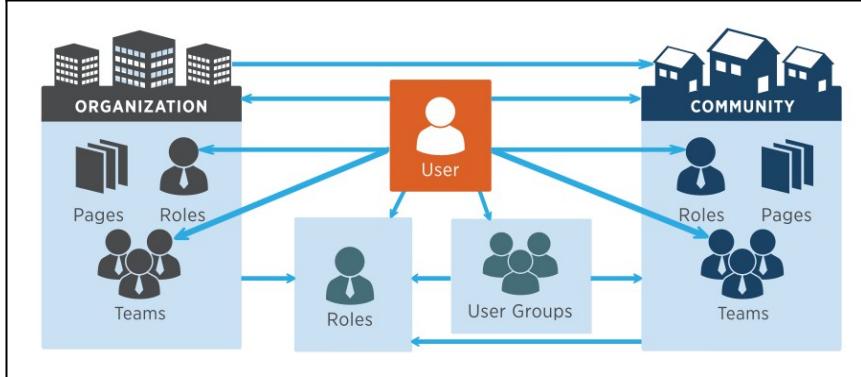


Illustration 15: Liferay permissions model. Don't worry, it's not as complicated as it seems.

Teams are inside individual organizations and communities, and are only available as they are created within those organizations and communities. Roles that appear inside organizations and communities are roles that are scoped just for organizations and communities. This means that though each organization and community in the portal has this role with its configured permissions, membership in this role is different for each organization and community.

Users

Users represent physical users of the system. These are the user accounts that people use to log into the system. By default, users get their own private communities with public and private pages that they can manage themselves, and this can be turned off or locked down by administrators. But this personal space is important: it's what enables users to have their own public blog or their own private calendar, a place to store their documents, and more.

Users can be collected in multiple ways. They can be members of organization hierarchies, such as Liferay, Inc. → Security → Internet Security. They can be collected into arbitrary user groups, such as Bloggers, which could be used to set apart users who get a Blog page in their personal space from users who do not. They can be members of communities which draw together common interests. And they can have roles which define their permissions in the system, and these roles can be scoped by Portal, Organization, or Community.

User Groups

User Groups are simple, arbitrary collections of users, created by administrators. They can be members of communities or roles. Permissions cannot be assigned to User Groups. Though User Groups do not have pages like some of the other collections of users (such as Communities or Organizations), they

do have page templates which can be used to customize users' personal sets of pages. This will be fully described below.

Roles

There are three kinds of roles:

- Portal Roles
- Organization Roles
- Community Roles

These are called role *scopes*. Roles are used to define permissions across their scopes: across the portal, across an organization, or across a community. For example, consider a role which grants access to create a Message Board category. A Portal role would grant that access across the portal, wherever there was a Message Board portlet. A Community role would grant that access only within a single community. An Organization role would grant that access only within an Organization.

Because Roles are used strictly for portal security, they also do not have pages, like Communities and Organizations.

Users, User Groups, Communities, or Organizations can be members of a role.

Organizations

Organizations are hierarchical collections of Users. They are one of the two types of portal resources that can have pages. There is also a special type of Organization called a *location*, which can define where users are specifically located.

Organizations are handy for defining where a user belongs in a particular hierarchy. For example, if you are implementing Liferay for a large organization, it may help to define user Joe Smith via his position in the organization chart. If Joe Smith is a Sales Engineer located in the New Jersey office, working in the North East division of the Sales department, he might be a member of the following organizations:

- Sales
- North East Division
- New Jersey Location

Now say that you have placed an Asset Publisher portlet as a static portlet on every user's home page (via a User Group page template) so that you can inform employees of various announcements via the content management system. If you tagged your content appropriately, you could ensure that Joe Smith gets any announcements that are meant for Sales, the North East Division, or the New Jersey location.

Organizations can be members of Communities.

Communities

Communities are collections of Users who have a common interest. Liferay's default pages are part of a community named for the portal, because everyone—whether they are anonymous or members of the portal—has a common interest in the default, public pages of your site. There are three types of Communities:

- Open
- Restricted
- Hidden

An Open Community (the default) allows portal users to join and leave the Community whenever they want to, using the Control Panel or a Communities portlet added to a page which they can access. A Restricted Community requires that users be added to the Community by a community administrator. Users may use the Control Panel or the Communities portlet to request membership. A Hidden community is just like a Restricted community, with the added concept that it does not show up at all in the Communities portlet or the Control Panel. Users will have to be added to a hidden community by a community administrator.

Teams

Teams are unique within a context of a Community or Organization. Teams are essentially sets of users that can be created within a community. This makes teams different from the Community and Organization Roles because teams appear only in the specific community or organization in which they are created. This is very useful if you need to create a team of users for a specific purpose within a community or organization and not for each community or organization in the portal.

Teams can also be essential for some use cases, because they can be created by Community or Organization Administrators. Community and Organization Administrators cannot create roles, so the ability to have teams empowers them to manage permissions at a level they weren't capable of previously.

Using the Control Panel

The Portal section of the Control Panel is used for most administrative tasks. You will find there an interface for the creation and maintenance of

- Users
- Organizations

- Communities
- User Groups
- Roles

Additionally, it allows you to configure many server settings, including:

- Password Policies
- Authentication options, including Single Sign-On and LDAP integration
- Default User Associations
- Reserved Screen Names
- Mail Host Names
- Email Notifications

You will use the Portal section of the Control Panel to create your portal structure, implement security, and administer your users. Note that only users with the Administrator role—a portal scoped role—have permission to view this section of the Control Panel. You can, of course, grant permissions to one or more sections to roles that you create yourself.

Adding Users

Let's begin by adding a user account for yourself. We will then configure this account so that it has the same administrative access as the default administrator account. Go up to the Dockbar, mouse over *Manage* and click the

The screenshot shows the 'Users' section of the Control Panel. At the top, there are buttons for 'View All', 'Add' (which is highlighted in green), and 'Export'. Below this is a 'New User' section with a 'Details' tab. The 'Details' tab contains fields for Title (dropdown), Screen Name (text input), Email Address (text input), First Name (text input), Middle Name (text input), Last Name (text input), and Suffix (dropdown). To the right of these are fields for Birthday (date picker showing January 1, 1970) and Gender (dropdown showing Male). A 'Job Title' field is also present. A modal dialog titled 'User Information' is overlaid on the page, showing the 'Details' tab and a 'Save' button. The background shows a faint watermark of a person using a computer.

Illustration 16: The Add User screen.

Configuration

Control Panel link, if you aren't there already. Then under the *Portal* category, click *Users*. Click the *Add* button.

You will then be presented with the Add User form. Fill out the form using your name and email address. When you are finished, click *Save*.

The page will then reappear with a message saying that the save was successful, and there will now be an expanded form which allows you to fill out a lot more information about the user. You don't have to fill anything else out right now, but one thing is important to note: when the user ID was created, a password was automatically generated and, if Liferay has been correctly installed (see Chapter 2), an email message with the password in it was sent to the user. This of course requires that Liferay can properly communicate with your SMTP mail server in your organization.

The screenshot shows the Liferay Control Panel's User account editor. At the top, there's a header bar with 'Users' and navigation links for 'View All', 'Add', and 'Export'. Below the header, the user 'John Smith' is selected. On the left, a sidebar lists 'Details' (Title dropdown, Screen Name 'jsmith', Email Address 'jsmith@liferay.com', First Name 'John', Middle Name, Last Name 'Smith', Suffix dropdown), 'User ID' (10406), 'Birthday' (January 1, 1970), 'Gender' (Male dropdown), and 'Job Title' (empty input field). On the right, a large panel titled 'User Information' contains a profile picture placeholder for 'John Smith'. It includes sections for 'Identification' (Addresses, Phone Numbers, Additional Email Addresses, Websites, Instant Messenger, Social Network, SMS, OpenID) and 'Miscellaneous' (Announcements, Display Settings, Comments, Custom Fields). At the bottom right of the panel are 'Save' and 'Cancel' buttons.

Illustration 17: Liferay's User Account editor.

If you haven't yet set up your mail server, you'll need to use this screen to change the default password for the user ID to something you can remember. You can do this by clicking on the *Password* link in the box on the right, entering the new password in the two fields, and clicking Save.

Next, you will want to give your user account the same administrative rights as the default administrator's account. This will allow you to perform administrative tasks with your own ID instead of having to use the default ID. And this allows you to make your portal more secure by deleting or disabling the default ID.

Click the *Roles* link. You will then be taken to a screen which shows the roles to which your ID is currently assigned. By default, you should have one role: Power User. By default, all users are also assigned the Power User role. You can give this role certain permissions if you wish or disable it altogether (we will see how to do this later). You can also define the default roles a new user receives; we will go over this later also.

To make yourself an Administrator, click the *Select* link. A window will pop up with a list of all the roles in the system. Select the Administrator role from the list and the window will disappear and you will see that the role has been added to the list of roles to which you are assigned. Next, click the *Save* button, which is at the bottom of the blue bar of links on the right. You are now an administrator of the portal. Log out of the portal and then log back in with your own user ID.

User Management

If you click the *Users* link on the left of the Control Panel, you will see that there are now two users in the list of users. If you wanted to change something about a particular user, you can click the *Actions* button next to that user.

Edit User: This takes you back to the Edit User page, where you can modify anything about the user.

Permissions : This allows you to define which Roles have permissions to edit the user.

Manage Pages: If the user has pages, this allows you to edit them.

Impersonate User: Opens another browser window which allows you to browse the site as though you were the user.

Deactivate: Clicking this will deactivate the user's account.

Note that most users will not be able to perform most of the above (in fact, they won't even have access to this section of the Control Panel). Be-

cause you have administrative access, you can perform all of the above functions.

Organizations

Organizations in Liferay are meant to model organizations in real life. They can be used to represent different companies, non-profit organizations, churches, schools, clubs, and so on. They have been used to represent a sports league, with various sports (soccer, baseball, basketball, etc.) and their teams as sub-organizations. If you have a collection of users that all belong to the same grouping, you may be able to model that as an organization.

Your portal may have only one organization or several, depending on what kind of site you are building. For example, a corporate site may model its own organization hierarchy in Liferay, while a social networking site may have users from many separate organizations who access the site. Organizations can be organized in a hierarchy to unlimited levels, and Users can be members of one or many organizations—inside of a hierarchy or across different hierarchies.

Additionally, Organizations can be associated with Roles. One application of this in a corporate setting could be an IT Security group. You may have an organization within your IT organization that handles security for all of the applications company-wide. If you had users as members of this organization, you could grant the Administrator role you just granted to your own ID to the whole Organization, thereby giving the members of the IT Security organization administrative access to the portal. If a user in this organization later was hired by the Human Resources department, the simple administrative act of moving the user from the IT Security organization to the HR organization would remove this privilege from the user, since the user would no longer be in an organization that has the Administrator role. By adding the user to the HR organization, any roles the HR organization has (such as access to a benefits system in the portal) would be transferred to the user. In this manner, you can design your portal to correspond with your existing organization chart, and have users' permissions reflect their positions in the chart.

Of course, this is only one way to design it. If you have more complex requirements, you can combine Organizations with Teams and scoped Roles to assemble the sets of permissions you wish to grant to particular users.

Organizations are one of two types of Liferay resources (the other being Communities) that can have its own pages. This allows members of the organizations (if they are granted the Manage Pages permission) to maintain their own pages. They can have a set of public pages which include information and applications appropriate for guests or logged in users who are not

members of the Organization to make use of (such as a help desk ticket entry system for an IT page), and they can have a set of private pages with applications for the organization's own use (such as the back-end portlets of the same ticketing system).

To add an organization, click the *Organizations* link on the left side of the Control Panel, and then click the Add button.

Spartan Software

Details

Name
Spartan Software

Type
Regular Organization

Group ID
10702

Parent Organization

Select

Organization Information

- Details**
- [Pages](#)
- [Categorization](#)

Identification

- [Addresses](#)
- [Phone Numbers](#)
- [Additional Email Addresses](#)
- [Websites](#)
- [Services](#)

Miscellaneous

- [Comments](#)
- [Reminder Queries](#)
- [Custom Fields](#)

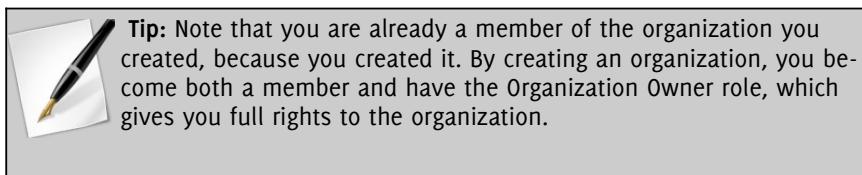
Save **Cancel**

Illustration 18: Adding an organization.

Name: The name of the organization.

Type: Use this to choose whether this is a regular organization or a location.

Parent Organization: Click the *Select* link to bring up a window which allows you to select the organization in the system that is the direct parent of the organization you are creating. Click the *Remove* button to remove the currently configured parent.



Fill out the information for your organization and click *Save*.

As before with users, the form reappears and you can enter more information about the organization. Organizations can have multiple email ad-

dresses, postal addresses, web sites, and phone numbers associated with them. The *Services* link can be used to indicate the operating hours of the organization, if any.

For now, click the *Back* button. This will take you back to the list of organizations.

Click the *Actions* button next to the new organization you have created. You will then see the many actions you can take to manipulate this organization.

Edit: Lets you edit the organization.

Manage Pages: Lets you create and manage public and private pages for the organization.

Manage Teams: Lets you create teams within this organization, to which you can assign users and permissions.

Assign User Roles: Lets you assign Organization-scoped roles to users. By default, Organizations are created with three roles: Organization Administrator, Organization Member, and Organization Owner. You can assign one or more of these roles to users in the organization. All members of the Organization get the Organization Member role.

Assign Members: Takes you to a screen where you can search and select users in the portal to be assigned to this organization as members.

Add User: Adds a new user in the portal who will be a member of this organization.

View Users: Shows a list of users who are members of this organization.

Add Regular Organization: Lets you add a child organization to this organization. This is how you create hierarchies of organizations with parent-child relationships.

Add Location: Lets you add a child Location, which is a special type of organization that cannot have any children added to it.

View Sub organizations: Shows a list of all the organizations that are children of this organization.

Delete: Deletes this organization from the portal. You will have to ensure that the organization has no users in it first.

Communities

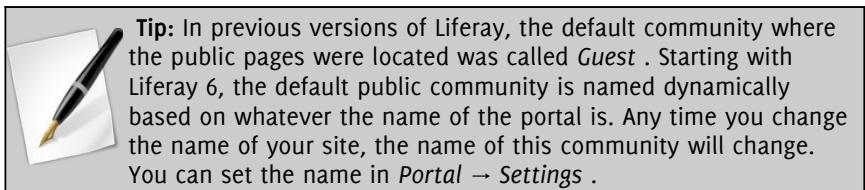
Communities are very much like Organizations except that they are not hierarchical. They are designed instead to be islands to themselves which anyone from any organization (or from no organization at all) can join. You

can use Communities, therefore, in any situation where you need to cut across the organizational structure of your portal, or where you have a site that would apply to almost anybody.

For example, a corporate Intranet running Liferay may have sites for all the organizations in the company: Sales, Marketing, product groups, Information Technology, Human Resources, and so on. But what about the corporate health and fitness center? That's something that everybody in the company—regardless of organization—potentially has an interest in, and may want to join. That's a good candidate for a Community. Using the same scenario, the home page for the Intranet is probably best placed in a community that any member of the portal can access.

For other kinds of web sites, you may want to use communities to bring people together who have a common interest. If you were building a photo sharing web site out of Liferay, you may have communities based on the types of photos people want to share. So those who enjoy taking pictures of landscapes can join a Landscapes community, and those who enjoy taking pictures of sunsets can join a Sunsets community. And if they lose interest, they can leave those communities too.

The default home page in Liferay is in a community which by default is called *liferay.com* (you can rename it; see the tip below), and this is where you would put your public web site. As you can see, there are several scenarios in which you would want to use something like a community instead of an organization, and this is why they have distinct roles within Liferay Portal.



Tip: In previous versions of Liferay, the default community where the public pages were located was called *Guest*. Starting with Liferay 6, the default public community is named dynamically based on whatever the name of the portal is. Any time you change the name of your site, the name of this community will change. You can set the name in *Portal → Settings*.

Communities can be created and managed in two ways. The first is through the Control Panel, like every other user/page collection in Liferay. The second is through the My Communities portlet, which can be added to any page in Liferay. Why are there two ways? Because the My Communities portlet also doubles as a way to navigate from community to community, and allows users to browse the list of communities and select whether or not they want to join one (if it is open or restricted). This enables you as a portal administrator to provide users with this functionality without giving them access to the Control Panel.

To add a community, click the *Communities* link on the left side of the Control Panel in the Portal section, and then click the *Add* button.

Name: Enter the name of the community you wish to create.

Description: Enter some descriptive text about the community.

Type: There are three kinds of communities: Open, Restricted, and Private. An open community appears in the My Communities portlet and users can join and leave the community whenever they want. A restricted community is the same except users can only request membership. A community administrator must then explicitly grant or deny users' requests to join. A private community does not appear in the My Communities portlet and users must be added to it manually by a community administrator.

Active: Communities can be active or inactive. If a community is inactive, no data can be added to it.

Tags: You can use Liferay's tagging mechanism on the community. This is helpful if the community has a specific, topical purpose within the portal.

Once you have created a community, it will appear in the list of communities in the Control Panel. The operations you can perform on it are very similar to the operations you can perform on organizations.

Edit: Lets you edit the community.

Manage Pages: Lets you create and manage public and private pages for the community.

Manage Teams: Lets you create and manage teams for the community.

Assign User Roles: Lets you assign community-scoped roles to users. By default, communities are created with three roles: Community Administrator, Community Member, and Community Owner. You can assign one or more of these roles to users in the community. All members of the community get the Community Member role.

Assign Members: Takes you to a screen where you can search and select users in the portal to be assigned to this community as members.

Join/Leave: If you are not a member of the community, you will have a Join or Request Membership option. If you are a member of the community you will see an option to leave the community.

Delete: Users with administrative access to the portal or who are owners of the community can delete it.

Site Templates

While we're on the subject of communities, it is important to mention Site Templates, which is several links down in the Portal category in the Control Panel. These allow you to create web sites within communities by select-

ing from pre-defined templates that you can make ahead of time. What this means is that you can create a template community that has a pre-defined set of pages and portlets, and then use that template to very quickly create multiple communities that are pre-populated with those pages and portlets.

The screenshot shows the 'Site Templates' portlet interface. At the top, there's a green header bar with the title 'Site Templates'. Below it, a toolbar has 'View All' and 'Add' buttons. The main area is titled 'Student Template'. It contains fields for 'Name' (set to 'Student Template') and 'Other Languages (1)' (with an American flag icon). A large 'Description' text area is empty. A checked checkbox labeled 'Active' is present. Under 'Configuration', there's a link 'Open site template (Opens New Window)'. At the bottom are 'Save' and 'Cancel' buttons.

Illustration 19: Site Templates

You can create templates for open, restricted, and private communities. Additionally, you can create a default template that applies to all kinds of communities. For our example, we will work with a template designed for student communities.

Go to the Control Panel and click *Site Templates*. Click the *Add* button and create a site template called *Students*. Make the template Active. Now you can select the *Open Site Template* link and begin adding content, pages, portlets and configuring the layouts. You'll learn more about how to do this in the next chapter. Once you're finished, return to the *Site Templates* portlet and select *Save*.

To create a community based on the new template, go to the Control Panel and click *Communities*. Click the *Add* button and create a community called *Freshmen*. In the Public Pages drop down menu, select *Student Template* and then click *Save*. The new community will have all the pages, content, and portlets you created in the template. This feature streamlines the community creation process for administrators, making it very easy to quickly create communities.

User Groups

User Groups are arbitrary groupings of users. These groups are created by portal administrators to group users together who don't have an obvious organizational or community-based attribute or aspect which brings them together. Groups cannot have permissions like roles, but User Groups can be added to Roles. Why would you use User Groups, then? They come into play when you have complex security requirements and for page templates, which we will discuss below.

Creating a User Group is easy. Click the *User Groups* link and then click the *Add* button. There are only two fields to fill out: Name (the name of the User Group) and Description (an optional description of what the group is for). Click *Save* and you will then be back to the list of groups.

As with the other resources in the portal, you can click the *Actions* button to perform various operations on User Groups.

Edit: Allows you to modify the name or description of the User Group.

Permissions: This allows you to define which Users, User Groups, or Roles have permissions to edit the User Group.

Manage Pages: Though User Groups don't have pages of their own, you can create page templates for a group. When a User Group has page templates, any users added to the group will have the group's pages copied to their personal pages. This allows you to do things like create a Bloggers user group with a page template that has the Blogs and Recent Bloggers portlets on it. The first time users who are added to this group log in to the portal, this page will get copied to their personal pages. They will then automatically have a blog page that they can use.

Assign Members: Takes you to a screen where you can search for and select users in the portal to be assigned to this User Group.

View Users : Lets you view the users who are in the User Group.

Delete: Deletes the User Group.

User Groups and Page Templates

Liferay allows users to have a personal set of public and private pages that each user can customize at will. The default configuration of those pages can be determined by the portal administrator through the `portal-ext.properties` file and optionally by providing the configuration in a LAR file. Though this has been a long-time feature of Liferay, it was not very flexible or easy to use.

Liferay version 5.1 introduced the concept of page templates which are tied to User Groups. This enables administrators to provide the same configuration for the personal pages of all (or just a subset of) users, using Liferay's GUI instead of the properties file. In some cases you may want to provide a different configuration for each user depending on his or her profile. For example, in a portal for University students, staff and undergraduates would get different default pages and portlets in their personal space. You can also set it up so that different groups are combined together to create the desired default configuration. When a user is assigned to a user group, the configured page templates are copied directly to the user's personal pages.

User Group Page Templates: Defining page templates for a user group

A User Group's page templates can be administered using the Control Panel. The User Groups link lists all of the existing user groups and allows you to perform several actions on each of them.

The screenshot shows the 'User Groups' section of the Liferay Control Panel. At the top, there is a success message: 'Your request processed successfully.' Below the header, there are buttons for 'View All' and 'Add'. A search bar is present. On the left, there is a 'Delete' button. The main area displays a table with two rows: 'Staff' and 'Students'. The 'Name' column is sorted by 'Name'. To the right of the table is a context menu with the following options: 'Edit', 'Permissions', 'Manage Pages' (which is highlighted), 'Assign Members', 'View Users', 'Delete', and two 'Actions' buttons. At the bottom of the table area, it says 'Showing 2 results.'

Illustration 20: Manage Pages action on a User Group

By selecting the Manage Pages action the administrator can access the common Liferay UI for creating pages and organizing them in a hierarchy.

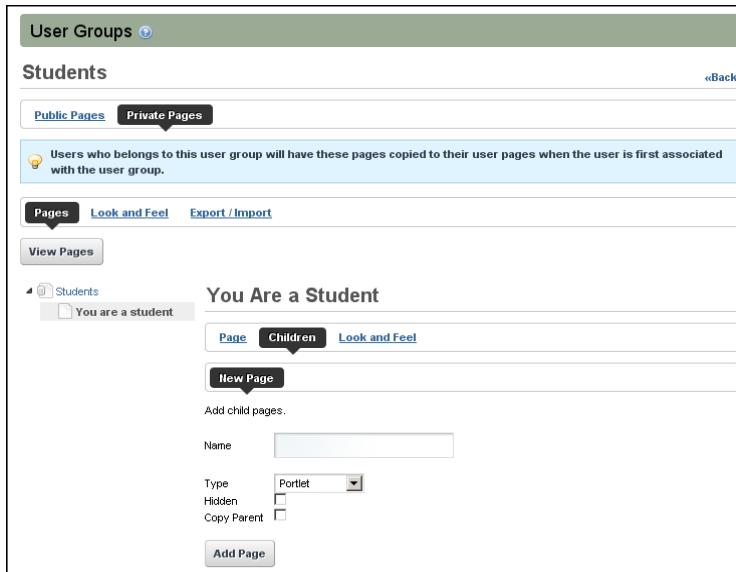


Illustration 21: Adding a Page Template

Note that it is possible to create both public and private pages. Each set will be used as templates to be copied to the user's personal public or private page sets respectively when the user becomes a member of the user group.

In the screen shot above, the administrator has created a new private page called *You are a student* within the Students user group. Since the page created is a portlet page, the administrator can now click the *View Pages* button to open the page and add as many portlets as desired to that page and configure them as needed. Let's assume for this example that the Loan Calculator and Calendar portlets are selected.

Applying the page templates by assigning members to the user group

The next step is to assign an existing user to that group to verify that the page template is copied as a user's private page. To do this, click *Actions* → *Assign Members* in the list of available user groups.

The screenshot shows a user interface for managing user groups. At the top, a green header bar contains the text "User Groups" and a help icon. Below this, a title "Students" is displayed. A navigation bar at the top right includes tabs for "Current" (which is selected) and "Available". A dropdown menu labeled "Match" with the option "All" is shown. Below the search controls, there are six input fields arranged in two rows of three: "First Name", "Middle Name", "Last Name" in the top row, and "Screen Name", "Email Address", "Active" (with a dropdown menu showing "Yes") in the bottom row. To the left of the "Active" field is a "Search" button. Below the search area are two links: "« Basic" and "Update Associations".

Illustration 22: Assigning Members to a User Group

By clicking the *Available* tab in the next screen, a list of all available users is shown. From that list, one or more users can be selected to make them members of the user group. When the *Update Associations* button is clicked, the users become members of the group and copies of any public or private page templates which are configured for the user group are copied to their page sets.

In the previous example, a user that already had an existing page called *Welcome* will now have a new page called *You Are A Student* the next time she accesses her personal space. That page will contain two portlets: Loan Calculator and Calendar as configured by the User Group administrator.

Additional details

Because the pages are copied to a user's set of pages, those pages are now owned by the user and they can be changed at any time if the portal is set up to allow users to edit their personal pages. When a user is removed from a user group the associated pages won't be removed: they have become that user's pages. The system is smart enough, however, to detect when a user is added again to a group of which he or she was already a part, and the pages are not added again.

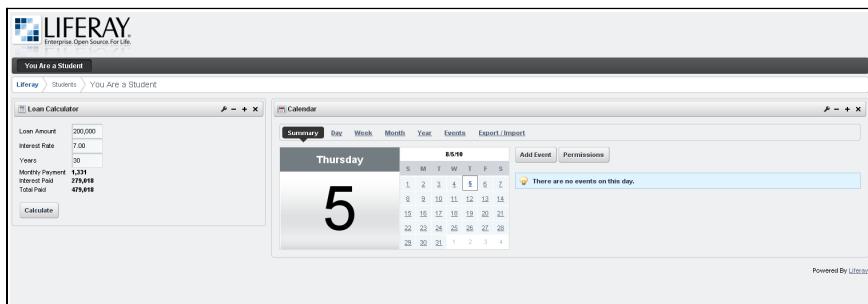


Illustration 23: Template copied to a user's page set

If an administrator modifies page templates for a User group after users have already been added to the group, those changes will be used when new users are assigned to the user group. Since the pages are templates, however, the changes won't be applied to users that were already members of the user group.

Composing A Page Out of Several User Groups

Users can belong to many user groups. If you have templates defined for a number of groups, this may result in having many page templates copied to users' pages. To prevent this, you can combine pages from different user groups into a single page.

Let's expand our previous example by dividing the Students into First Year Students, Second Year Students, Third Year Students, International Students, and Prospective Students. For each of these types of students we want to have a page with the Loan Calculator and Calendar, but depending on which type, we also want other different portlets to be on that page too.

This can be achieved by using a naming convention for the pages. If two or more pages of different user groups have the same name, they will be combined into a single page when they are copied to a user's personal pages set.

In the example above, a User was added to a Students group which had a page called You are a Student. If the administrator creates a page template with the same name (You are a Student) in the First Year Students group and puts in it an RSS portlet pointing to information interesting for them, that page would be combined with the You are a Student page that's in the Students group, and the resulting page would contain the portlets configured for both User Groups:

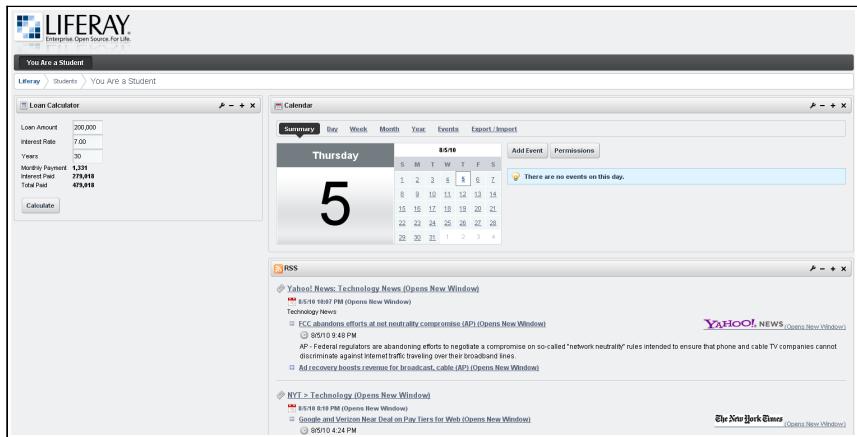


Illustration 24: Combined portlet pages.

Page Combination Rules

The following rules are used when composing a page by combining pages from different user groups:

- If a user becomes a member of a User Group that has a page template with the same name in the same set (public or private) as a page that the user already has, those pages will be combined.
- If any of the pages has the name translated to several languages, only the default language is considered in the comparison.
- The portlets on the new page will be copied to the bottom of the equivalent columns of the existing page.
- If the existing and the new pages have different layout templates, the existing one is preserved.
- If the new layout template has portlets in columns that do not exist in the existing page, those portlets will be automatically copied to the first column of the existing layout template.

As you can see, it is possible to have a very flexible configuration for the default pages of portal users. Furthermore, that configuration can be changed at any time using the UI administrators are used to and then assigning users to new user groups.

While these examples are somewhat simple, the system allows for as many user groups as desired. By using the convention of matching the page names, it is possible to build any default page composition that you want for your users.

Roles

Roles are groupings of users that share a particular function within the portal, according to a particular scope. Roles can be granted permissions to

various functions within portlet applications. Think of a role as a description of a function, such as Message Board Administrators. A role with that name is likely to have permissions to functions of the Message Board portlet delegated to it. Users who are placed in this role then inherit those permissions.

Roles are scoped by Portal, Organization, or Community. The Control Panel makes it easy for you to assign users to Roles and to assign permissions to Roles. You only have to go to one place: the Roles link. From there, you can add roles scoped by Portal, Organization, or Community from one interface.

To create a Role, click the *Roles* link, and then click the *Add* button. Type a name for your role and an optional description. The drop down box at the bottom of the form lets you choose whether this is a Regular, Community, or Organization role. When you have finished, click *Save*.

You will be back at the list of roles. To see what functions you can perform on your new role, click the *Actions* button.

Edit: Click this action to edit the role. You can change its name or description.

Permissions : This allows you to define which Users, User Groups, or Roles have permissions to edit the Role.

Define Permissions : Click this to define what permissions this role has. This is outlined in the next section.

Assign Members: Takes you to a screen where you can search and select users in the portal to be assigned to this role. These users will inherit any permissions given to the role.

View Users : Lets you view the users who are in the Role.

Delete: Deletes the Role.

Defining Permissions on a Role

Roles exist as a bucket for granting permissions to the users who are members of them. So one of the main tasks you will be doing with a role is granting it the permissions that you want members of the role to have.

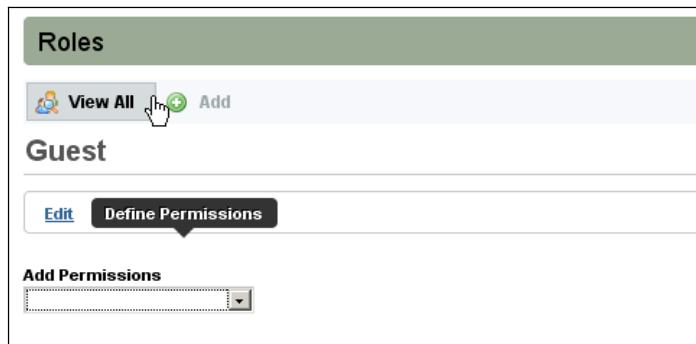


Illustration 25: Defining Permissions on a Role

When you click the *Define Permissions* action on a Portal scoped Role, you are given a choice of four kinds of permissions that can be defined for this role: Portal, Content, Applications, and Control Panel. For other Roles, you need to use the *Permissions* link in individual portlets to assign permissions for the community or organization in which that portlet is placed.

Portal permissions cover portal-wide activities that are in several categories, such as Community, Location, Organization, Password Policy, etc. This allows you to create a Role that, for example, can create new Communities in the portal. This would allow you to grant users that particular permission without making them overall portal administrators.

Content permissions cover the content that the installed portlets create. If you pick one of the portlets from this list, you'll get options for defining permissions on its content. For example, if you pick Message Boards, you'll see permissions for creating categories or threads, or deleting and moving topics.

Application permissions affect the application as a whole. So if we stick with our Message Boards example, an application permission might define who can add the Message Boards to a page.

Control Panel permissions affect how the portlet appears to the user in the Control Panel. Some Control Panel portlets have a Configuration button, so you can define who gets to see that, as well as who gets to see an application in the Control Panel.

Configuration

The screenshot shows the 'Roles' section of the Control Panel. A breadcrumb trail at the top includes 'View All', 'Add', 'Message Board Administrator', and 'Edit'. Below this, tabs for 'Edit', 'Define Permissions', and 'Assign Members' are visible, with 'Define Permissions' being the active tab. A sub-header 'Add Permissions' has a dropdown menu set to 'Message Boards'. The main area displays two tables: 'Message Boards' and 'Message Boards Category', each listing actions with checkboxes, scopes, and 'Limit Scope' links.

Action	Scope	Limit Scope
Add Category	Portal	Limit Scope
Add File	Portal	Limit Scope
Add Message	Portal	Limit Scope
Ban User	Portal	Limit Scope
Move Thread	Portal	Limit Scope
Lock Thread	Portal	Limit Scope
Permissions	Portal	Limit Scope
Reply to Message	Portal	Limit Scope
Subscribe	Portal	Limit Scope
Update Thread Priority	Portal	Limit Scope

Action	Scope	Limit Scope
Add File	Portal	Limit Scope
Add Message	Portal	Limit Scope
Add Subcategory	Portal	Limit Scope
Delete	Portal	Limit Scope

Illustration 26: Message board content permissions

Each possible action to which permissions can be granted is listed. To grant a permission, choose the permission. If you want to limit the scope of the permission to a community or organization, click the *Limit Scope* link, and then choose the community or organization that is in scope.

Once you have chosen the permissions granted to this role, click *Save*. For a Message Boards Admin role, you would likely grant Content permissions to every action listed. After you click *Save*, you will see a list of all permissions that are currently granted to this role. From here, you can add more permissions or go back by clicking a link in the breadcrumb list or the *Return to Full Page* link.

Roles are very powerful, and allow portal administrators to define various permissions in whatever combinations they like. This gives you as much flexibility as possible to build the site you have designed.

Special Note about the Power Users Role

By default, many portlets within Liferay are configured so that Power Users have access to them, but regular users do not. If you decide to remove the Power Users role from the default user associations, you will need to modify the permissions on certain portlets. To do this, see the section on Plugins Configuration below.

Teams

Teams don't appear as a link in the Control Panel because they are user groupings that are *inside* communities and organizations. They exist because there will be times when you need to create a bucket for permissions within a community or organization that only affect that community or organization. That is when you create a team. Teams are a part of the community or organization in which they are created. If you create a team for one community, it is not available for another community. This is beneficial when you want a particular set of people and permissions for a specific one time function.

Creating a team for a community is simple. Go to *Control Panel → Communities* and select the *Actions→ Manage Teams*.

Select the *Add Team* button and then type in a name and add a description. Once you're finished, click *Save*. Your new team will appear in the list. To add members, simply click on *Actions → Assign Members*. That's it! Your team is ready.

The screenshot shows the 'Asset Publisher - Configuration' interface with the 'Permissions' tab selected. A table lists various user roles with checkboxes for 'View', 'Add to Page', and 'Configuration'. The 'Test Team' role has all three checkboxes checked. A 'Save' button is at the bottom.

Role	View	Add to Page	Configuration
Community Content Creator	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Guest	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Owner	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Portal Content Reviewer	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Power User	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Publisher	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
User	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Writer	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Community Content Reviewer	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Community Member	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Test Team	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Illustration 27: Assigning permissions to a team

Creating a team for the organization is just as simple. To create a team for an organization go to *Control Panel* → *Organizations*, choose the organization from the list, select *Actions* → *Manage Teams*, and perform the same steps you would for a community.

Permission management for teams is handled at the individual portlet level, using the *Permissions* button on the portlet itself. This enables users who wouldn't have access to all of the necessary options in the control panel to manage permissions through teams.

To give a team access to a particular portlet function, access the configuration menu for that portlet where it resides on a page, click the *Permissions* tab, check the boxes corresponding to permissions you want to assign to these teams, and then click *Save*.

Global Server Settings

Now that you have navigated in the Control Panel, you should be pretty familiar with how it works, and that all the options appear in the left navigation, their interfaces appear in the middle, and any sub-options appear on the right. We have focused so far on the maintenance of users and portal security. The remaining links in the *Portal* category focus on various portal settings which cover how the portal operates and integrates with other systems you may have.

Password Policies

Password policies can help to enhance the security of your portal. Using password policies, you can set password rules such as password strength, frequency of password expiration, and more. Additionally, you can apply different rule sets to different sets of portal users.

If you are viewing a page other than the Control Panel, go up to the Dock-bar and select *Control Panel*. Next, click on the *Password Policies* link on the left side of the screen in the *Portal* category. You will see that there is already a default password policy in the system. You can edit this in the same manner as you edit other resources in the portal: click *Actions* and then click *Edit*.

You will then see the Password Policy settings form:

Changeable: Selects whether a user can change his or her password.

Change Required: Selects whether a user must change his or her password upon first log in.

Minimum Age: You can choose how long a password must remain in effect before it can be changed.

Reset ticket max age: You can choose how long a password reset link remains valid.

Syntax Checking: Allows you to choose whether dictionary words can be in passwords as well as the minimum password length.

Password History: Keeps a history (with a defined length) of passwords and won't allow users to change their passwords to one that was previously used. You can enable it or disable it using the check box.

Password Expiration: Lets you choose an interval where passwords can be active before they expire. You can select the age, the warning time, and a grace limit, and you can enable or disable it using the check box.

Lockout: Allows you to set the number of failed log in attempts before a user's account becomes locked. You can choose whether an administrator needs to unlock the account or if it becomes unlocked after a specific duration. You can enable or disable it using the check box.

From the list of password policies, you can perform several other actions.

Edit: Brings you to the form above and allows you to modify the password policy.

Permissions: This allows you to define which Users , User Groups , or Roles have permissions to edit the Password Policy.

Assign Members: Takes you to a screen where you can search and select users in the portal to be assigned to this password policy. The password policy will be enforced for any users who are added here.

Delete: This shows up for any password policies that you add beyond the default policy. You cannot delete the default policy.

Settings

The Settings link is where most of the global portal settings are.

General: This lets you configure global settings, such as the company name, domain, the virtual host, a global portal logo, and more.

Authentication: Allows you to configure log in IDs, connection to LDAP, and Single Sign-On.

Users: Has three tabs, labeled *Fields*, *Reserved Credentials*, and *Default User Associations*. The Fields tab enables or disables some user fields, such as birthday or terms of use. The Reserved Credentials tab lets you reserve screen names and email addresses so that users cannot register using them. You might use this to prevent users from registering on the portal with user names that contain profanity or that sound official, such as *admin* or *presid-*

ent. The Default User Associations tab lets you configure default membership to Roles, User Groups, and Communities for new users, and provides a check box which allows you to retroactively apply these to existing users.

Mail Host Names: You can add a list of other mail host names that are associated with your organization. For example, your main domain might be `mycompany.com`, but you might use `mycompany-marketing.com` for your email newsletters. Any domain names associated with your organization can go here.

Email Notifications: Liferay sends email notifications for certain events, such as user registrations, password changes, etc. You can customize those messages here.

We will go over these settings in detail below.

General

The *General* link allows you to set the name of the company / organization / site which is running the portal. Setting the name here will also define the name of your portal's default community. By default it is `liferay.com`, so you will definitely want to set this to reflect your organization. You can also set the virtual host, the mail domain, and several other items of miscellaneous information about the organization.

Authentication: General Settings

The *Authentication* link has several tabs under it. All of these are used for configuring how users will authenticate to Liferay. Because Liferay supports a number of authentication methods, there are settings for each.

The general settings affect only Liferay functionality, and don't have anything to do with any of the integration options on the other tabs. This tab allows you to customize Liferay's out-of-box behavior regarding authentication. Specifically, the *General* tab allows you to select from several global authentication settings:

- Authenticate via email address (default), screen name, or user ID (a numerical ID auto-generated in the database—not recommended).
- Enable / Disable automatic log in. If enabled, Liferay allows a user to check a box which will cause the site to “remember” the user's log in by placing a cookie on his or her browser. If disabled, users will have to log in manually always.
- Enable / Disable forgotten password functionality.
- Enable / Disable request password reset links.

- Enable / Disable account creation by strangers. If you are running an Internet site, you will probably want to leave this on so that visitors can create accounts on your site.
- Enable / Disable account creation by those using an email address in the domain of the company running the site (which you just set on the General tab). This is handy if you are using Liferay to host both internal and external web sites. You can make sure that all internal IDs have to be created by administrators, but external users can register for IDs themselves.
- Enable / Disable email address verification. If you enable this, Liferay will send users a verification email with a link back to the portal to verify that the email address they entered is a valid one they can access.

By default, all settings except for the last are enabled. One default that is important is that users will authenticate by their email address. Liferay defaults to this for several reasons:

1. An email address is, by definition, unique to the user who owns it.
2. People can generally remember their email addresses. If you have a user who hasn't logged into the portal for a while, it is possible that he or she will forget his or her screen name, especially if the user was not allowed to use his or her screen name of choice (because somebody else already used it).
3. If a user changes his or her email address, it is more likely that the user will forget to update his or her email address in his or her profile, if that email address is not used to authenticate. If the user's email address is not updated, all notifications sent by the portal will fail to reach the user. So it is important to keep the email address at the forefront of a user's mind when he or she logs in to help the user keep it up to date.

For these reasons, Liferay defaults to using the email address as a user name.

Authentication: LDAP

Connecting Liferay to an LDAP directory a straightforward process through the Control Panel. There are still, however, two places in which you can configure the LDAP settings: the `portal-ext.properties` file (which will be covered in the next chapter) and the Control Panel, where the settings will get stored in the database. Note that if you use both, the settings in the database will override the settings in `portal-ext.properties`. For this reason, we recommend for most users that you use the Control Panel to configure the

LDAP settings—it's easier and does not require a restart of Liferay. The only compelling reason to use the `portal-ext.properties` file is if you have many Liferay nodes which will be configured to run against the same LDAP directory. In that case, for your initial deployment, it may be easier to copy the `portal-ext.properties` file to all of the nodes so that the first time they start up, the settings are correct. Regardless of which method you use, the settings are the same.

The LDAP settings screen is where you configure the global values.

Enabled/Required

Enabled: Check this box to enable LDAP Authentication.

Required: Check this box if LDAP authentication is required. Liferay will then not allow a user to log in unless he or she can successfully bind to the LDAP directory first. Uncheck this box if you want to allow users with Liferay accounts but no LDAP accounts to log in to the portal.

LDAP Servers

This is where you add LDAP Servers. If you have more than one, you can arrange the servers by order of preference using the up/down arrows. When you add an LDAP Server, you will need to provide several pieces of data so that Liferay can bind to that LDAP server and search it for user records. Regardless of how many LDAP servers you add, each server has the same configuration options:

Default Values

Several leading directory servers are listed here. If you are using one of these, select it and the rest of the form will be populated with the proper default values for that directory.

Connection

These settings cover the basic connection to LDAP.

Base Provider URL: This tells the portal where the LDAP server is located. Make sure that the machine on which Liferay is installed can communicate with the LDAP server. If there is a firewall between the two systems, check to make sure that the appropriate ports are opened.

Base DN: This is the Base Distinguished Name for your LDAP directory. It is usually modeled after your organization. For a commercial organization, it may look something like: `dc=companynamehere, dc=com`.

Principal: By default, the administrator ID is populated here. If you have removed the default LDAP administrator, you will need to use the fully

qualified name of the administrative credential you do use. You need an administrative credential because Liferay will be using this ID to synchronize user accounts to and from LDAP .

Credentials: This is the password for the administrative user.

This is all you need in order to make a regular connection to an LDAP directory. The rest of the configuration is optional: generally, the default attribute mappings are sufficient data to synchronize back to the Liferay database when a user attempts to log in. To test the connection to your LDAP server, click the *Test LDAP Connection* button.

If you are running your LDAP directory in SSL mode to prevent credential information from passing through the network unencrypted, you will have to perform extra steps to share the encryption key and certificate between the two systems.

For example, assuming your LDAP directory happens to be Microsoft Active Directory on Windows Server 2003, you would take the following steps to share the certificate:

On the Windows 2003 Domain Controller, open the *Certificates* MMC snap-in. Export the Root Certificate Authority certificate by selecting *Certificates (Local Computer) mmc snapin -> Trusted Root Certification Authorities -> MyRootCACertificateName*. Right click this certificate and select *All Tasks -> export -> select DER encoded binary X.509 .CER*. Copy the exported .cer file to your Liferay Portal server.

As with the CAS install (see the below section entitled **Single Sign-On**), you will need to import the certificate into the *cacerts* keystore. The import is handled by a command like the following:

```
keytool -import -trustcacerts -keystore  
/some/path/jdk1.5.0_11/jre/lib/security/cacerts -storepass changeit  
-noprompt  
-alias MyRootCA -file /some/path/MyRootCA.cer
```

The *keytool* utility ships as part of the Java SDK.

Once this is done, go back to the LDAP page in the Control Panel. Modify the LDAP URL in the **Base DN** field to the secure version by changing the protocol to *https* and the port to 636 like the following:

```
ldaps://myLdapServerHostname:636
```

Save the changes. Your Liferay Portal will now use LDAP in secure mode for authentication.

Users

This section contains settings for finding users in your LDAP directory.

Authentication Search Filter: The search filter box can be used to determine the search criteria for user log ins. By default, Liferay uses the email address as a user log in name. If you have changed this setting—which can be done on the *General* tab that's next to the *LDAP* tab in the *Settings->Authentication* section of the Control Panel—you will need to modify the search filter here, which has by default been configured to use the email address attribute from LDAP as search criteria. For example, if you changed Liferay's authentication method to use the screen name instead of the email address, you would modify the search filter so that it can match the entered log in name:

(cn=@screen_name@)

Import Search Filter: Depending on the LDAP server, there are different ways to identify the user. Generally, the default setting (*objectClass=inetOrgPerson*) is fine, but if you want to search for only a subset of users or users that have different object classes, you can change this.

User Mapping: The next series of fields allows you to define mappings from LDAP attributes to Liferay fields. Though your LDAP user attributes may be different from LDAP server to LDAP server, there are five fields that Liferay requires to be mapped in order for the user to be recognized. You must define a mapping to the corresponding attributes in LDAP for the following Liferay fields:

- Screen Name
- Password
- Email Address
- Full Name
- First Name
- Middle Name
- Last Name
- Job Title
- Group

The Control Panel provides default mappings for commonly used LDAP attributes. You can also add your own mappings if you wish.

Test LDAP Users: Once you have your attribute mappings set up (see above), click the *Test LDAP Users* button, and Liferay will attempt to pull LDAP users and match them up with their mappings as a preview.

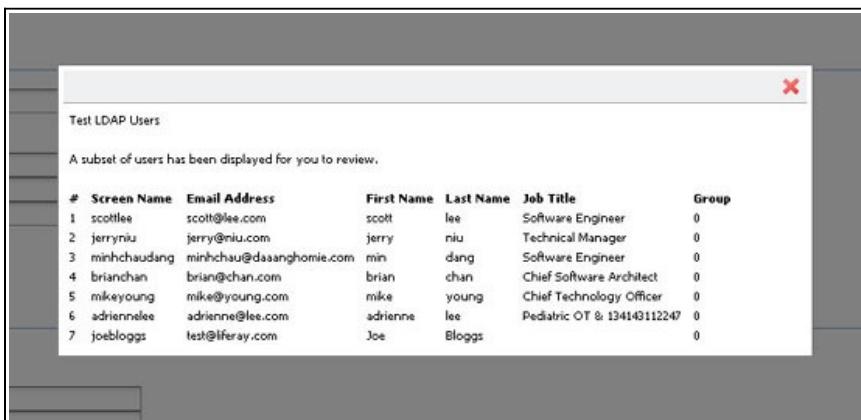


Illustration 28: Testing LDAP Users

Groups

This section contains settings for mapping LDAP groups to Liferay.

Import Search Filter: This is the filter for finding LDAP groups that you want to map to Liferay. Enter the LDAP group attributes that you want retrieved for this mapping. The following attributes can be mapped:

- Group Name
- Description
- User

Test LDAP Groups: Click the *Test LDAP Groups* to display a list of the groups returned by your search filter.

Export

Users DN: Enter the location in your LDAP tree where the users will be stored. When Liferay does an export, it will export the users to this location.

User Default Object Classes: When a user is exported, the user is created with the listed default object classes. To find out what your default object classes are, use an LDAP browser tool such as *JXplorer* to locate a user and view the Object Class attributes that are stored in LDAP for that user.

Groups DN: Enter the location in your LDAP tree where the groups will be stored. When Liferay does an export, it will export the groups to this location.

Group Default Object Classes: When a group is exported, the group is created with the listed default object classes. To find out what your default

object classes are, use an LDAP browser tool such as *Jxplorer* to locate a group and view the Object Class attributes that are stored in LDAP for that group.

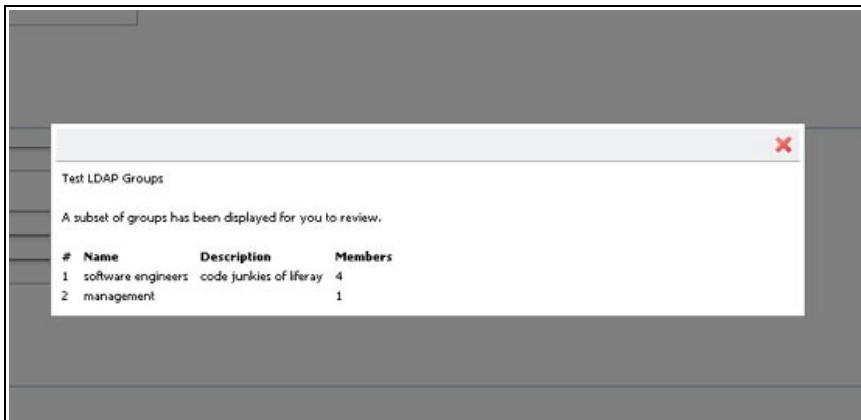


Illustration 29: Mapping LDAP Groups

Once you've set all your options and tested your connection, click *Save*. From here, you can add another LDAP server or set just a few more options that apply to all of your LDAP server connections.

Import/Export

Import Enabled: Check this box to cause Liferay to do a mass import from your LDAP directories. If you want Liferay to only synchronize users when they log in, leave this box unchecked. Definitely leave this unchecked if you are working in a clustered environment. Otherwise, all of your nodes would try to do a mass import when each of them starts up.

If you check the box, one more option becomes available.

Import on Startup Enabled: Check this box to have Liferay run the import when it starts up. This box only appears if you check *Import Enabled* above.

Export Enabled: Check this box to enable Liferay to export user accounts from the database to LDAP . Liferay uses a listener to track any changes made to the User object and will push these changes out to the LDAP server whenever the User object is updated. Note that by default on every log in, fields such as `LastLoginDate` are updated. When export is enabled, this has the effect of causing a user export every time the user logs in. You can disable this by setting the following property in your `portal-ext.properties` file:

```
users.update.last.login=false
```

Use LDAP Password Policy: Liferay uses its own password policy by default. This can be configured on the *Password Policies* link in the *Portal* section on the left side of the Control Panel . If you want to use the password policies defined by your LDAP directory, check this box. Once this is enabled, the *Password Policies* tab will display a message stating that you are not using a local password policy. You will now have to use your LDAP directory's mechanism for setting password policies. Liferay does this by parsing the messages in the LDAP controls that are returned by your LDAP server. By default, the messages in the LDAP controls that Liferay is looking for are the messages that are returned by the Fedora Directory Server. If you are using a different LDAP server, you will need to customize the messages in Liferay's *portal-ext.properties* file, as there is not yet a GUI for setting this. See below for instructions describing how to do this.

Once you have completed configuring LDAP, click the *Save* button.

LDAP Options Not Available in the GUI

Though most of the LDAP configuration can be done from the Control Panel, there are several configuration parameters that are only available by editing *portal-ext.properties*. These options will be available in the GUI in future versions of Liferay Portal, but for now they can only be configured by editing the properties file.

If you need to change any of these options, copy the LDAP section from the *portal.properties* file into your *portal-ext.properties* file. Note that since you have already configured LDAP from the GUI, any settings from the properties file that match settings already configured in the GUI will be ignored. The GUI, which stores the settings in the database, always takes precedence over the properties file.

```
ldap.auth.method=bind  
#ldap.auth.method=password-compare
```

Set either bind or password-compare for the LDAP authentication method. Bind is preferred by most vendors so that you don't have to worry about encryption strategies. Password compare does exactly what it sounds like: it reads the user's password out of LDAP, decrypts it, and compares it with the user's password in Liferay, syncing the two.

```
ldap.auth.password.encryption.algorithm=  
ldap.auth.password.encryption.algorithm.types=MD5, SHA
```

Set the password encryption to used to compare passwords if the property *ldap.auth.method* is set to *password-compare*.

```
ldap.import.method=[user,group]
```

If you set this to *user*, Liferay will import all users from the specified portion of the LDAP tree. If you set this to *group*, Liferay will search all the

groups and import the users in each group. If you have users who do not belong to any groups, they will not be imported.

```
ldap.error.password.age=age
ldap.error.password.expired=expired
ldap.error.password.history=history
ldap.error.password.not.changeable=not allowed to change
ldap.error.password.syntax=syntax
ldap.error.password.trivial=trivial
ldap.error.user.lockout=retry limit
```

These properties are a list of phrases from error messages which can possibly be returned by the LDAP server. When a user binds to LDAP , the server can return *controls* with its response of success or failure. These controls contain a message describing the error or the information that is coming back with the response. Though the controls are the same across LDAP servers, the messages can be different. The properties described here contain snippets of words from those messages, and will work with Red Hat's Fedora Directory Server. If you are not using that server, the word snippets may not work with your LDAP server. If they don't, you can replace the values of these properties with phrases from your server's error messages. This will enable Liferay to recognize them.

Single Sign-On

Single Sign-On solutions allow you to provide a single log in credential for multiple systems. This allows you to have people authenticate to the Single Sign-On product and they will be automatically logged in to Liferay and to other products as well.

Liferay at the time of this writing supports several single sign-on solutions. Of course if your product is not yet supported, you may choose to implement support for it yourself by use of the extension environment—or your organization can choose to sponsor support for it. Please contact sales@liferay.com for more information about this.

Authentication: Central Authentication Service (CAS)

CAS is an authentication system that was originally created at Yale University. It is a widely-used open source single sign-on solution, and was the first SSO product to be supported by Liferay.

Please follow the documentation for CAS to install it on your application server of choice.

Your first step will be to copy the CAS client .jar file to Liferay's library folder. On Tomcat, this is in [Tomcat Home]/webapps/ROOT/WEB-INF/lib.

Once you've done this, the CAS client will be available to Liferay the next time you start it.

The CAS Server application requires a properly configured Secure Socket Layer certificate on your server in order to work. If you wish to generate one yourself, you will need to use the keytool utility that comes with the JDK. Your first step is to generate the key. Next, you export the key into a file. Finally, you import the key into your local Java key store. For public, Internet-based production environments, you will need to either purchase a signed key from a recognized certificate authority (such as Thawte or Verisign) or have your key signed by a recognized certificate authority. For Intranets, you should have your IT department pre-configure users' browsers to accept the certificate so that they don't get warning messages about the certificate.

To generate a key, use the following command:

```
keytool -genkey -alias tomcat -keystore changeit -keyalg RSA
```

Instead of the password in the example (*changeit*), use a password that you will be able to remember. If you are not using Tomcat, you may want to use a different alias as well. For First and Last name, enter *localhost*, or the host name of your server. It cannot be an IP address.

To export the key to a file, use the following command:

```
keytool -export -alias tomcat -keystore changeit -file server.cert
```

Finally, to import the key into your Java key store, use the following command:

```
keytool -import -alias tomcat -file %FILE_NAME% -keystore changeit -keystore $JAVA_HOME/jre/lib/security/cacerts
```

If you are on a Windows system, replace `$JAVA_HOME` above with `%JAVA_HOME%`. Of course, all of this needs to be done on the system on which CAS will be running.

Once your CAS server is up and running, you can configure Liferay to use it. This is a simple matter of navigating to the *Settings -> Authentication -> CAS* tab in the Control Panel. Enable CAS authentication, and then modify the URL properties to point to your CAS server.

Enabled: Set this to true to enable CAS single sign-on.

Import from LDAP: A user may be authenticated from CAS and not yet exist in the portal. Select this to automatically import users from LDAP if they do not exist in the portal.

The rest of the settings are various URLs, with defaults included. Change `localhost` in the default values to point to your CAS server. When you are fin-

ished, click *Save*. After this, when users click the *Sign In* link, they will be directed to the CAS server to sign in to Liferay.

Authentication: Facebook

Liferay Portal also enables users to log in using their Facebook accounts. To enable this feature, you simply need to select the *Enable* box and enter the Application ID and Application Secret which should have been provided to you by Facebook. Facebook SSO works by taking the primary Facebook email address and searching for the same email address in Liferay's *User_table*. If a match is found, the user is automatically signed on (provided that user clicked *allow* from the Facebook dialog). If there isn't a match, the user is prompted in Liferay to add a user from Facebook. Once selected, a new user is created by retrieving four fields from Facebook (first name, last name, email address, and gender).

Authentication: NTLM

NTLM is a Microsoft protocol that can be used for authentication through Microsoft Internet Explorer. Though Microsoft has adopted Kerberos in modern versions of Windows server, NTLM is still used when authenticating to a workgroup. Liferay Portal now supports NTLM v2 authentication. NTLM v2 is more secure and has a stronger authentication process than NTLMv1.

Enabled: Check the box to enable NTLM authentication.

Domain Controller: Enter the IP address of your domain controller. This is the server that contains the user accounts you want to use with Liferay.

Domain: Enter the domain / workgroup name.

Authentication: OpenID

OpenID is a new single sign-on standard which is implemented by multiple vendors. The idea is that multiple vendors can implement the standard, and then users can register for an ID with the vendor they trust. The credential issued by that vendor can be used by all the web sites that support OpenID. Some high profile OpenID vendors are AOL (<http://openid.aol.com/screenname>), LiveDoor (<http://profile.livedoor.com/username>), and LiveJournal (<http://username.livejournal.com>). Please see the OpenID site (<http://www.openid.net>) for a more complete list.

The obvious main benefit of OpenID for the user is that he or she no longer has to register for a new account on every site in which he or she wants to participate. Users can register on *one* site (the OpenID provider's site) and then use those credentials to authenticate to many web sites which

support OpenID. Many web site owners often struggle to build communities because end users are reluctant to register for so many different accounts. Supporting OpenID makes it easier for site owners to build their communities because the barriers to participating (i.e., the effort it takes to register for and keep track of many accounts) are removed. All of the account information is kept with the OpenID provider, making it much easier to manage this information and keep it up to date.

Liferay Portal can act as an OpenID consumer, allowing users to automatically register and sign in with their OpenID accounts. Internally, the product uses OpenID4Java (<http://code.google.com/p/openid4java/>) to implement the feature.

OpenID is enabled by default in Liferay, but can be disabled here.

Atlassian Crowd

Atlassian Crowd is a web-based Single Sign-On product similar to CAS. Crowd can be used to manage authentication to many different web applications and directory servers.

Because Atlassian Crowd implements an OpenID producer, Liferay works and has been tested with it. Simply use the OpenID authentication feature in Liferay to log in using Crowd.

Authentication: OpenSSO

OpenSSO is an open source single sign-on solution that comes from the code base of Sun's System Access Manager product. Liferay integrates with OpenSSO, allowing you to use OpenSSO to integrate Liferay into an infrastructure that contains a multitude of different authentication schemes against different repositories of identities.

You can set up OpenSSO on the same server as Liferay or a different box. Follow the instructions at the OpenSSO site (<http://opensso.dev.java.net>) to install OpenSSO. Once you have it installed, create the Liferay administrative user in it. Users are mapped back and forth by screen names. By default, the Liferay administrative user has a screen name of *test*, so in OpenSSO, you would register the user with the ID of *test* and an email address of *test@liferay.com*. Once you have the user set up, log in to Open SSO using this user.

In the same browser window, go to the URL for your server running Liferay and log in as the same user, using the email address *test@liferay.com*. Go to the Control Panel and click *Settings -> Authentication -> OpenSSO*. Modify the three URL fields (Login URL, Logout URL, and Service URL) so that they point to your OpenSSO server (i.e., only modify the host name portion of the

URLs), click the *Enabled* check box, and then click *Save*. Liferay will then redirect users to OpenSSO when they click the *Sign In* link.

Authentication: SiteMinder

SiteMinder is a single sign-on implementation from Computer Associates. Liferay 5.2 now has built-in integration with SiteMinder. SiteMinder uses a custom HTTP header to implement its single sign-on solution.

To enable SiteMinder authentication in Liferay, check the *Enabled* box on the *SiteMinder* tab. If you are also using LDAP with Liferay, you can check the *Import from LDAP* box. If this box is checked, user authenticated from SiteMinder that do not exist in the portal will be imported from LDAP.

The last field defines the header SiteMinder is using to keep track of the user. The default value is already populated. If you have customized the field for your installation, enter the custom value here.

When you are finished, click *Save*.

Users

The *Users* link has three tabs: *Fields*, *Reserved Credentials*, and *Default User Associations*.

The *Fields* tab allows you to enable/disable the following fields:

- Enable/disable requiring Terms of Use
- Enable/disable user screen names autogeneration
- Enable/disable the birthday field
- Enable/disable the gender field

The next tab is *Reserved Credentials*. You can enter screen names and email addresses here that you don't want others to use. Liferay will then prevent users from registering with these screen names and email addresses. You might use this feature to prevent users from creating IDs that look like administrative IDs or that have reserved words in their names.

The *Default User Associations* tab has three fields allowing you to list (one per line) communities, roles, and user groups that you want new users to be members of by default. Liferay's default is to have new users be members of both the Users role and the Power Users role.

The Power Users role has the rights to use most of the portlets that ship with Liferay, so it is enabled by default. If you remove this role from the list, you may find that you need to grant users via a different role the ability to use some portlets (such as the Wiki or Message Boards).

If you have defined other user groups, communities, or roles that you want newly created users to be members of by default, enter them here. For example, you may have defined page templates in certain user groups to pre-populate end users' private pages. If there is a particular configuration that you want everyone to have, you may want to enter those user groups here.

Mail Host Names

The next link is *Mail Host Names*. You can enter (one per line) other mail host names besides the one you configured on the General tab. This lets the portal know which mail host names are owned by your organization.

Email Notifications

There are four tabs under the *Email Notifications* link. The *Sender* tab allows you to set the portal's administrative name and email address. By default, this is *Joe Bloggs* and *test@liferay.com*. You can change it to anything you want. This name and address will appear in the **From** field in all email messages sent by the portal.

The other three tabs (*Account Created Notification*, *Password Changed Notification*, and *Password Reset Notification*) allow you to customize the email messages that are sent to users on those three events. A list of tokens for inserting certain values (such as the portal URL) is given if you wish to use those.

Identification

The identification section has several links for addresses, phone numbers, and other information you can configure in your portal. This allows you to set up contact information for the organization that is running the portal. Developers can query for this information in their applications.

Miscellaneous: Display Settings

This section allows you to set the default portal language and the time zone. You can also set the portal-wide logo which appears in the top left corner of themes that are configured to display it. Be careful when using this option to choose an image file that fits the space. If you pick something that is too big, it will mess up the navigation.

Custom Fields

Custom fields are a way to add attributes to many types of assets in the portal. For example, if you are using Liferay Portal to create a site for rating books, you might add custom fields to several assets in the portal. You might give the User object a custom field called Favorite Books. If you're using the wiki for book reviews, you might add fields for Book Title and Book Author.

Configuration

To add a custom field simply click on the *Custom Fields* link in the Control Panel then click on the *Edit* link and then select *Add Custom Field*.

The screenshot shows a Liferay control panel form. On the left, there are several input fields: 'Book Author' (text), 'Book Title' (text), 'Categories' (button with a magnifying glass icon labeled 'Select'), and 'Tags' (text). On the right, there is a sidebar titled 'Other' containing code snippets: '<>TableOfContents>>' and '{{{ Preformatted }}}'. Below the code is a link 'Learn More » (Opens New Window)'. The entire interface has a clean, modern design with a light gray background and white input fields.

Illustration 30: The Wiki shows custom fields integrated with the rest of the fields on the form.

From here you will need to add the custom field key. The key appears as the label for the field on the form. For some portal assets (like the User), custom fields are a separate section of the form. For others, as can be seen above, custom fields are integrated with the default fields on the form. Additionally, developers can access custom fields programmatically through the `<liferay-ui:custom-attribute />` tag.

You can create fields of many different types: text fields (indexed or secret), integers, selection of multiple values, and more. Once you've created a field, you cannot change its type.

Monitoring

The next link on the left side of the Control Panel is for monitoring. Using this, you can see all of the live sessions in the portal. For performance reasons, this setting is generally turned off in production, but if you have it turned on, you can view the active sessions here.

Plugins Configuration

The Plugins Configuration link contains tabs for three types of plugins: portlets, themes, and layouts. You can use these tabs to view which roles can add plugins to pages or you can make the plugins active or inactive.

Note that this is for basic configuration: if you want to view the existing permission configuration for a given portlet and/or modify that configuration for existing roles, this is where you can do that. If you need to add permissions to new roles for a given portlet, use the Roles section of the Control Panel and the *Actions → Define Permissions* button.

Page Templates

The Page Templates link allows you to create a custom page with the layout, portlets and web content you will want to reuse. From this link you can also edit existing templates and modify their permissions.

The screenshot shows a web-based administration interface for managing page templates. At the top, a green header bar displays the title "Page Templates" with a help icon. Below this, a green success message box contains the text "Your request processed successfully." To the left of the message are two buttons: "View All" (with a magnifying glass icon) and "Add" (with a plus sign icon). The main content area has a title "Return to School". On the right side of the title is a "Back" link. The form fields include a "Name" input field containing "Return to School" and a dropdown menu "Other Languages (1)" with an American flag icon. A "Description" input field is present but empty. Below the description is a checked checkbox labeled "Active". Under the "Configuration" section, there is a link "Open Page Template (Opens New Window)". At the bottom of the form are two buttons: "Save" and "Cancel".

Illustration 31: You can create page templates with predefined layouts and portlets that can be used over and over.

To add a page template click on the *Page Template* link in the Control Panel. From there select *Add* and then enter a name and description for your template. Below the description field is a toggle that lets you make the template active or inactive, and below this is a link to open the page template so you can edit it. The template pictured above in a university portal might be for returning students. Clicking the *Open Page Template* link brings you to the Manage Pages interface, which we'll see in the next chapter.

Once you're finished adding the content, layout, and portlets, return to the Page Template portlet (which is in another browser tab) and select *Save*.

When you want to use the new template to create a new page on a site, simply navigate to *Control Panel* → *Pages* and the page is available in the template drop down menu when you're adding a page.

Site Templates

The Site Templates link allows you to define a set of predefined pages to include when creating new sites for communities and organizations. You may define and edit site templates as well as their permissions.

The screenshot shows a web-based application for managing site templates. At the top, there's a green header bar with the title "Site Templates". Below it is a toolbar with "View All" and "Add" buttons. The main content area has a section titled "Student Template". It contains fields for "Name" (set to "Student Template") and "Description" (containing the text "This is a template for the Student Community"). There's also a dropdown menu for "Other Languages (1)". A checked checkbox labeled "Active" is present. At the bottom, there are "Save" and "Cancel" buttons. On the right side of the main content area, there's a "«Back" link.

Illustration 32: Site templates help you to quickly build many similar sites using the same pattern.

To add a site template click on the *Site Template* link in the Control Panel. From there you can select *Add* and then enter a name and description for your template. Also, below the description field is where you can make this template active.

Once this is complete, click on *Actions* → *Manage Pages*. From here you can add pages to your template, configure the look and feel by managing themes or CSS files, and export the template as a .lar file. You can also import a .lar file from here as well.

On the pages you've created for the site template, you can add the portlets and content you want. Once you've added the pages, portlets, content, and themes, you can use this template when you create a new community. The template will appear in the drop down menus for the Public Pages/Private Pages options when you are adding a community.

Server Administration

The Server Administration link lets you perform various tasks relating to administration of the overall portal server, as opposed to administering resources in the portal. Clicking the link makes this abundantly clear: you're immediately presented with a graph showing the resources available in the JVM.

Resources

The first tab is called *Resources*. This tab contains the aforementioned graph plus several server wide actions that an administrator can execute. These are:

Garbage collection: You can send in a request to the JVM to begin the garbage collection task.

Clearing VM caches: You can send in a request to the JVM to clear a single VM cache.

Clearing caches across the cluster: You can send in a request to the JVM to clear content cached across the entire cluster.

Clearing database caches: You can send in a request to the JVM to clear the database cache.

Reindex all search indexes: You can send in a request to regenerate all search indexes. If you are not using a Solr search server (see Chapter 7 for further information). This will impact portal performance, so try not to do this except at non-peak times.

Generate Thread Dump: If you are performance testing, you can generate a thread dump which can be examined later to determine if there are any deadlocks and where they might be.

Verify database tables of all plugins: Checks all tables against their indexes for accuracy of data retrieval.

Log Levels

Here you can dynamically modify the log levels for any class hierarchy in the portal. If you have custom code that you have deployed which isn't in the list, you can use the *Add Category* tab to add it. If you change the log level near the top of the class hierarchy (such as at `com.liferay`), all the classes under that hierarchy will have their log levels changed. If you are testing something specific, it is much better to be as specific as you can when you change log levels. Modifying them too high in the hierarchy generates a lot more log messages than you need.

Properties

Liferay and the JVM contain many settings which are defined as properties. There are two tabs here: one showing system properties and one showing portal properties.

The system properties tab shows an exhaustive list of system properties for the JVM, as well as many Liferay system properties. This information can be used for debugging purposes or to check the configuration of the currently running portal.

The portal properties tab shows an exhaustive list of the portal properties. These properties can be customized, as will be seen in the next chapter. If you need to check the current value of a particular property, it can be viewed from this screen without having to shut down the portal or open any properties files.

Captcha

By default, Liferay ships with its own simple captcha service which is designed to thwart bots from registering for accounts on sites powered by Liferay. If you want to instead use Google's reCaptcha service, you can do that.

Simply check the *Enable ReCaptcha* box and enter your public and private keys into the text fields provided, and then click *Save*. Liferay Portal will then use reCaptcha instead of simple captcha.

Data Migration

If you are upgrading from a previous release of Liferay Portal or if you need to migrate your data from one system to another, this section helps you to do that without your developers having to write custom scripts.

The first section lets you copy your entire Liferay database from the current database under which it is running to the database you specify in this set of fields. You'll need to enter the driver class name (and the driver will need to be on Liferay's classpath), the JDBC URL of the database to which you'll be copying your data, and a user name and password that has access to that database. Once you have all of this information entered, click *Execute* to copy the data.

The next section helps you migrate your documents. If you want to move off of the Jackrabbit JSR-170 repository to the file system, or to the Jackrabbit repository from the file system, or to any of the other repositories supported by the document library, you can do this very easily. Make sure you have already set up your `portal-ext.properties` file so that the hook is properly configured before running this migration. Select the Document Library hook

that represents where you want your documents migrated, and click *Execute*. Your documents will be migrated to the new repository, and you can then shut down Liferay, make the new repository the default in the `portal-ext.properties` file, and then restart.

Similarly, you can migrate images from the Image Gallery in the same manner.

File Uploads

Since Liferay allows users to upload files in various places, you may want to lock down the type of files and the size of files users are allowed to upload. Here, you can set the overall maximum file size and then override that size for specific applications within Liferay. You can limit the allowed file extensions generally or by application. You have a lot of flexibility as to how you want files to be managed within your portal.

Mail

Rather than using the `portal-ext.properties` file as we did in the installation chapter, you can configure a mail server here. If the portal is to receive mail (see the Message Boards portlet in Chapter 5), you can connect a POP mail server. And of course, if the portal is to send mail—which it needs to do in order to send notifications to users—you can connect an SMTP server here as well, and this is highly recommended.

Note that if you add your mail server settings here, they will override anything that is in your `portal-ext.properties` file.

OpenOffice

Liferay Portal enables users to add content in many formats: web content, images, and files. This is done using the Web Content Management System, the Image Gallery, the Document Library, and in other portlets, both built-in and custom written. Sometimes, it is helpful to convert this content from whatever format it is in to a format that is more convenient for the user browsing the content. Liferay Portal allows users to do this by integrating with OpenOffice.org.

OpenOffice.org is an open source office suite which is normally run in graphical mode to create documents, but it can be run in “server” mode. When run in server mode, OpenOffice.org can be used to convert documents to and from all of the file types it supports. Liferay can then make use of this feature to automatically convert content on the fly.

Use this tab to tell Liferay how to connect to your running instance of OpenOffice.org. You can install OpenOffice.org on the same server upon

which Liferay is running. Once you have it installed, you can start OpenOffice.org in server mode with the following command:

```
soffice -headless -accept="socket,host=127.0.0.1,port=8100;urp;"  
-nofirststartwizard
```

As you can see, the command above specifies that OpenOffice.org will run on port 8100, which is the default port in the Control Panel . If you can use this port, all you need to do is check the *Enabled* box, and Liferay will be integrated with OpenOffice.org.

If you have something else running on this port, find a port that is open and specify it both in the command above and on the Control Panel's OpenOffice.org configuration page. When you are finished, click *Save*.

Script

Liferay includes a scripting console which lets administrators execute migration or management code instantly. Several scripting languages are supported, including JavaScript, Groovy, Python, Ruby, and Beanshell. For further information about Liferay's APIs, see the JavaDoc, the Liferay Wiki (<http://wiki.liferay.com>), or *Liferay in Action*.

Shutdown

If you ever need to shut down your Liferay Portal server while users are logged in, you can use the Shutdown tab to inform your logged-in users of the impending shutdown. You can define the number of minutes until the shutdown and a custom message that will be displayed.

Users will see your message at the top of their portal pages for the duration of time you specified. When the time expires, all portal pages will display a message saying the portal has been shut down. At this point, the server will need to be restarted to restore access.

Portal Instances

Liferay Portal allows you to run more than one portal instance on a single server. Data for each portal instance are kept separate from every other portal instance. All portal data, however, is kept in the same database.

Each portal instance requires its own domain name. Liferay will direct users to the proper portal instance based on this domain name. So before you configure an instance, configure its domain name in your network first. When you're ready to add an instance, click the *Add* button here.

You'll be prompted for three fields:

Web ID: A general convention is to use the domain name for this. It's a user-generated ID for the instance.

Virtual Host: Put the domain name you configured in your network here. When users are directed to your Liferay server via this domain name, Liferay will then be able to send them to the proper portal instance.

Mail Domain: Enter the domain name for the mail host for this instance. Liferay will use this to send email notifications from the portal.

When you are finished filling out the form, click *Save*. Now navigate to the portal using your new domain name. You will see that you are brought to what looks like a clean install of Liferay. This is your new portal instance which can now be configured any way you like.

Plugins Installation

The *Plugins Installation* link shows all of the plugins that are currently installed. These are divided into tabs for portlets, themes, layout templates, Hook plugins, and Web plugins. If you want to install a new plugin, click the *Install More Portlets* button. You will then be brought to the **Plugin Installer**, where you can browse Liferay's repository of portlets or install your own plugins. The Plugin Installer will be covered in Chapter 6.

Summary

This chapter has described the resources in Liferay Portal that can be configured to build the foundation of your web site. We have seen how to navigate Liferay's user interface so that you can get anywhere you need to in the portal. We have also looked at overall portal architecture and how you might go about designing your site using Liferay.

Next, we went in-depth through Liferay's Control Panel. Using the Control Panel, we learned how to manage users, organizations, user groups, and roles. We also learned how to configure various server settings, such as authentication, LDAP integration, and single sign-on. We also learned how to associate users by default with different user groups, communities, and roles, and we saw how to reserve screen names and e-mail addresses so that users cannot register in the portal with them.

Next, we saw how to view and configure overall server settings. We saw how to view the memory currently being used by the server, as well as how to initiate garbage collection, a thread dump, search engine re-indexing, and the clearing of various caches. We learned how to debug parts of the portal by changing log levels, and by viewing the various properties that are defined in the portal.

Finally, we learned how to properly notify users that the portal is about to shut down and how to enable the OpenOffice.org integration. The ability to run multiple portal instances on one installation of Liferay was covered, and we saw how to view the plugins that are currently installed.

All of this information was designed to put you on the path to becoming a seasoned Liferay Portal Administrator.

4. WEB CONTENT MANAGEMENT

With most products, you would read the previous chapter, understand what the software can do in terms of setting up your environment and security model, and go ahead and build your system. You'd design your infrastructure and get your server environment up and running while your developers would go ahead and write the applications that would live on your web site. Liferay Portal, however, doesn't leave you to start that far behind. Liferay Portal is more than just a *container* for applications with a robust security model. It already includes many of the applications you'll need, out of the box, ready to go, and integrated with all the user management and security features you've already learned about.

Perhaps the key application that ships with Liferay is Liferay's Web Content Management system (CMS). We call it the key application because it is the one that most everybody uses, because everyone needs to load content onto their web sites. Liferay's CMS empowers you to manage all of the content on your site quickly, easily, and in the browser. Beyond managing existing content, Liferay CMS lets users easily create and manage everything from a simple article of text and images to full functional web sites. Web publishing works alongside Liferay Portal's larger collection of applications, which means that you can add in shopping cart functionality, visitor polls, web forms, community collaboration tools and more. Everything is done with our collection of easy-to-use tools with familiar rich-text editors and intuitive interface.

This chapter covers all aspects of Liferay CMS, including:

- Page types

- Layouts
- Page and content permissions
- Importing and exporting content
- Content creation and editing
- Staging
- Content publishing
- Structures and templates
- CMS Workflow
- Asset publisher

By the time we're done, you should be able to apply all of these concepts to your own content. To demonstrate all of Liferay's Content Management features, we'll create and manage content on the portal for a fictitious company called *Spartan Software*.

First, a little housekeeping. If we're going to be Spartan Software, our portal should also be called Spartan Software. To set the general information about your portal like the name and mail domain, go to *Control Panel* → *Portal* → *Settings*. The configuration for Spartan Software might look something like the below screenshot.



Illustration 33: Changing Portal Settings

You can also customize the logo in the top left corner of every page by selecting *Display Settings* under the *Miscellaneous* tab on the panel to the right. Once you've made the changes, we're ready to begin creating pages.

Page Creation and Management

Your pages must contain something, right? The reason web pages exist at all is to display content. Whether you're using the web to share project information, advertise to potential customers, or demonstrate a golf swing—

you're displaying content. And with Liferay's CMS, you'll find that the process of loading your pages up with content is extremely easy and straightforward, and you don't ever need to leave your browser.

Essentially, CMS is a suite of various portlets that include functionality for creating and displaying various types of content. Don't be concerned by the sheer number of applications you see in this category in the *Add → More* menu: we'll go over everything in a step-by-step fashion. We'll start by taking a look at page creation and management.

Managing Pages

There are three ways to access the Manage Pages interface. If you are currently on the set of pages you wish to manage, simply go up to the Dockbar and select *Manage*, then *Page*. You can also access the Manage Pages tool two other ways in the Control Panel.



Illustration 34: Two paths to managing pages

The first way is to simply select the community or organization that you want to work with in the Content section, and click on *Pages* in the left column. The second way is to select *Actions* and then select *Manage Pages* from the community or organization you're working with down in the portal section of the Control Panel.

If you only need to add a single page very quickly, you can simply go to *Add → Page* in the Dockbar, and a page will be added immediately. You can then click on the page in the navigation and begin working on it immediately.

Once you're in Manage Pages, you'll see an interface to create new pages, change various settings related to the set of pages you've selected, and export or import pages using Liferay Archive (LAR) files. We're just going to stay on the first tab under the Public Pages tab for now, so we can focus on the set of pages available to anyone visiting the site. By default, Liferay contains a single page called *Welcome*, and we'll leave it that way.

Understanding Public and Private Pages

Liferay's page groups are always associated with Users, Organizations, or Communities. And even under the hood, a user's personal pages are part of a private community just for that user.

All sets of pages are divided into two groups: Public Pages and Private Pages. Public Pages, by default, are accessible to anyone—including those who haven't logged in to the portal.

Private Pages, by default, are only accessible to the users who belong to the Organization or Community to which the pages belong. This means that an Organization's private pages would only be viewable by members of the Organization. The tabs for public and private pages have the same interface.

Manage Pages Interface

As you can see, the screen defaults to the New Page tab. Because the name of the community is selected on the left, adding a page here will add a top level page next to the Welcome page. But you can nest pages as deeply as you wish. If you wanted to create a sub-page under the Welcome page, all you'd have to do is select the Welcome page first and then create your page.

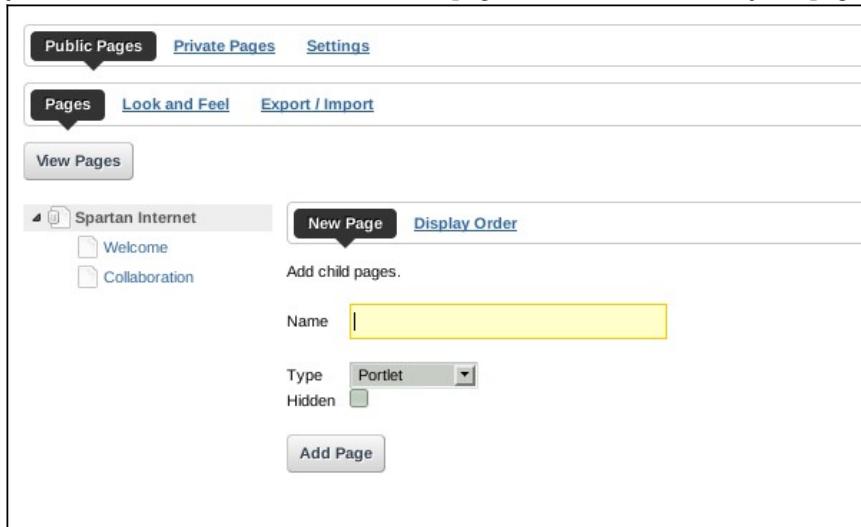


Illustration 35: Adding pages to a community is easy to do.

And if you later decide you don't like the order of your pages, you can drag and drop them in the list to put them in whatever order you want. Let's go ahead and add another top level page and name it *Collaboration*. We'll use this page for portlets such as Blogs and Wiki.

When you're finished adding your pages, you can get back to them by clicking the *View Pages* button just above the page hierarchy. Let's not do that just yet; we've got a few more options to cover.

Look and Feel

If you click the second tab, you'll be presented with an interface which allows you to choose a theme for the site upon which you are currently working. Themes can transform the entire look of the portal. Themes can be created by developers and can be installed very easily by using the Plugin Installer from the Control Panel, and this is detailed in Chapter 6.

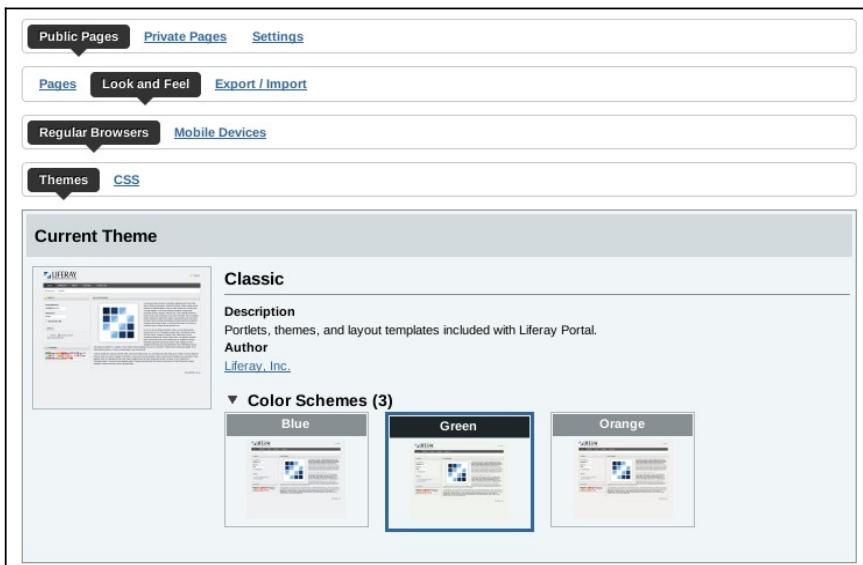


Illustration 36: Many themes include more than one color scheme. This allows you to keep the existing look and feel, but give your site a bit of a different flavor than the default.

Since we don't have any themes beyond the default one installed yet, we'll use the default theme for our pages. Some themes have more than one color scheme which can be selected, including the default theme. Let's change the color scheme by selecting *Themes* and then clicking on the *Green* color scheme. If you then go back to the site (by clicking *Back to Spartan Internet* in the top left corner of the Control Panel), you'll see that some parts of the theme are now tinged in a greenish hue.

Also, notice that you can choose themes for regular browsers or mobile devices. You might create another community for mobile users attached to the <http://m.spartansoftware.com> address and serve up a page that is designed for the smaller screens on phones.

The CSS tab allows you to enter custom CSS that will also be served up by your theme. In this way, you can tweak a theme in real time by adding new styles or overriding existing ones.

Let's move on to the final tab under *Pages*.

Export / Import

The Export / Import tab allows you to export the pages you create into a single file, called a LAR (Liferay Archive). You can then import this file into any server running Liferay, and all of your pages will be copied. This is a great way to take your content from one environment (say, a development or QA environment) and move it all in one shot to your production server.

This is also a good way to back up your site's content. You can export it to a specific location on your server which is backed up, and if you ever have to restore your site, all you need to do is import the latest LAR file. One limitation on LAR files, however, is that they are version dependent, so you can't use an export from an old version of Liferay and import it into a newer version.

Let's be good admins and export a LAR file for backup purposes. Select the *Export / Import* tab and then name the file spartansoftwarev1.lar. We're asked what we would like to export. Let's select everything for this initial export. Note that if you select the *More Options* link, the list expands to include data from many of Liferay's applications, including the Document Library and the Wiki. You can also export the theme you're using.

Once you click the *Export* button, your browser will prompt you to save the file. Once you have the file, you can copy it to a backup location for safe keeping or import it into another installation of Liferay Portal. If you must rebuild or wish to revert back to this version of your site, you can import this file by selecting *Export / Import* and then selecting *Import* and then browsing to the file.

Settings

The Settings tab gives us several powerful tools. In this tab you'll find options to customize the logo, stage the content, connect a domain to a community or organization, take advantage of the Liferay's integration with Google Analytics, and more. Let's take a look at them in order.

Virtual Host

You can make web navigation much easier for your users by connecting a domain name to a community or organization. This tab allows you to define a domain name (i.e., www.mycompany.com) for your community. This can be a full domain or a subdomain. This enables you to host a number of web sites as separate communities on one Liferay server.

For instance, if we set this up for Spartan Software's marketing community, users in that community could use marketing.spartansoftware.com to get to their community, provided that Spartan's network administrators created the domain name and pointed it to the Liferay server.

To set this up, you would simply configure the DNS for marketing.spartansoftware.com at the web address for your portal, and enter <http://marketing.spartansoftware.com> in the Virtual Host tab for the marketing community.

This can help users quickly access their community without recalling an extended URL.

Logo

If you want to use your own logo for a specific community or organization, you can configure an alternate logo here. To add a custom logo, select the Logo tab, then browse to the location of your file and select the *Use Logo* box. Be careful to make sure that your logo fits the space in the top left corner of the theme you are using for your web site.

Sitemap

The next tab lets you generate a sitemap to help you optimize your site for search engines. The Sitemap tab publishes your site using the sitemap protocol, a protocol that helps search engines crawl your web site to index all relevant information. You can publish your site to Yahoo or Google, and their search indexes will use the sitemap to index your site.

Liferay Portal makes this very simple for administrators by generating the sitemap XML automatically for all public web sites.

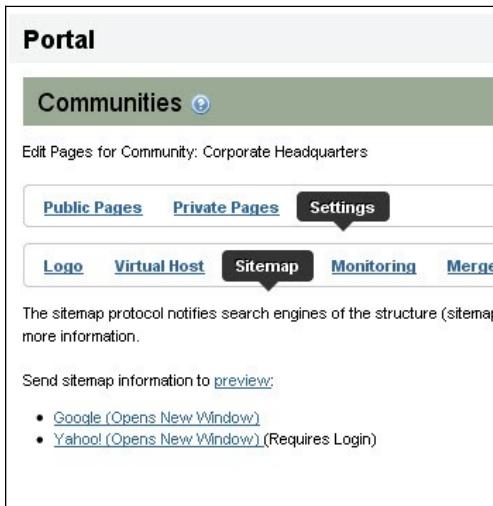


Illustration 37: Site Map Options

By selecting one of the search engine links, the sitemap will be sent to them. It's only necessary to do this once per community or organization. The search engine crawler will periodically crawl the sitemap once you've made the initial request.

If you're interested in seeing what is being sent to the search engines, selecting the *Preview* link allows you to see the generated XML.

Monitoring

The Monitoring tab allows you to integrate your pages with Google Analytics. Liferay provides seamless integration with Google Analytics, allowing you to place your ID in one place, and then it will get inserted automatically on every page. This enables you to focus your efforts on building the page, rather than remembering to put the code everywhere. Google Analytics is a fantastic, free service which lets you do all kinds of traffic analysis on your site, so you can see who visits, where they visit from, and what pages they most often visit. This helps you to tweak your site so that you can provide the most relevant content to your users.

Staging

Staging is a major feature of Liferay CMS. The concept of staging is a simple one: you don't want your users seeing your web site change before their eyes as you are modifying it, do you? Liferay's staging environment al-

lows you to make changes to your site in a specialized *staging area*, and then when you are finished, publish the whole site to your users.

You can use staging in multiple ways. Larger organizations may consider having a staging server—a separate instance of Liferay Portal which is used just for staging. Content creators can then use this server to make their changes while the live server handles the incoming user traffic. When changes to the site are ready to be published, they can be pushed remotely to the live server.

Alternatively, you may want to host both your staging environment and your live environment on the same server—particularly if you are part of a smaller organization with less resources to purchase servers. Either way, once set up, the interface is the same; the only difference comes when it's actually time to publish your content.

Enabling the Staging Environment

The Staging tab allows us to make changes in a staging environment and preview our work before publishing it to the live site. Let's create a staging environment for the Spartan Software's *Corporate Headquarters* community.

First, we'll need to create the community. Go the *Control Panel*, select the *Communities* tab, and then select *Add*. In the **Name** field, type in *Corporate Headquarters*. Add a description and leave the type set to *Open*. After you've completed these steps, select *Save*.

The screenshot shows the 'Communities' section of the Liferay Portal. At the top, there's a green header bar with the title 'Communities'. Below it, there are two buttons: 'View All' and 'Add'. The 'Add' button is highlighted with a yellow background. The main form area starts with a 'Name' field containing 'Corporate Headquarters', which is also highlighted with a yellow background. A 'Description' field follows, containing the text 'This community is meant for those based at Corporate Headquarters'. Under 'Type', a dropdown menu is set to 'Open'. A checked checkbox labeled 'Active' is present. A 'Categories' section includes a 'Select' button. The 'Tags' section has a text input field, an 'Add' button, a 'Select' button, and a 'Suggestions' button. At the bottom of the form are 'Save' and 'Cancel' buttons.

Illustration 38: Setting up an Open Community

Now we're going to add a page to our new community. Since you're already in the *Communities* interface, you can simply select *Actions* → *Manage Pages* for the Corporate Headquarters community.

Add a new public page named *News and Events*. Next, click the *View Pages* button and add the Alerts and Announcements portlets to it.

Now we're ready to activate the staging feature for this community. You should now have two tabs or windows open in your browser to Liferay: one is on the Control Panel, and one contains the page you have just created. Return to the Control Panel tab where you're editing the page structure and select *Settings* → *Staging*.

Staged Portlets

⚠️ When a portlet is checked, its data will be copied to staging and it may not be possible to edit them directly in live. When unchecking a portlet make sure that any changes done in staging are published first, because otherwise they might be lost.

Portlets that are checkmarked will be *Staged*. This means that their data is published automatically whenever a page containing them is published. Those portlets which are disabled and checkmarked are always automatically exported even if they are not on the page. Those which are disabled and not checkmarked are never automatically published. Collaboration portlets, such as *Blogs*, *Message Boards* and *Wiki* are excluded from being *Staged* by default as their data typically originates in *Live*. Effectively, when a portlet is *NOT Staged* the *Live* environment contains the only important data. When *Local Live* staging is used, the same data will be displayed whether viewing the portlet from either environment.

Blogs
 Bookmarks
 Calendar
 Document Library
 Document Library Display
 Image Gallery
 Message Boards
 Page Comments
 Page Ratings
 Polls
 Polls Display
 Web Content
 Web Content Display
 Wiki
 Wiki Display

Advanced Options

Number of Editorial Stages

Illustration 39: You can choose what content should be staged and what content should not be staged.

We'll assume we don't have a separate staging server, and so we'll select a staging type of *Local Live*. If you were to select *Remote Live*, you would also have needed to supply the name or IP of the remote server where staged content should be published, the port (80 if Liferay is sitting behind a web server, or the port your application server is listening on if not), and the remote community or organization ID. You can find this ID by selecting *Actions → Edit* on any community or organization in the Control Panel. Either way, once we make a selection (*Local Live* or *Remote Live*), many more options will become available to us.

We'll cover many of the collaboration portlets listed here when we come to chapter 5. For now you just need to be aware that the option is available to enable or disable staging for any of them, and you need to decide if you would like to stage content for these portlets. In the case of the collaborative portlets, the answer would of course be "no." Why? Because portlets such as the *Message Boards* are designed for user interaction. If their content were

staged, you'd have to manually publish your site whenever somebody posted a message on the message boards in order to make that message appear on the live site.

Generally, you'll want Web Content to be staged, because end users aren't creating that kind of content—that's the stuff you publish to your site. But portlets like the message boards or the wiki would likely benefit from *not* being staged.

Under *Advanced Options*, you can select a number of Editorial Stages your content needs to go through. This means you can have your pages reviewed through up to six levels of approval before they get published. If you select a number of stages here, you'll next be asked to define roles for each stage. Users in these roles will need to approve the changes through however many stages you have selected in order for the changes to be published live. Let's see how all of this works.

Using the Staging Environment

If we navigate back to the News and Events page of the Corporate Headquarters community we'll now notice an orange border around the page, and the *Staging* link will be in the Dockbar. Select the *Staging* button and then *View Staged Area*. The page now has a red border.



Illustration 40: The live site has an orange border; the staged site has a red border.

Add the wiki portlet and then from the Dockbar select *Staging* → *View Live Page*. Notice that the wiki portlet isn't there. As you can see, only the page changes are staged; web content itself can go through a whole different workflow process (which we'll get to later). If you select *Staging* → *View Staged Page*, you'll also see that since we didn't configure the Wiki portlet to stage its content (because that content is user-submitted), the Wiki portlet in our staged area displays a warning message telling us that its content is not staged.

Now, let's say that our page is exactly the way we want it, and we want to publish it to the live site. If we have enabled more than one Editorial Stage, we can submit a proposal to have the content reviewed. If not, we can simply publish it to the live site.

To submit a proposal, you must be viewing the staged page. From there, you can go up to the Dockbar and select *Staging → Propose Publication*. Once you do this, a small dialog box will pop up asking you for the proposal description as well as allowing you to select the reviewer who should be able to review this proposal. The list of reviewers is populated from the roles you selected when you enabled Editorial Stages.

Once you've submitted your proposal, it will no longer be possible to submit more proposals. Your proposal will have to be dealt with by someone with the role who reviews the content at this editorial stage. To view the proposals, go back up to the Dockbar and select *Staging → View Proposals*. You'll then see the proposal you've submitted.

Name	Type	ID	User	Due Date	Status
News and Events	Page	10429	Richard Sezov	Never	Stage 2 Pending Review

Illustration 41: Proposals must be approved or deleted.

Click on the proposal to view it. Here, you'll be able to change the approver, set a due date, select the next reviewer (from the role which is configured for that editorial stage), and, of course, the other options you'd expect: preview the proposed changes, approve, and reject.

Once rejected, the proposal needs to either be approved later or deleted. If the proposal is approved, it goes on to the next stage, if there's another approver. Otherwise, a *Publish to Live* button is enabled. In order to get the modified pages onto the live site, somebody has to push this button; there isn't a way to cause the pages to be published automatically upon the last approval.

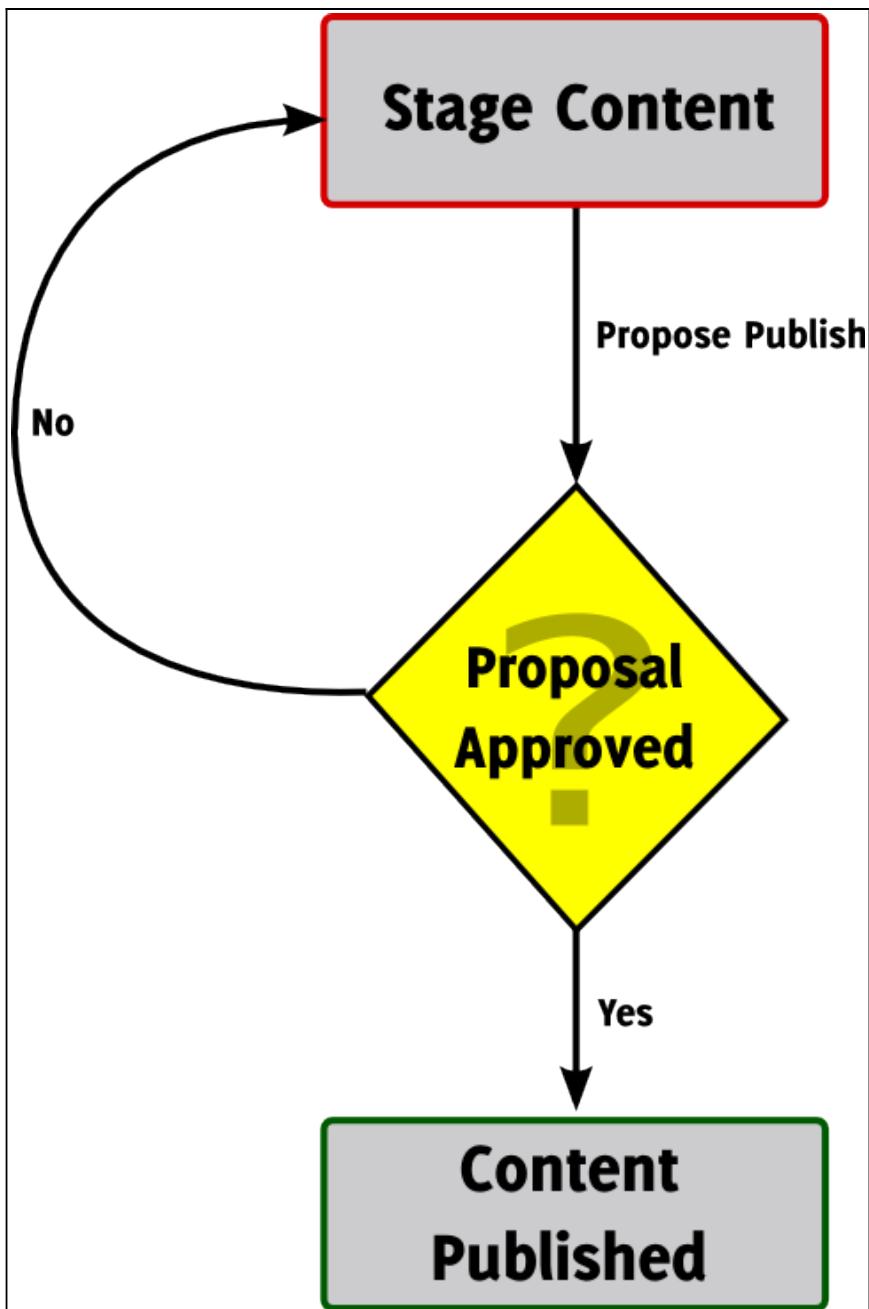


Illustration 42: There's a specific workflow to use when publishing content using the staging environment.

To publish the modified page we simply need to return to the proposal and click *Publish to Live*. Once this is done, Liferay will ask you what you wish to publish. In our example, we have only one page to publish, and it's already selected. All we have to do is click *Publish*. The changes are now live!

When it comes time to modify your site again, you simply repeat the process. You can enable staging on an individual community or organization basis, depending on your needs. This makes it really easy to put strict controls in place for your public web site, but to open things up for individual communities that don't need such strict controls. Liferay's staging environment is extremely easy to use and makes maintaining a content-rich web site a snap.

Editing a Page

There are a lot of other things you can do beyond placing portlets on a page. So let's move on from staging back to the Spartan Software community, which is not staged. You can do this by going up to the Dockbar and clicking *Go To → Spartan Internet*.

We'll use the *Collaboration* page you created earlier in the chapter. Navigate to the *Collaboration* page and select *Manage → Page* from the Dockbar.

This screen should now be familiar to you, but we haven't yet looked at all of its options.

The Page tab allows you to:

- change the name of the page
- enter HTML code for the title
- choose the page type
- hide the page from the theme navigation
- define a friendly URL to the page
- choose an icon to be displayed
- choose a frame target for the page
- copy an existing page

You can also enter custom meta tags or JavaScript to the page if you're a web developer. Additionally, if you click the *Permissions* button, you can define which users, groups, roles, or organizations can view or edit the page.

The Children tab lets you create child pages underneath the page you've selected. You can nest pages as deep as you like, but for every page below the top level hierarchy, you must provide navigation to it via a Navigation or

Breadcrumb portlet, at least with most themes (including the default). Developers can create themes which have cascading menu bars which show the full hierarchy, and some examples of that are in Liferay's plugin repositories.

For now, let's just stick with the Page tab. Change the title to *Collaborate with Spartan!*, click *Save*, and then click *Return to full page*.

Modifying Page Layouts

Page layouts allow you to arrange your pages so that the content appears the way you want it to. Liferay comes shipped with many layouts already defined. Developers can create more and they can be deployed to your portal for your use.

In order to prepare for the portlets we'll soon be adding, let's change the layout of the Collaboration page. To access layouts, go up to the Dockbar and select *Manage → Page Layout*.

Now, select the *2 Columns (50/50)* layout and then *Save*. Once saved, you'll return to the page and it'll seem as though nothing has happened. However, once we start adding portlets you'll notice how the page is now equally divided into two columns. You can stack portlets on top of each other in these columns. There are, of course, more complicated layouts available, and you can play around with them to get the layout that you want.

Sometimes a particular layout is *almost* what you want, but not quite. In this case, you can use the Nested Portlets portlet to embed a layout inside another layout. This portlet is simply a container for other portlets. Its configuration allows you to select from any of the layouts installed in Liferay, just like the layouts for a page. This gives you virtually unlimited options for laying out your pages.

Portlets

As we discussed earlier, Liferay Portal pages are composed of portlets. All of your site's functionality, from blogs to shopping, is composed of portlets.

Adding portlets to a page is simple. Let's add some to our Collaboration page.

1. In the Dockbar, select *Add → More*.
2. In the window that appears, expand the *Collaboration* category.
3. Drag the *Blogs* portlet off the Add Application window onto the *right column* of our page.
4. Next, drag the *Wiki* portlet to the *left column*.

See how easy it is to add applications to your pages? We've gone one step further: we've got the Wiki portlet, the Blogs portlet, and then a nested portlet with a different layout and the Alerts, Unit Converter, and RSS portlets.

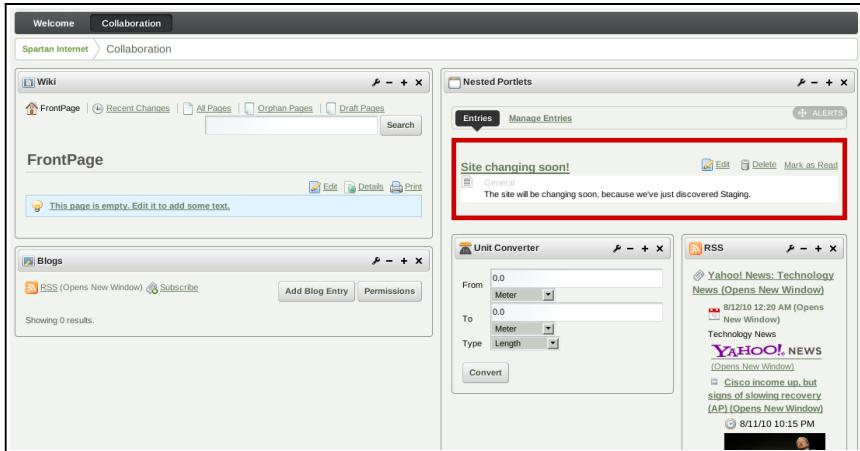


Illustration 43: Yeah, we're showoffs. But as you can see, your page layout options are virtually limitless.

You'll find it is very easy to make your pages look exactly the way you want them to. If the layout options provided aren't enough, you can even develop your own. More information about that can be found in Liferay's official guide to development, *Liferay in Action*.

Page Permissions

By default, public pages are just that: public. They can be viewed by anybody, logged in or not logged in. And private pages are really only private from non-members of the community. If someone has joined your community or is a member of your organization, that person can see all of the private pages. You can, however, modify the permissions on individual pages in either page group so that only certain users can view them.

So, let's say we wanted to create a page only for administrators to see. We can do this with the following procedure:

1. Go to the Dockbar and select *Manage → Control Panel*.
2. Click the *Pages* link in the left navigation.
3. Click the *Private Pages* tab to switch to the Private Pages. Remember, these pages by default are viewable only by members of the community or organization.
4. Create a page called *Admin Forums*.

5. Click on the page in the tree on the left and then select the *Page* tab.
6. Click the *Permissions* button.
7. Uncheck the *View* permission next to the Guest role.
8. Uncheck the *View* and *Add Discussion* permissions next to the Community Member role.
9. Click the *Save* button.

Role	Add Discussion	Delete	Delete Discussion	Update	Update Discussion	View
Guest	<input type="checkbox"/>	<input checked="" type="checkbox"/>				
Owner	<input checked="" type="checkbox"/>					
Power User	<input type="checkbox"/>					
User	<input type="checkbox"/>					
Community Member	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Illustration 44: Permissions for Admin Forums

Congratulations! You've just changed the permissions for this page so that only community administrators can view it. Any users you add to this role will now be able to see the page. Other users, even members of this community, won't have the permissions to see it.

You now understand how to manage pages in Liferay Portal. It's time to move on to adding content to those pages. Liferay's Web Content Management (WCM) is a highly powerful, yet flexible, set of tools that enables you to successfully manage your web site.

You'll soon discover that Liferay's WCM is easy to learn and highly configurable. If you already have experience with WCM, you'll see some new features and improvements to old ones. If you're new to Liferay's WCM, then you'll be surprised at how fast you will be adding, editing, and scheduling content on your site. Once you're familiar with portlets such as Web Content Display and Asset Publisher, your ability to manage an immense site with a large amount of content will simply amaze you.

We'll be using Liferay's WCM to publish simple pieces of content, develop templates to define how content is to be displayed, set up a workflow for content to be approved, schedule when content is to be published and much, much more.

What is Web Content Management?

Web Content Management is a system which allows non-technical users to publish content to the web without having advanced knowledge of web technology or programming of any sort. Liferay CMS empowers you to publish your content with a simple point and click interface, and it helps you to keep your site fresh. You'll find yourself easily creating, editing, and publishing content within just a few minutes of being exposed to its features. But Liferay CMS doesn't sacrifice power for simplicity. If need be, you can use your developer skills to create complex presentation layer templates that make your content "pop" with dynamic elements. Once these templates have been deployed into the portal, your non-technical users can manage content using these templates as easily as they would manage static content. All of this makes Liferay CMS an appropriate choice for sites with only a few pages or sites with gigabytes of content.

How Can Liferay's WCM Help You?

With Liferay's WCM you have the ability to create, edit, stage, publish, and approve content with easy to learn yet powerful tools. Liferay's WCM streamlines site changes for the end user versus doing a site in HTML. Some ways Liferay WCM makes this possible include:

- Once set up, non-technical users can manage the site.
- Liferay's fine-grained permissions system ensures your content gets to the right users.
- To manage the site, no programming is required.
- Content can be staged.
- Content can be passed through a workflow.
- Content can be published on a schedule.
- WCM is integrated with Liferay's services, so advanced template developers can use them to query for data stored elsewhere in Liferay.

What Features Does Liferay WCM Have?

Liferay's WCM has a host of features the makes managing the content of your site easier.

- **WYSIWYG Editor:** A complete HTML editor that allow you to modify fonts, add color, insert images and much more.
- **Structure Editor:** Easily add and remove fields you want available to content creators and then dynamically move them

around. This editor includes an entire suite of form controls you can drag and drop onto your structure.

- **Template Editor:** Import template script files that inform the system how to display the content within the fields determined by the structure.
- **Web Content Display:** A portlet that allows you place web content on a page in your portal.
- **Asset Publisher:** A portlet which can aggregate different types of content together in one view.
- **Scheduler:** Lets you schedule when content is reviewed, displayed, and removed.
- **Workflow Integration:** Run your content through an approval or review process.

Liferay's Web Content Management is a powerful and robust tool for creating and organizing content on your web site. Now that you've seen the basics of what you can do with Liferay's WCM, let's apply some of these concepts and create some content.

Building a Site with Liferay's WCM

You've just been assigned the task to build the web site for a small company that makes video games named Spartan Software. You've decided to take advantage of Liferay Portal and its rapid deployment features as well as its ability to get a fully functional, content-rich web site up and running in little time.

We'll walk through the creation of Spartan Software's web site, starting by creating some simple content using Liferay's built-in WYSIWYG editor and then publishing it. We'll then take advantage of Liferay's robust structure editor. We'll use templates to display the content and explore some of the advanced publishing features such as the built-in workflow and Asset Publisher.

Simple Content Creation

As we've stated above, content is the reason web sites exist. Liferay Portal has made it easier than ever to get content published to your site. Because Liferay Portal is so flexible, you can use basic authoring tools right away or take advantage of the more advanced features. It's adaptable to your needs.

We'll begin by creating simple content using Liferay's WYSIWYG Editor and then we'll publish it to the home page of Spartan Software's web site. This a fast and straightforward process that demonstrates how easy it is to

create and publish content onto your Liferay Portal. So let's get familiar with the Web Content section of the Control Panel so we can create and publish our first pieces of content.

Web Content Section of the Control Panel

When you manage web content from the control panel, you have the ability to select the location where you will be adding or editing the content. For instance, you can add content that will be available to a specific community, organization, or globally across the portal. The Content section of the Control Panel displays the name of the community or organization where you are currently working as its heading. You can change where you're working using the drop down attached to the heading.

We will add our first piece of content to the *Spartan Internet* community, which we defined earlier in the chapter as the default community.

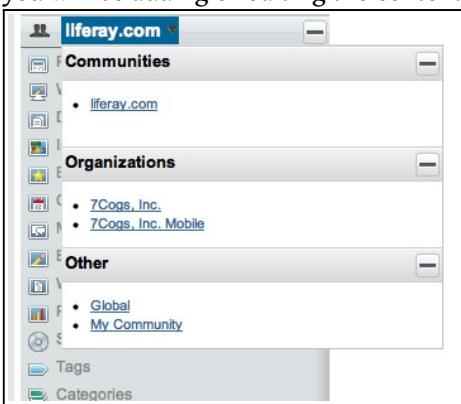


Illustration 45: Choosing an organization or community in the Content section

Rich, WYSIWYG Editing

Once you have the Spartan Internet community selected, click on the *Web Content* link in the control panel. Next, click the *Add Web Content* button. This is a highly customizable form that is asking for a title and it contains a powerful WYSIWYG editor. From here we can add images and links, change fonts, italicize, increase font size and more. We will cover the other features such as structures, templates, and content scheduling later in this chapter.

For now, type the words *Welcome to Spartan* in the **Name** field. Notice that content can be localized in whatever language you want. If you click the *localize* check box, two select boxes appear which allow you to pick the language you're working in and the language that is the default. You can enter translations of your content for any language in the list. The screenshot below shows this interface but for now, we won't be using it, so you can leave it un-checked. In the content field, add a short sentence announcing that the web site is up and running.

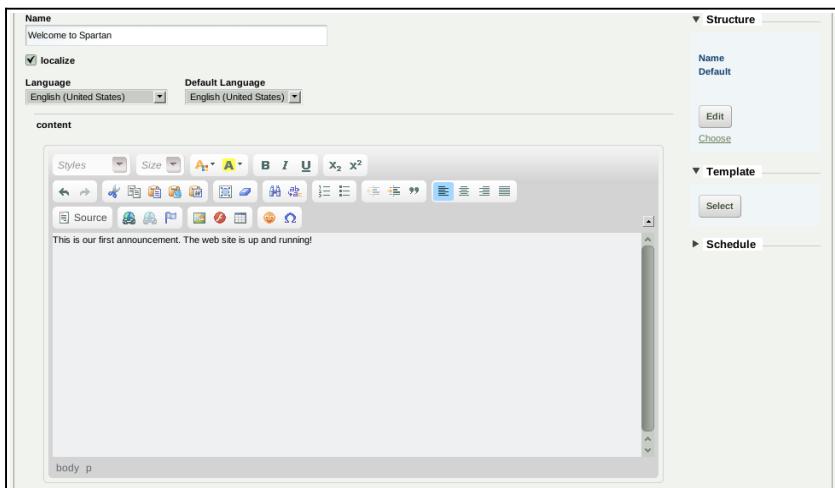


Illustration 46: Web Content Editor

Images, Fonts, Links, and More

Getting a new web site up and running is an exciting step for any enterprise, whether it is a large corporation or a small non-profit charity. To celebrate this momentous achievement at Spartan Software, let's give our announcement some of the pomp and circumstance we think it deserves!

Using the editor, select all of the current text, and then change the style to Heading 1, the font size to 20, and the color to medium blue.

A couple of lines lower, return the style, font, and color to normal or default. Type in another sentence. As you can see, you can adjust the style, size, and color to match the needs of the content. You can also bullets, numbering, links to another site, or custom images. You can even add an emoticon. Let's add a smiley face at the end of our announcement.

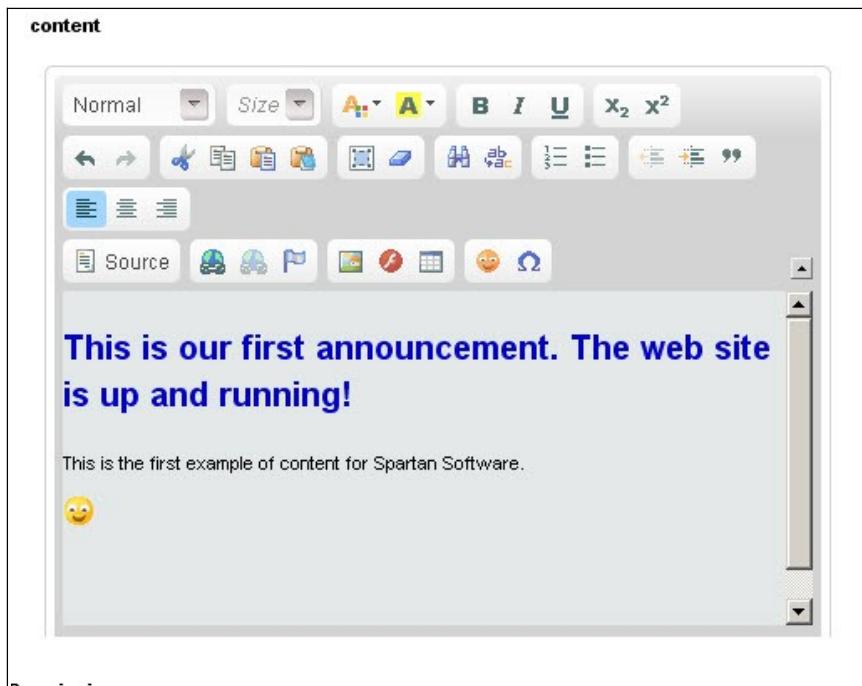


Illustration 47: Customizing Content

The WYSIWYG editor is a flexible tool that gives you the ability to add text, images, tables, links, and more. Additionally, you can modify the display to match the purpose of the content. Plus it's integrated with the rest of Liferay Portal: for example, when you upload an image to be added to a page, that image can be viewed and manipulated in the Image Gallery portlet.

If you're HTML savvy, Liferay CMS doesn't leave you out in the cold. You can click the *Source* button and write your own HTML if you wish.

Assigning Permissions

You can determine who will and who won't see your content. By default, the content is viewable by Anyone (Guest Role). You can limit viewable permissions by selecting any Role from the drop-down or in the list. Additionally, Liferay Portal provides the ability to customize permissions in more detail. Select the *More Options* link next to the drop down button, and you'll find the different activities you can grant or deny to your web content.

Permissions							
Roles	Add Discussion	Delete	Delete Discussion	Expire	Permissions	Update	Update Discussion
Guest	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Power	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
User	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Illustration 48: Permissions for Web Content

For this piece of web content, we don't need to change anything. After you're finished with permissions, click *Save*. This will save the content in draft form. Once you're satisfied with the your changes, select *Publish*. This makes the content available for display, but we still have some work to do to enable users to see it. In Liferay CMS, all content resides in a container, which is one of two portlets: *Web Content Display* or *Web Content List*. By far the most frequently used is the *Web Content Display* portlet. So let's go back to the page where we want the content displayed and add the *Web Content Display* portlet (in this case, the Welcome page).

Publishing Content with the Web Content Display Portlet

Now that we've created and published our first piece of web content for Spartan Software, it's time to display it. First, we'll need to add the *Web Content Display* portlet to our Welcome page. Do this by selecting *Add → Web Content Display* from the Dockbar.

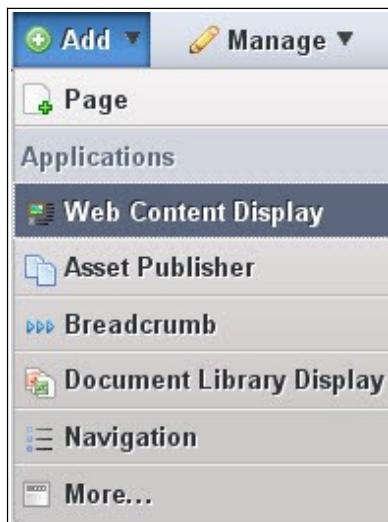


Illustration 49: Adding the Web Content Display Portlet

Once the portlet appears, drag it to the position on the page where you want your content to appear. You can use the Web Content Display portlet to lay out the content on your pages any way you like. You can have as many Web Content Display portlets on a page as you need.

Publishing Existing Content

To add existing web content, select the *gear* icon on the lower left of the portlet. You will see the message “Please select a web content from the list below.” You have several options here.

Naturally, if your content appears in the list, you can simply select it. If there were lots of published content available, you could search for the content by name, ID, type, version, content, and community (click the *Advanced* link to see all the options). You can also show the available locales for your content. If you're working on the page for a particular language, you can select the translation of your content that goes with your locale.

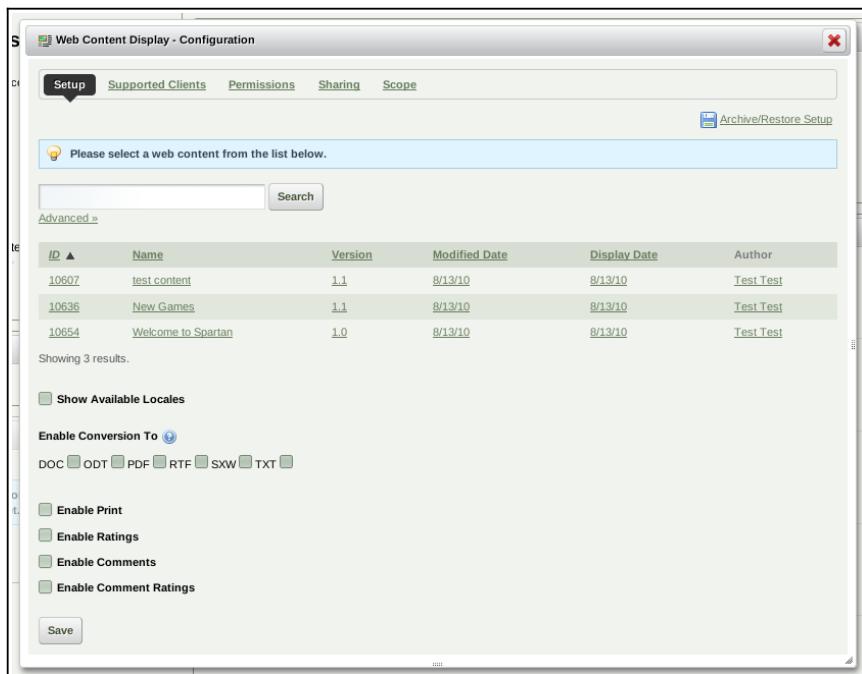


Illustration 50: Publishing web content is a snap. At a minimum, you only have to select the content you wish to publish. You can also enable lots of optional features to let your users interact with your content.

If you have enabled OpenOffice.org integration with your portal, you can also enable the document conversion. This gives your users the ability to download your content in their format of choice. This is especially handy if you are running a research or academically oriented site; users can very quickly download PDFs of your content for their research projects.

Note that you also have other options, such as enabling a Print button, enabling ratings so that users can rate the content, enabling comments, and enabling ratings on comments.

The Print button pops the content up in a separate browser window that contains just the content, without any of the web site navigation. This is handy for printing the content. Enabling ratings shows one of two ratings interfaces Liferay has: five stars or thumbs up and thumbs down. This can be set globally in the portal-ext.properties file. See Chapter 6 for further information about this.

Enabling comments creates a discussion forum attached to your content which users can use to discuss your content. Enabling ratings on comments gives your users the ability to rate the comments. You may decide you want

one, some, or none of these features, which is why they're all implemented as simple check boxes to be enabled or disabled at need.

If you click the *Supported Clients* tab, you'll see that you can choose the type of client to which you want to expose content. This lets you target the large screens of users' computers for expansive graphics and lots of special effects, as well as target the small screens of mobile devices with pertinent information and a lightweight page. For now, leave both checked and click the *Save* button. You can now close the configuration window.

Publishing New Content

To publish new content, select the *page and green plus* icon on the lower left of the portlet. This launches the same full-featured editor you've already seen in the Control Panel, which lets you add and edit content in place as you are working on your page. This is another example of the flexibility that Liferay Portal offers. At times, you may want to add content directly into the Web Content Display portlet of the page you're managing, especially if you are in the process of building the page. At other times, you may want to use the control panel to create content, because at that moment you're more concerned with the creation of the content and not where the content will later be displayed. Either way, Liferay CMS supports both processes equally.

Editing Content

Once the content is displayed—whether you've selected content or created it in the Web Content Display portlet—you can edit the content directly from the Web Content Display portlet or from the Control Panel. To edit it from the Web Content Display portlet, select the *pencil* icon to the lower left of the portlet. This will launch the WYSIWYG editor and from there you can make any necessary changes.

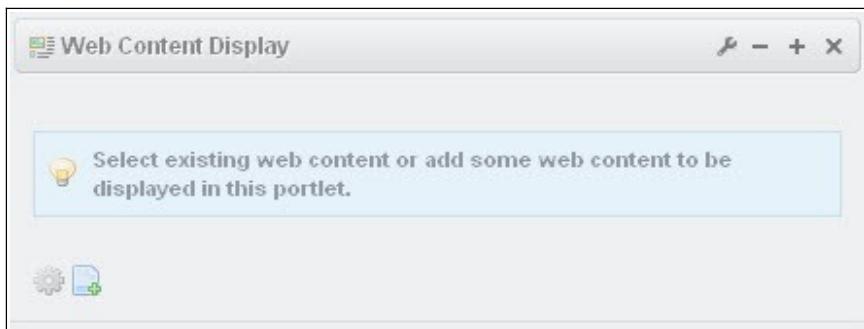


Illustration 51: Web Content Display Portlet

When you publish your content this way, it will become immediately available on the site (unless, of course, you have a workflow enabled, which we'll see below). It's nice to be able to edit the content where it is when you need to, as well as edit it in the Control Panel.

One thing to note here that's important: if you want to view your page the way your users will see it (i.e., without all those portlet controls and icons), you can go up to the Dockbar and select *Toggle Edit Controls*. This makes all those extra controls you see as a portal administrator disappear. If you need to use those controls again, just select *Toggle Edit Controls* again.

Advanced Content Creation

You will find that as your web site grows, managing it becomes more challenging. Without preset limitations, users can display content in any order and in any manner they desire (think huge, flashing letters in a font nobody can read). Also, without a scheduling feature, manually removing or replacing old content becomes cumbersome. Furthermore, without a workflow for approving content, some content might get published that should not have been, or content could get published that contains errors or omissions.

Thankfully, with Liferay Portal's Advanced Content Creation features, managing a complex and rapidly evolving web site is easy. You can use *Structures* to define which fields are available to users when they create content and use *Templates* to define how to display that content. Additionally, you can configure Liferay's built-in *Workflow* system to set up a review and publishing process and then take advantage of the *Scheduling* feature to determine when content is displayed and when it's removed. Liferay Portal gives you the management tools you need to run everything from a simple, one-page web site to an enormous, content-rich site.

Structures

Structures are the foundation for web content. They determine which fields are available to users as they create a new item for display. Structures not only improve manageability for you the administrator, but also make it much easier for the users to quickly add content.

For example, say you're managing an online news magazine. All of your articles need to contain the same types of information: a title, a subtitle, an author, and one or more pages of text and images that comprise the body of the article. If Liferay only supported simple content as has been described above, you would have no way to make sure that your users entered a title, subtitle, and author. You might also get articles that don't match the look and feel of your site. If titles are supposed to be navy blue and they come in from your writers as light blue, you need to spend time reformatting them before they are published.

Structures give you the ability to provide a format for your content so that your users know what needs to be entered to have a complete article. Using structures, you can provide for your users a form which spells out exactly what is required, and which can be automatically formatted properly using a template.

To create a structure, you simply add form controls such as text fields, text boxes, text areas (HTML), check boxes, select boxes, and multi-selection lists, as well as specialized, Liferay-specific *Application Fields* such as Image Gallery and Document Library right onto the structure. Furthermore, you can move the elements around by dragging them where you want them. This allows you to quickly brainstorm different orders for your input fields. Additionally, elements can be grouped together into blocks which can then be repeatable. Template writers can then write a template which loops through these blocks and presents your content in innovative ways, such as in sliding navigation bars, content which scrolls with the user, and more.

Let's take a look at how we edit a structure.

Editing a Structure

Go back to the Control Panel and the Web Content section. Click *Add Web Content* to add another piece of content to your portal. Instead of going right for the content this time, however, we're going to create a structure. To edit a structure, simply select the *Edit* button from the Structures tab.

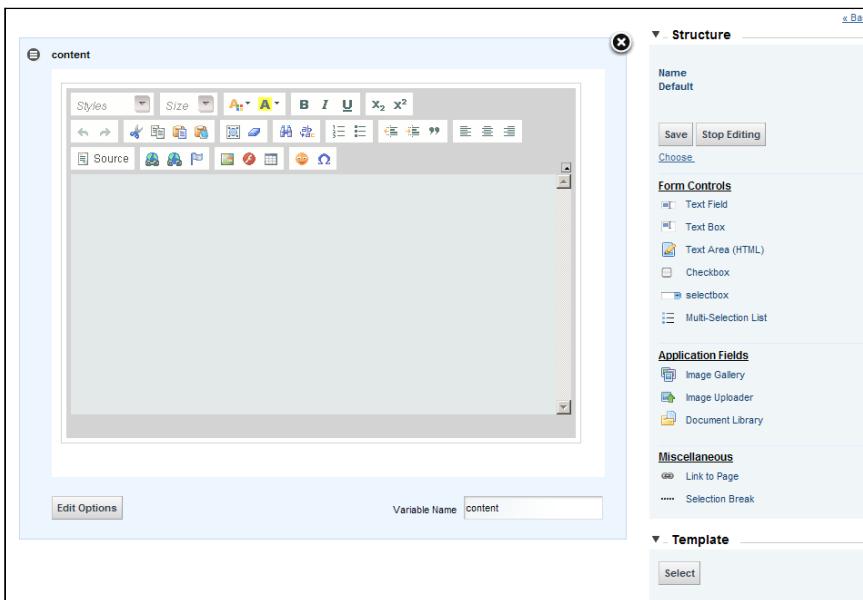


Illustration 52: Structure Editor

It is very easy to edit structures: all you have to do is drag elements into the structure and then give them names. For instance, select the *Checkbox* element under the *Form Controls* tab and drag it onto the structure. You can do the same with any of the elements. To remove it from the structure, simply select the *Delete* icon (black circle with X) in the upper right corner of the element. Take a moment to add, delete, and rearrange different elements.

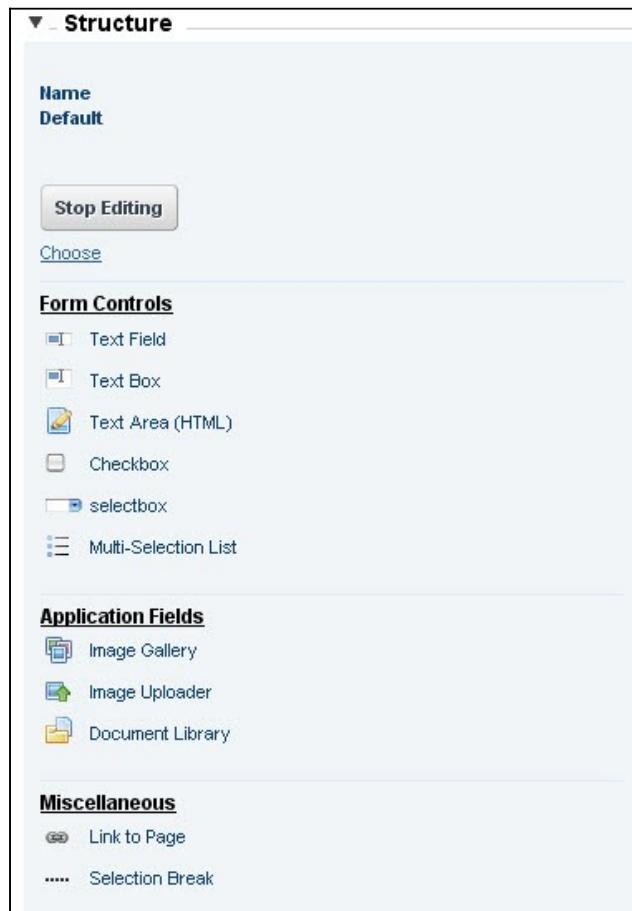


Illustration 53: Structure Elements

Liferay supports the following elements in structures:

FORM FIELDS

Text Field: Used for items such as titles and headings.

Text Box: Used for the body of your content or long descriptions.

Text Area (HTML): An area that uses a WYSIWYG editor to enhance the content.

Checkbox: Allows you to add a checkbox onto your structure. Template developers can use this as a display rule.

Selectbox: Allows you to add a select box onto your structure.

Multi-selection Lists: Allows you to add a multi-selection list onto your structure.

APPLICATION FIELDS

Image Gallery: Allows you to add the Image Gallery application into your structure.

Upload Image: Allows you to add the upload image application into your structure.

Document Library: Allows you to add the Document Library application to your structure.

MISCELLANEOUS

Link to Page: Inserts a link to another page in the same community.

Selection Break: Inserts a break in the content.

Editing Elements

When creating a new structure it is essential that you set the variable names for the elements for the template writers. Otherwise, the generated variable names will be very difficult for a template writer to follow. For Spartan Software, we want to create a series of guides that give a step-by-step list of instructions for the game level designers. In your structure, you can add the element **Text Area (HTML)** which has the Field Label **Instructions**. However, we want to give it the variable name of **Steps**. This can be done very easily: at the bottom of every form element is a **Variable Name** field. Replace the generated name with the name you want to use. There are many other options for fields, including setting tooltips for users. To set these options, select the **Edit Options** button in the lower right corner of the element.

The **Instructions for the User** field is where you can type in instructions for the user and even display it

The screenshot shows the 'Edit Options' dialog box for a form element. The 'Field Type' is set to 'Text Area (HTML)'. The 'Field Label' is 'Instructions'. The 'Variable Name' is 'Steps'. The 'Index type' is 'Not Searchable'. The 'Predefined Value' field is empty. The 'Instructions for the User' field contains the text: 'Add the step-by-step instructions for level designers.' A checked checkbox labeled 'Display as Tooltip' is present. Below it are three unchecked checkboxes: 'Repeatable', 'Required', and 'Localized'. At the bottom are 'Save' and 'Cancel' buttons.

Illustration 54: Editing Options for Elements

as a tooltip. For the Spartan Software Company News structure, type in something that will help users know what to put into the Body element (example: this is an HTML Text area for the body of your content). Also, enable the Display as Tooltip box. Now, when users hover over the Help icon near your title, they will see the instructions you entered.

Assigning Permissions

Permissions to the structure are straightforward. Generally, you don't want most users editing structures as this often requires a developer to modify the template assigned to the structure. However, you do want to make your structure viewable to everyone that will be using it to add web content. You can determine who views a structure by selecting from the *Viewable By* select box beneath the *Permissions* tab. By default the *Anyone (Guest Role)* is selected.



Illustration 55: View
Permissions for Structure

You will also want to determine how users can interact with the structure. You can do this by selecting the *More* link.

Permissions									
	Viewable by	< Hide Options							
Roles	Add Discussion	Delete	Delete Discussion	Expire	Permissions	Update	Update	Discussion	
Guest	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>					
Community Member	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Illustration 56: More Permissions for Structures

From the *More* link, you have the ability to grant or deny permissions based on Roles. For instance, you can give the *Guest* role the ability to *Add Discussion* or you may want to give a *Community Member* the ability to *Delete*. Liferay Portal makes it easy to configure the permissions based on your specific needs for the site.

Templates

Developers create templates to display the elements of the structure in the markup that they want. Content can then be styled properly using CSS, because markup is generated consistently when users enter content. In essence, templates are scripts that tell Liferay how to display the content within the fields determined by the structure. Any changes to the structure re-

quire corresponding changes to the template. Otherwise, new or deleted fields will produce errors on the page. Without a template, the portal has no idea how to display content which has been created using a custom structure.

Template Types (VM, XSL, FTL, and CSS)

Liferay supports templates written in four different templating languages. This is so that you can get started right away. If you have experience with one over another, you can use whichever one you've already used before. If you are just starting with any one of them, we recommend Velocity, as it is less "chatty" than XSL and extremely simple to understand.

VM (Velocity Macro): Velocity is a scripting language that lets you mix logic with HTML. This is similar to other scripting languages, such as PHP, that you may have seen before, though Velocity is much simpler. It's been in the product the longest, so it is probably the most widely used language for templates in Liferay CMS. If you haven't used any of the template languages before, we recommend using Velocity: you'll get up to speed the fastest.

XSL (Extensible Style Sheet Language): XSL is used in Liferay templates to transform the underlying XML of a structure into markup suitable for the browser. While it may not be as clean and compact as Velocity or FTL, it is widely used for transforming XML into other formats.

FTL (FreeMarker Template Language): Freemarker is a templating language which could be considered a successor to Velocity, though it is not yet as popular. It has some advantages over Velocity for which it sacrifices some simplicity, yet it is still easy to use and less "chatty" than XSL.

CSS (Cascading Style Sheets): You can use CSS if your structure is very straightforward and modifications are simple (colors, fonts, layouts, etc). If your structure is more complex, however, you'll need to use one of the other options.

Adding a Template

Liferay CMS makes it easy to create structures, templates, and content from the same interface. Let's go through the entire flow of how we'd create a structure, link it to a template, and then create content based on what we've defined. We'll use Velocity for our template, and we'll lay out the structure fields systematically to go along with the format we've defined for our content.

The screenshot shows the 'Add Template' interface in Liferay's WCM. The form includes fields for ID (disabled), Name, Description, Cacheable (checked), Structure (with Select and Remove buttons), Language Type (VM selected), Script (Browse... button), Format Script (unchecked), Small Image URL (Browse... button), -- OR --, Small Image (Browse... button), and Small Image (unchecked). Permissions are set to Viewable by Anyone (Guest Role) with More Options. Action buttons at the bottom include Save, Save and Continue, and Cancel.

ID

Autogenerate ID

Name

Description

Cacheable ?

Structure

Language Type

Script

Format Script

Small Image URL

-- OR --

Small Image

Small Image

Permissions
Viewable by More Options > ?

Illustration 57: Adding Template Interface

1. Go back to the Web Content section of the Control Panel and click *Add Web Content*.
2. Select *Edit* from the Structure tab.
3. Remove the Content field and add the following fields:

Variable Name	Field Type
Title	Text
Abstract	Text Box
Image	Image Gallery
Body	Text Area

4. Select *Save*.
5. After you have saved the structure, select the *Templates* tab.
6. Select *Add Template*.
7. Type in a name and description and enable the *Autogenerate ID* box.
8. De-select the box labeled *Cacheable*.
9. Select VM as the language.
10. If you've written the script beforehand, you can select *Browse* to upload it from your machine. Otherwise, you can click *Launch Editor* to type the script directly into the small editor window that appears.
11. Select *Save*.
12. Return to the Web Content tab and open the Company News content. You'll see the new element labeled Abstract just below the Title.

Below is the template script for this structure. It is written in Velocity:

```
#set ($renderUrlMax = $request.get("render-url-maximized"))
#set ($namespace = $request.get("portlet-namespace"))
#set($readmore = $request.get("parameters").get("read_more"))
<h1>$title.getData()</h1>
#if ($readmore)
    <p>$abstract.getData()</p>
    <p>$body.getData()</p>
#else
    <p>
        
        $abstract.getData()
    <p>
        <a href="$renderUrlMax&$namespace;read_more=true">Read More</a>
#end
```

This template is pretty small, but it actually does quite a bit. First, a portlet URL which maximizes the portlet is created. Once this is done, we get the namespace of the portlet. This is important because we don't want our URL to collide with another URL that might be on the page.

After this, we attempt to get a request parameter called `read_more`. Whether or not we were successful in getting this parameter is the key to the rest of the script:

- If we were successful in getting the `read_more` parameter, we display the abstract and the body below the title (which is always displayed).
- If we were not successful in getting the `read_more` parameter, we display the image, the abstract, and the link we created above, which sets the `read_more` parameter.

When this template is rendered, it looks something like this:

New titles coming soon!

Spartan software is poised to announce the release of new titles!

[Read More](#)



Your Rating Average (0 Votes)

★ ★ ★ ★ ★ ★ ★ ★ ★ ★

No comments yet. [Be the first.](#)

Illustration 58: By default, the content displays like this. If the user clicks the Read More link, the portlet will display the body field as well. Note we've also enabled comments and ratings in this Web Content Display portlet.

Of course, there is much, much more you can do with structures and templates. Check out the Liferay Wiki (<http://wiki.liferay.com>) for further information and examples.

Assigning Template Permission

Permissions for templates are very similar to permissions for structures. Generally, you only want specific developers accessing the template. However, you may want to make the templates viewable to some content creators who understand the template scripting language, but are not directly writing the scripts. You can determine who views the template by selecting from the *Viewable By* select box beneath the *Permissions* tab. By default the *Anyone (Guest Role)* is selected.

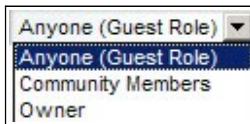


Illustration 59: View
Permissions for Templates

You'll also want to determine how users can interact with the template. You can do this by selecting the *More* link.

Permissions

Viewable by **Anyone (Guest Role)** [« Hide Options](#)

Roles	Delete	Permissions	Update
Guest	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Community Member	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Save **Save and Continue** **Cancel**

Illustration 60: More Permissions for Templates

From the *More* link, you have the ability to grant or deny permissions based on Roles. For instance, you may create a role with the ability to update the template and create a second role that can both update and delete. Liferay Portal makes it possible to assign permissions based on the roles and responsibilities within your organization.

Advanced Publishing Options

As we discussed above, as your site becomes larger and more complex, management of the content becomes more challenging. We've discussed Liferay management tools that help you create content quickly and in an orderly fashion. We created a simple announcement and then covered Liferay's new structure editor that allows you to quickly design a structure and prepare it for the template designers. Then, we went through applying a template to the structure. We demonstrated how to display our content using the Web Content Display portlet. Now, we're ready to take advantage of Liferay's advanced publishing options.

If a web site isn't properly managed, it can quickly become out of date, and that drives viewers away. If people are finding your site because of search engines, you don't want them presented with outdated (and now inaccurate) web content.

Additionally, you may want the ability to create content and send it through a review and approve process weeks before you want it displayed on the web site. Liferay gives you this flexibility with the *Schedule* feature in the Web Content portlet.

Scheduling Web Content

You can publish your content on a schedule. You can determine when the content will be displayed, expired, and/or reviewed. This is an excellent way to keep your site current and free from outdated (and perhaps incor-

rect) information. The scheduler is built right into the form that your users make use of to add web content, in the same column as the structure and template selectors.

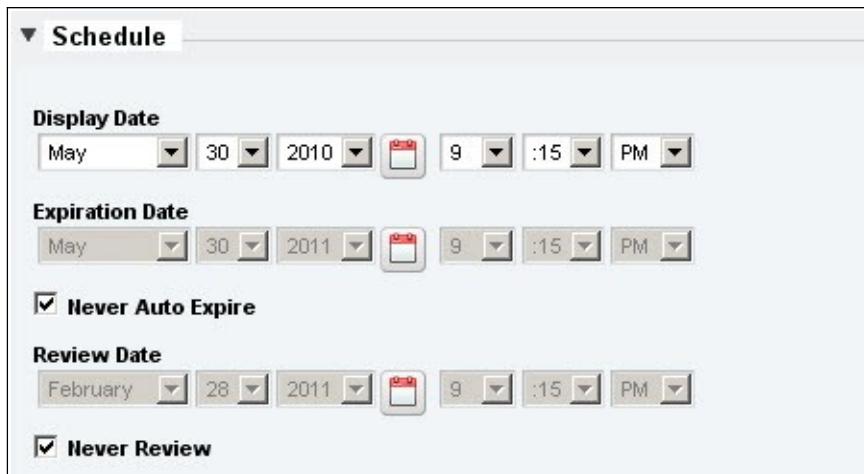


Illustration 61: Schedule for Publishing Content

Display Date: Allows you to determine (within a minute) when content will be displayed.

Expiration Date: Allows to set the date when the content will expire. The default is one year.

Never Auto Expire: Allows you set your content to never expire.

Review Date: Allows you to set a date when you want the content re-reviewed.

Never Review: Allows you to determine that your content will not be re-reviewed.

As you can see, the scheduling feature in Liferay Portal gives you great control in managing when, and for how long, your web content is displayed on your web site. Additionally, you have the ability to determine when your content should be reviewed for accuracy and/or relevance. This makes it possible to manage your growing inventory of content.

Tags and Categories

Though tags and categories will be more fully described in Chapter 5, it is important to mention them here. Tags are keywords that can be attached to web content in order to help users find content. Categories are a hierarchical

organization of content whose structure can be defined by administrators. With tags and categories, you can make it easier for your users to find your content through search or navigation.

Why Tag?

Tags are keywords attached to a piece of web content in the portal. By assigning a tag to web content, you define metadata about that content. This can be used by Liferay's search engine to score the results of a search, enabling users to find content that is most relevant to their search. Tags can be created on the fly by the creator of the content, and it is important to tag your content whenever it is created. If you don't tag your content, all the search engine has to go on is the full text of the content when a user does a search, and that might not produce the most optimal results.

Tagging also helps with navigation. Liferay Portal has two portlets specifically designed for navigating content using tags: Tag Cloud and Tag Navigation. If you add either of these to a page, you can use them to show the topics contained in your content.

Who Tags?

Tags in web content are added by the creator of the content. They can be added on the fly or they can be selected from the existing library of tags. For most of the portal, users tag content, but for web content, only the content creator tags the content, because there is no user interface for regular users to tag web content.

It is important that you both tag and categorize your content when you enter it.

What is the Difference Between a Tag and a Category?

Categories are defined by someone with administrative access to the content. They are hierarchical, tree-like structures that users can use to find content. Categories are different from tags in that they are never created by end users. Instead, categories define how your content is organized from the point of view of the owner of the content. A good example of categories might be the table of contents of a book: it shows the hierarchical structure and organization for all of the content within that book. This shows that the structure of the book has been planned ahead of time by the creator of the book. Categories do the same thing. By contrast, tags are like the index of a book: they show where many different topics are mentioned within the book in alphabetical order. When a search is done throughout the book, even the author might be surprised at how many times he or she mentions a particular topic outside of its category. So both ways of organizing your content are important, especially if your users will be using search to find content.

Tagging and categorizing web content is easy. You can do it at the bottom of the same form you use to add content. If you open the *Categorization* section of the form, you'll be presented with an interface for adding tags and categories.

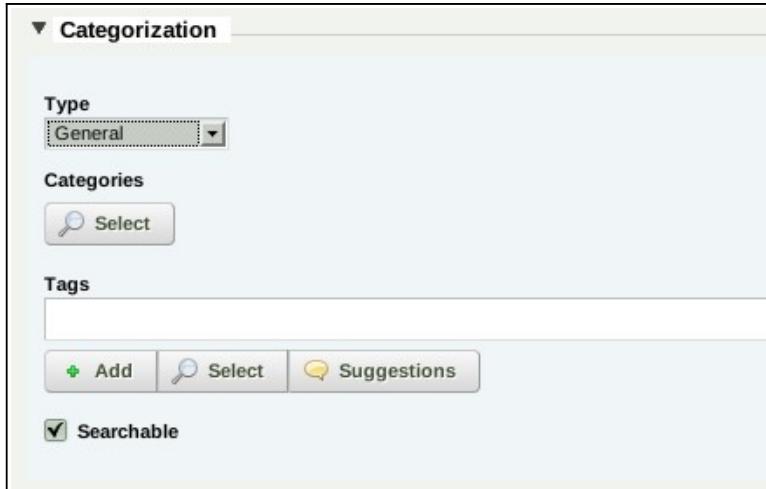


Illustration 62: Tagging and categorizing content can be done at the same time you create it.

The Control Panel contains an interface for managing tags and categories for each community or organization in the portal. This interface can be used administratively to manage your tags and categories. We'll take a look at this interface in the next chapter.

Using Liferay's Integrated Workflow with Content Management

Workflow is essentially a predetermined sequence of connected steps. In Liferay CMS, workflow is designed to manage the creation, modification, and publication of web content. You can set up a workflow so that content cannot be published without going through an approval process that you design. In this way, content goes up on your site only after it has been reviewed and approved.

Liferay's workflow engine is called Kaleo workflow, and it ships with Liferay CE. If you have uninstalled it or are using EE, it needs to be installed and configured separately, and this is covered in Chapter 7. For now, we'll assume it's installed and show you how you can take advantage of workflow in getting your content through any approval steps between creation and publication.

To enable workflow for Web Content, navigate to the Control Panel and select *Workflow Configuration*. From there, select a workflow that has been deployed to Liferay.

The screenshot shows a 'Workflow Configuration' page. It has a header 'Workflow Configuration'. Below it is a table with two columns: 'Resource' and 'Workflow'. The resources listed are 'Blogs Entry', 'Comments', 'Message Boards Message', 'Web Content', 'Document Library Document', and 'Wiki Page'. The 'Workflow' column contains dropdown menus. For 'Blogs Entry', 'Comments', 'Message Boards Message', and 'Wiki Page', the dropdown says 'Default: No workflow'. For 'Web Content', the dropdown says 'Single Approver (Version 1)'. For 'Document Library Document', the dropdown says 'Default: No workflow'. At the bottom left, it says 'Showing 6 results.' and there is a 'Save' button.

Resource	Workflow
Blogs Entry	Default: No workflow
Comments	Default: No workflow
Message Boards Message	Default: No workflow
Web Content	Single Approver (Version 1)
Document Library Document	Default: No workflow
Wiki Page	Default: No workflow

Showing 6 results.

Save

Illustration 63: Enabling Workflow for Content Management

As you will discover in Chapter 7, you can design workflows to suit your organization's approval process. For Spartan Software's implementation we will use the *Single Approver* workflow which ships with the product.

Defining Workflows for Web Content

Let's go ahead and set up Liferay's Workflow for the Spartan Software web site.

1. Go to the Control Panel and select *Workflow Configuration* from the left panel.
2. From the select box, choose *Single Approver* for Web Content. Click *Save*. Note that you can add workflow to many of Liferay's portlets.

That's all it takes to set up workflow for web content. Now publishing content works a little bit differently. Let's go through the process of publishing some sales goals for a new Spartan Software game. Return to the home page and click on the *Add Web Content* icon on the Web Content Display portlet. Call it *Sales Goals* and enter some content. Notice that the Publish button is now gone. In its place is a *Submit for Publication* button. Go ahead and click it.

The screenshot shows the 'Web Content' section of the Liferay Control Panel. A content item titled 'Sales Goals' is displayed. The 'Content' tab is selected. Key details shown include:

- ID: 10619
- Version: 1.0
- Status: Pending (REVIEW_CONTENT)
- Name: Sales Goals
- Language: English (United States)
- Default Language: English (United States)

Illustration 64: Pending Workflow

Next, go to the *Workflow Tasks* in Control Panel and then select *My Workflow Tasks*. You will see the option to Review Content for Sales Goals. It shows because you are logged in as an Administrator. There is also a Content Approvers role which is defined by this workflow, and anyone in this role can approve content as well.

To approve the content, you must first take ownership of it. Click on the task. You should see the screen below.

The screenshot shows the 'Review' screen for a content item titled 'Welcome to Spartan'. The left panel displays:

- Assigned to: nobody - [Assign to Me](#) - [Assign to...](#)
- Create Date: 8/13/10 4:23 AM
- State: Review
- Due Date: Never - ([Change](#))
- Preview of Web Content: Shows a preview of the 'Welcome to Spartan' page.
- Activities: Shows a list of activities.
- Comments: Shows a comment from 'By Test Test'.

The right panel shows a summary of assigned tasks:

- Review (with two checked items)
- Assign to Me
- Assign to...
- Update Due Date

Illustration 65: My workflow tasks shows tasks that are assigned to your role. To take up a task, you must assign it to yourself first.

Taking ownership of, reviewing, and approving content is very easy:

1. Click the *Assign to Me* button. Alternatively, you could assign it to someone else in the Content Approvers role or create / update a due date for the content's approval.
2. Once you've assigned it to yourself, buttons allowing you to approve or reject the content appear. Click *Approve*.
3. You're asked to submit a comment. You'd have to do this for either *Approve* or *Reject*. Add a comment and click *Save*.
4. The content is now approved.

In a real world situation, you obviously wouldn't want the person who created the content to be the one who approves it. Instead, you would have one or more roles designed for users who will be creating content, and you will have specific users assigned to one or more roles for approving content. Our example was of a very straightforward workflow, as it has only a single approver. Kaleo workflow allows you to design workflows that go through as many steps as you need to conform to your business processes. This is beyond the scope of this chapter, but is covered in Chapter 7.

Using the Asset Publisher Portlet

As we create web content, it's important to keep in mind that to Liferay, the pieces of content are assets, just like message board entries and blog posts. This allows you to publish your web content using Liferay's Asset Publisher.

You can use the Asset Publisher to publish a mixed group of various kinds of assets such as images, documents, blogs, and of course, web content. This helps in creating a more dynamic web site: you can place user-created wiki entries, blog posts, or message board messages in context with your content. Let's take a look at some of its features.

Querying for Content

The Asset Publisher portlet is a highly configurable application that lets you query for mixed types of content on the fly. By giving you the ability to control what and how content is displayed from one location, the Asset Publisher helps you to "bubble up" the most relevant content to your users.

To get to all of the portlet's options, click the *Configuration* link in the portlet's menu (the wrench icon).

Selecting Assets

The ability to configure how content is displayed and selected by your users further demonstrates the flexibility of the Asset Publisher. You get to

choose how content is displayed. You can select it manually for display in a similar way to the Web Content Display portlet, or you can set up predefined queries and filters and let the portal select the content for you, based on its type or its tags and categories.

Let's first take a look at how we might select content manually. You'll see that it's very similar to the Web Content Display portlet.

Selecting Assets Manually

By selecting *Manual* from the select box beneath *Asset Selection*, tell the Asset Publisher that you want to select your content manually. You can select what you want to be published within the portlet, or you can create new content right from within the Asset Publisher.

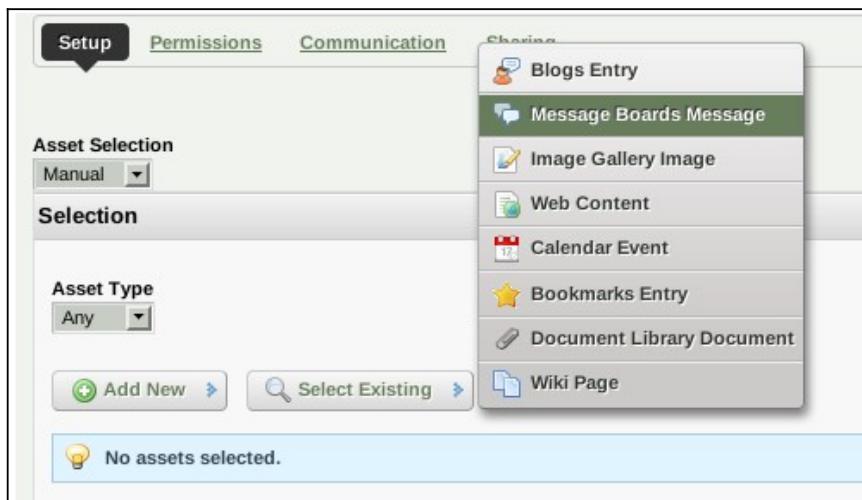


Illustration 66: Selecting assets manually is very similar to the Web Content Display portlet, except you have many other content types from which to choose.

Clicking *Add New* gives you a menu of options, enabling you to create the content right where you are. You can create blogs, bookmarks, calendar entries, documents, images, and of course, web content. Anything you create here will be added to the list below of assets that will be displayed by the portlet.

Clicking *Select Existing* gives you a similar menu, except this time you can pick from existing content in the portal that either you or your users have created. Has someone written an excellent wiki page that you want to highlight? Select it here, and it will be displayed.

The Asset Publisher gives you the ability to mix and match different content types in the same interface. Once you have your content selected, you can move on to the display types to configure how the content appears.

Most of the time, however, you'll likely be using the Asset Publisher to select content dynamically.

Selecting Assets Dynamically

The default behavior for the Asset Publisher is to select assets dynamically according to rules that you give it. These rules can be stacked on top of each other so that they compliment each other to create a nice, refined query for your content. You have the following options for creating these rules:

Scope: Choose the communities or organizations from which the content should be selected.

Asset Type: Choose whether you will display any asset or only assets of a specific type, such as only web content, only wiki entries, or any combinations of multiple types.

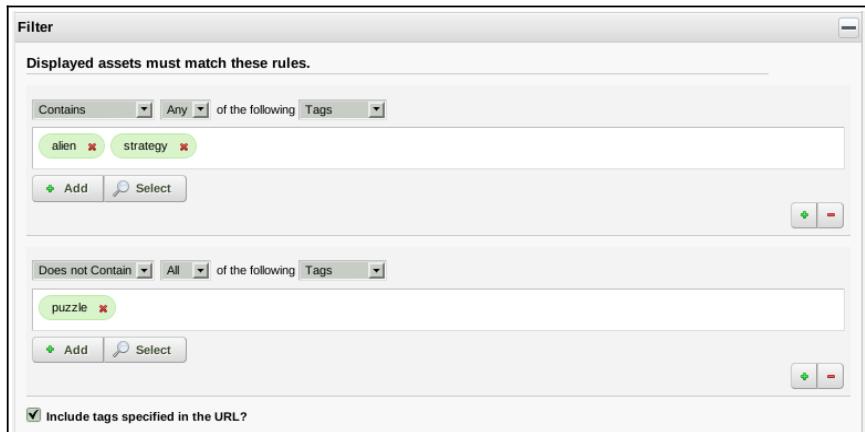


Illustration 67: You can filter by tags and categories, and you can set up as many filter rules as you need.

Filter Rules: Add as many filters on tags or categories as you like. You can choose whether the content contains or does not contain any or all categories or tags that you enter.

Once you've set up your filter rules for dynamically selecting your content, you can then decide how the content will be displayed.

Ordering and Grouping

You can display the content returned by the filters above in order by title, create date, modified date, view count, and more in ascending or descending order. For instance, you may have a series of “How To” articles that you want displayed in descending order based on whether the article was tagged with the *hammer* tag. Or, you may want a series of video captures to display in ascending order based on a category called *birds*. You can also group by *Asset Type* or *Vocabularies*. Vocabularies are groups of categories defined by administrators in the *Categories* section of the Control Panel. Again, we'll see more about categories in Chapter 5.



Illustration 68: Ordering and Grouping

In the *Ordering and Grouping* section of the Asset Publisher, you have great control over how content is ordered and grouped in the list, but this is only one aspect of how your content will be displayed. You can refine the display through many other display settings.

Display Settings

The Display Settings section gives you precise control over the display of your assets. There are a multitude of options available to configure how you want your content to appear. You can configure the style, length of abstracts, behavior of the asset link, maximum items to display, pagination type, and file conversions. Additionally, you can enable printing, flags, ratings, comments, and comment ratings, and these work the same way they do in the Web Content Display portlet.

Display Settings

Display Style

Abstract Length

Asset Link Behaviour

Maximum Items to Display

Pagination Type

Exclude Assets with 0 Views

Show Available Locales

Enable Conversion To

DOC ODT PDF RTF SXW TXT

Enable Print

Enable Flags

Enable Ratings

Enable Comments

Enable Comment Ratings

Illustration 69: Display Settings

DISPLAY STYLE

Abstracts: Shows the first 200-500 characters of the content, defined by the **Abstract Length** field.

Table: Displays the content in an HTML table which can be styled by a theme developer.

Title List: The content's title as defined by the user who entered it.

Full Content: The entire content of the entry.

OTHER SETTINGS

Asset Link Behavior: When the link to the asset is clicked, it can be displayed in the Asset Publisher or in the portlet to which the asset belongs, such as the Blogs or Message Boards.

Maximum Items to Display: You can display 1-100 items.

Pagination Type: Select Simple or Regular. Simple shows previous and next navigation; regular includes a way of selecting the page to which you'd like to navigate.

Exclude Assets with 0 Views: If an asset has not been viewed, exclude it from the list.

Show Available Locales: Since content can be localized, you can have different versions of it based on locale. This will show the locales available, enabling the user to view the content in the language of his or her choice.

Enable Conversion To: If you have enabled Liferay Portal's OpenOffice.org integration, you can allow your users to convert the content to one of several formats, including PDF.

Below these options are the same ones in the Web Content Display portlet: enable print, enable comments, enable ratings, etc.

Show Metadata: Allows you to select from the available metadata types (see below).

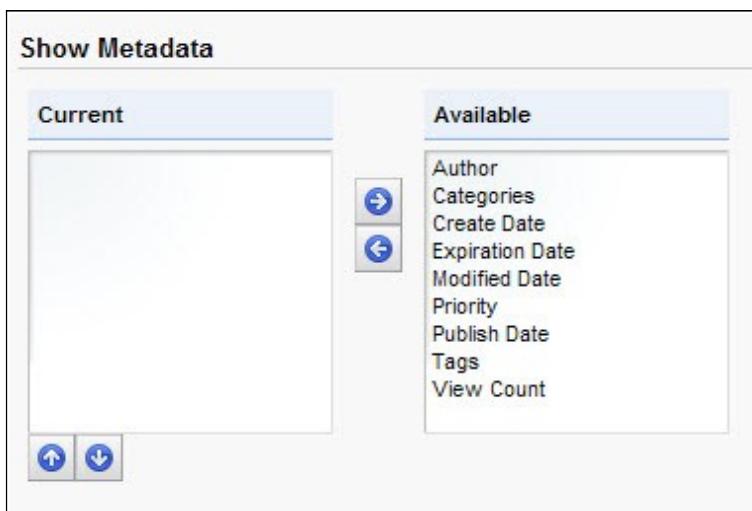


Illustration 70: Show Metadata

Enable RSS Subscription: This gives users the ability to subscribe to the content via RSS Feeds.

The Display Settings section of the Asset Publisher has numerous options to help you configure how your content selections are displayed to your users. Even though there are many choices, it's easy to go through the options and quickly adjust the ones that apply to you. You'll want to use the Asset Publisher to query for mixed assets in the portal that have relevant information for your users.

Summary

This chapter has been your guide to Liferay Web Content Management. We've seen how pages are created and managed in Liferay communities and organizations. It is easy to create whole page hierarchies without ever leaving your browser. You can import and export pages using LAR archives, and these can also be used to transfer a site from one Liferay Portal server to another.

Liferay CMS also includes a powerful staging environment, allowing you to stage content locally on the same server or remotely to another server. And when working on pages, you have the ability to use layouts and nested portlets to design every page to look exactly the way you want it to look.

Whether your site is small and static or large and dynamic, Liferay's WCM enables you to easily plan and manage it. With tools such as the WYSIWYG Editor, Structures, and Templates, you can quickly add and edit content. With the Web Content Display and Asset Publisher, you can rapidly select and configure what and how your content is displayed. And by using Liferay's integrated workflow, you can set up custom publishing rules to fit your organization. You will find that managing your site becomes far easier when using Liferay's Web Content Management system.

5. LIFERAY COLLABORATION SUITE

Liferay Portal ships with a robust suite of collaboration applications which you can use to build communities of users for your site. These applications are best-of-breed applications with all the features you would expect of standalone versions that are outside a portal. The difference with Liferay's collaboration suite, however, is that all of the applications share a common look and feel, security model, and architecture. And they inherit all of the strengths of being a part of the Liferay development platform, so you can use them in combination with Liferay's user management and Content Management features to build a well-integrated, feature-rich web site.

This chapter will focus on the use of Liferay's collaboration suite. You will learn how to set up and administer:

- Blogs
- Calendars
- Chat
- Mail
- Message Boards
- Wikis

You will see how all of these features can work together to provide an enhanced experience for your users, as well as giving you the tools to help build a community that keeps coming back.

Scopes

As we learned earlier, roles can be scoped by the portal, by a community, or by an organization. This means that the role only takes effect for the scope in which it resides. For example, a Message Boards Administrator role with complete access to the Message Boards portlet would have different permissions based on the role's scope. If it's a portal role, members have permission to administer message boards across the portal. If it's a community role, members have permission to administer message boards only within the community in which they are members of the role. If it's an organization role, members have permission to administer message boards only within the organization in which they are members of the role.

We also use the word *scope* to refer to the data set of a portlet. When a portlet is added to a page in a community or organization, it is *scoped* for that community or organization. This means that its data belongs to that community or organization. If the portlet is added to a page in a different community or organization, it will have a completely different data set. This is how the Message Boards portlet can be placed in one community and have one set of categories and threads, and then be placed in a different community and contain a completely different set of categories and threads.

One limitation of this is that by default, you can have only one Message Boards portlet per community or organization. If you have one Message Boards portlet on one page in a community and you add the portlet to another page in the same community, the second Message Boards portlet will contain exactly the same data as the first, and this is because of its scope. It is scoped for that community. Many of Liferay's portlets work this way.

To get around this, some Liferay portlets can have scopes that go beyond just the specific community or organization upon which the portlet has been placed. You can now scope them by page. If you set the scope of a portlet to be the page instead of the whole community or organization, you can add any number of these portlets to different pages, and they will have different sets of data. This allows you to have more than one message board per community or organization if you wish. All portlets, however, default to the "native" configuration, and have their scopes set to the community or organization where they are placed.

Unless otherwise noted, all of the portlets in this chapter support scoping by either the page or the community / organization to which they belong. This gives you much more flexibility in how you want to set up your portal. By default, however, the scope remains the same as it always has, and is set to be by the community or organization. If you want to change the scope, it only takes a few simple steps.

1. Click the *Menu* icon in the portlet window (the wrench).

2. Select *Configuration*.
3. Select the *Scope* tab.
4. Modify the Scope to be the current page.
5. Click *Save*.



Illustration 71: Changing the scope for a portlet

That's all it takes to change the scope for a particular portlet instance. By setting the scope to *page*, you can add as many of these portlets to a particular community or organization as you want, provided they all are added to different pages.

Archived Setups

Another useful feature in Liferay's portlets is Archived Setups. This means that once you configure a portlet, you can save those settings in an "archive" that the portal will save for you. If someone goes in and changes the settings of a particular portlet, it then becomes easy to revert those changes back to the original, archived configuration.

To create an archived setup, click the *Configuration* option from the menu in the portlet's title bar. If the current settings of the portlet you are configuring are the ones you want to archive, click the *Archived* tab. If not, change and save the settings until you have the portlet configured the way you want it, and then click the *Archived* tab.

There is only one field to fill out: a name for your archive. Create a name for your archive and click *Save*. You will now see your archive in the list. If

for whatever reason you need to revert the portlet to these archived settings, you can click *Actions* → *Restore* next to the archived setup you want to restore.

Unless otherwise noted, all of the portlets in this chapter support this feature. This is particularly useful for portlets that have a lot of configuration options, such as the Message Boards portlet.

Permissions

All of Liferay's portlets support Liferay's robust, fine-grained permissions system. Some higher level permissions can be configured here: whether a role can add the portlet to a page, can configure the portlet, or whether the role can view the portlet. To set these permissions, go to the *Configuration* menu and click on *Permissions*. This will show you a table of roles defined in the portal. Select which roles are allowed to see the portlet and which roles are allowed to configure the portlet, and then click *Submit*.

Sharing

The web was once thought of as a number of islands of applications in a vast universe of “cyberspace.” Everybody tried to make their island the biggest. Some succeeded and some failed. More recently, the concept of the web as an application itself has taken hold, and so widgets have become very popular nowadays. This concept is part of the “Web 2.0” concept and is very much enabled by widgets. So what is a widget? A widget is a small piece of code which provides a piece of functionality, can be included on any web site, but does not necessarily have to be hosted by that web site. If you have ever embedded a YouTube video on your own web site so that users could watch a video without actually having to visit <http://youtube.com>, then you have already used a widget.

Liferay supports serving its portlets as widgets. You can embed a particular instance of a portlet running on your site into another site, such as Facebook. This opens up a whole new avenue of exposure to your web site that you would not otherwise have had. In fact, this is how all those Facebook games work.

To share one of your portlets as a widget, go to the *Configuration* option in the menu in the portlet's title bar. Then click the *Sharing* tab. There are five subtabs under sharing: Any Web Site, Facebook, Google Gadget, Netvibes, and Friends.

Any Web Site

Copy and paste the snippet of code into the web site to which you want to add the portlet as a widget. That's all you need to do. When a user loads the page on the other web site, the code will pull the relevant portlet from your site and display it.

Facebook

You can add any Liferay portlet as an application on Facebook. To do this, you must first get a developer key. A link for doing this is provided to you in the Facebook tab. You will have to create the application on Facebook and get the key and canvas page URL from Facebook. Once you have done this, you can copy and paste their values into the Facebook tab. Your portlet will now be available on Facebook as a Facebook application.

The screenshot shows the Liferay Message Boards interface. At the top, there are tabs for 'Categories', 'Recent Posts', and 'Statistics'. Below the tabs is a search bar and a 'Search Messages' button. A message indicates 'Showing 11 results.' The main area lists categories with their subcategories and statistics:

Category	Categories	Threads	Posts
Liferay English Subcategories: 1. Installation / Deployment / Setup, 2. Using Liferay, 3. Development, 4. Custom Theme Development, 5. Portal Framework, More »	13	10076	33539
Liferay Core Development & Contributions This category functions as a traditional "developers mailing list" for discussing the development of the Liferay core. To limit traffic, only Liferay core developers can post new threads to this category. Anyone can read or reply to an existing thread. Subcategories are available to propose and discuss about some specific topics such as accessibility, translations and contributions. Subcategories: Accessibility, Beta Testing, Contributions, PortletFaces, Translations	5	187	1373
Liferay em português	0	131	451
Liferay en español Subcategories: 1. Instalación / Despliegue / Configuración, 2. Usando Liferay, 3. Desarrollo, 4. Creación de temas de apariencia, 5. Portal Framework, More »	9	1480	5101
Liferay en français	0	485	1648
Liferay in deutsch	0	176	554
Liferay in italiano	0	288	806
Liferay Legacy Posting of new threads has been disabled for this category. Replies to existing threads is still allowed so conversations can continue. Please use the new section with categories for all new posts.	0	12290	38191
Liferay Social Office	0	55	305
Liferay 中文 Subcategories: Liferay 使用, Liferay 开发, Liferay 社区	3	239	894
Liferay 日本語	0	4	13

Showing 11 results.

Illustration 72: Liferay's forums in Facebook

Google Gadget

iGoogle is a service provided by Google that lets users create a customizable page and add *Gadgets* to that page. Liferay can serve up portlets to be used as Google Gadgets on an iGoogle page.

Check the box labeled *Allow users to add [portlet-name] to iGoogle*. Copy and paste the URL provided into Google's *Add a feed or gadget* feature on the iGoogle configuration page, and Liferay will serve that portlet directly onto your iGoogle page. The URL provided is unique to the specific instance of the portlet, so you could serve multiple instances of the same portlet as different Google Gadgets.

This feature could be useful to allow users to view what's happening on your portal at a glance, with asset publishers or custom RSS feeds. You could also use Liferay's API to build your own portlet and provide the URL for users to place on their iGoogle pages.

Netvibes

Netvibes offers a similar service to iGoogle – users can log in, create their own personal portal, called a *dashboard*, and add customizable widgets to the dashboard that they create. To set up Netvibes support for a particular portlet, check the *Allow users to add [portlet-name] to Netvibes pages* box. You can then use the provided URL to create a custom Netvibes widget based on the instance of the portlet that you're using.

Friends

The final sub-tab in the *Sharing* tab is called *Friends*. This tab has a single check box that allows you to give your friends permission to add the application as a widget to another web site. This could be particularly useful for your blog or your calendar if you wish to share them.

Blogs

The word *Blog* is an apostrophe-less contraction of the two words *web log*. Blogs were first popularized by web sites such as Slashdot (<http://slash-dot.org>) which have the format of a running list of entries to which users could attach comments. Over time, more and more sites such as Digg, del.icio.us, and Newsvine adopted the format, empowering users to share their opinions and generating lively discussions.

Over the course of time, blogging sites and applications began to appear, such as blogger.com, blogspot.com, TypePad, WordPress, and Web Roller. These applications allow *individuals* to run their own web sites in the same format: a running list of short articles to which readers who are registered with the site can attach threaded comments. People who run a blog are called *bloggers*, and sometimes they build a whole community of readers who are interested in their blog posts. Additionally, there are several famous people who run their own blogs. It gives people an outlet for self-expression which they would not otherwise have, and the ubiquity and wide reach of the Internet ensures that if you have something important and interesting to say, somebody will read it.

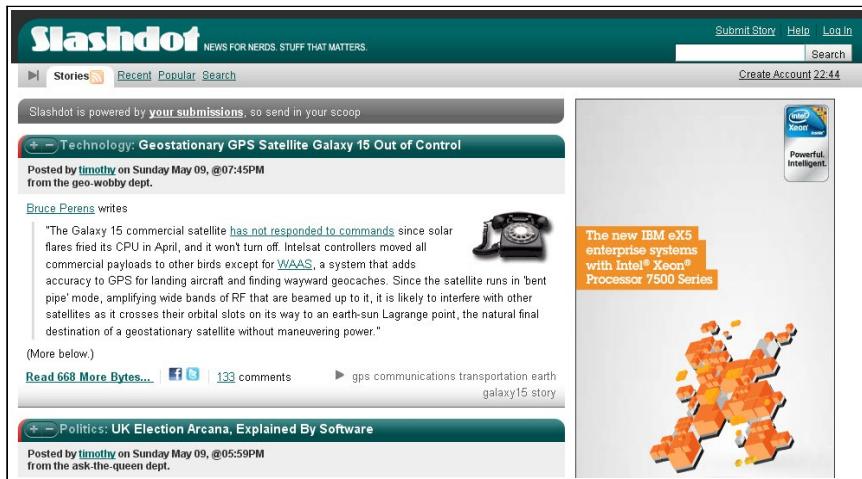


Illustration 73: Slashdot, one of the first blogs on the Internet

Liferay Portal has a portlet called the Blogs portlet which allows you to provide a blogging service to users of your web site. In fact, Liferay extensively uses the Blogs portlet on <http://www.liferay.com> to provide employees with blogs of their own. In addition to the Blogs portlet, there is also a **Blogs Aggregator** portlet which can take entries from multiple users' blogs and put them all in one larger list. We will go over how to use both of these portlets to create a blogging site for your users.

The Blogs Portlet

The Blogs portlet is available from the *Collaboration* section of the *Add → More* menu. You will notice that it is an Instanceable portlet, meaning that it can only be added once to any community or organization. This allows you to use the Blogs portlet to create a shared blog to build a site like Slashdot or to create multiple personal blogs to build a site like <http://blogger.com>. What's the difference? Adding the Blogs portlet to a Community or Organization page creates a shared blog for members of the Community or Organization. Adding the Blogs portlet to a user's personal space creates a blog for just that user. Either way, the Blogs portlet works the same. And of course, you can change



Illustration 74: Initial view of the Blogs portlet

the Blog portlet's scope to have different blogs on different pages in the same community.

By default, the Blogs portlet will display the latest entry in its entirety. Since we have just added the portlet to a page, we have no entries, so the portlet will be empty. Before we start adding entries, we'll configure the portlet so that it displays entries according to our specifications.

Configuring the Blogs Portlet

The Blogs portlet is easy to configure. Click on the *Menu* icon in the portlet's title bar and select *Configuration*. Beneath the Setup tab, you will see another row of options.

Email From: Selecting the *Email From* tab allows you to define the *From:* field in the email messages that users receive from the Blogs portlet.

Entry Added Email: This option allows you to enter a subject and body of the emails sent out when a new Blog entry has been added.

Entry Updated Email: This option allows you to enter a subject and body of the emails sent out when a new Blog entry has been updated.

Display Settings: Here, you can configure the various display options for the Blogs portlet.

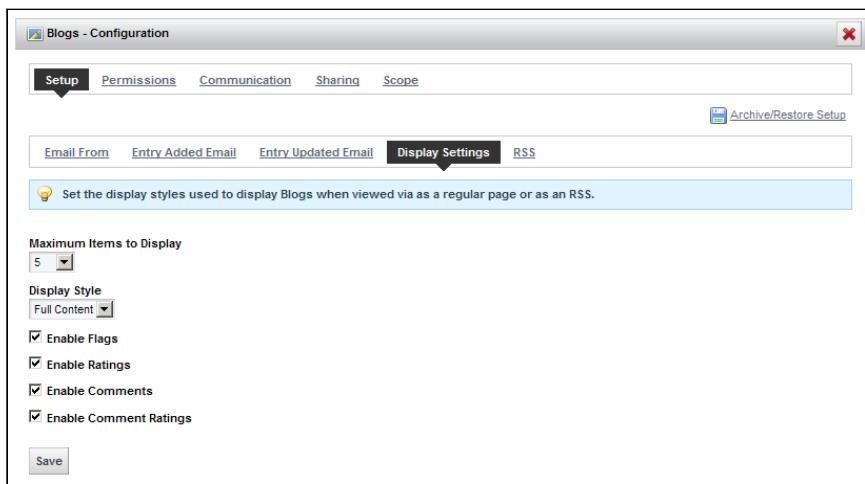


Illustration 75: Blogs Configuration

The *Display Settings* tab lets you configure the formatting of the Blogs portlet. You can choose the most optimal ways to display your entries as well as configure how you want your users to interact with you as you post those entries.

Maximum Items to Display: This allows you to choose the total number of blog entries to display on the initial page. You can choose up to 100 to be displayed.

Display Style: Choose between Full Content, the Abstract, or just the Title. Setting this to Abstract causes Liferay to display only the first 30 words of your blog entries, with a Read More link at the bottom of each to allow users to read the whole entry if they wish.

Enable Flags: Allow users to flag content as inappropriate and send an email to the administrators.

Enable Ratings: Allows you to enable your users to rate your blog entries from one to five stars.

Enable Comments: Allows readers to comment on your blog entries.

Enable Comment Ratings: Allows readers to rate the comments which are posted to your blog entries.

Click on the RSS tab to configure the way in which blogs will display to RSS readers. Here, you can choose how you want your blog entries to be published to feed readers and outside web sites.

Maximum Items to Display: Allows you to determine how many blog entries will be displayed at once. The default is set to twenty.

Display Style: You can select to display the full content, abstracts, or title of your blogs.

Format: You can choose which format you want to deliver your blogs: RSS 1.0, RSS 2.0, or Atom 1.0.

Permissions

Depending on whether this is a personal blog or a shared blog, you may want to modify the permissions on the blog. By default, the permissions are set up for a personal blog, so only the owner of the community to which the portlet has been added will be able to add entries. If you want to share a blog with multiple users, it is easy to do.

First, create a role for your bloggers and add them to the role. Next, click the *Permissions* button on the Blogs portlet. You will now see a list of both portal and Community / Organization roles, and currently only the owner is checked. Check off any other role or team that should have the ability to add blog entries, and then click *Save*.

Adding Blog Entries

Now you're ready to begin adding blog entries. Click the *Add Blog Entry* button. You will see the following data entry screen:

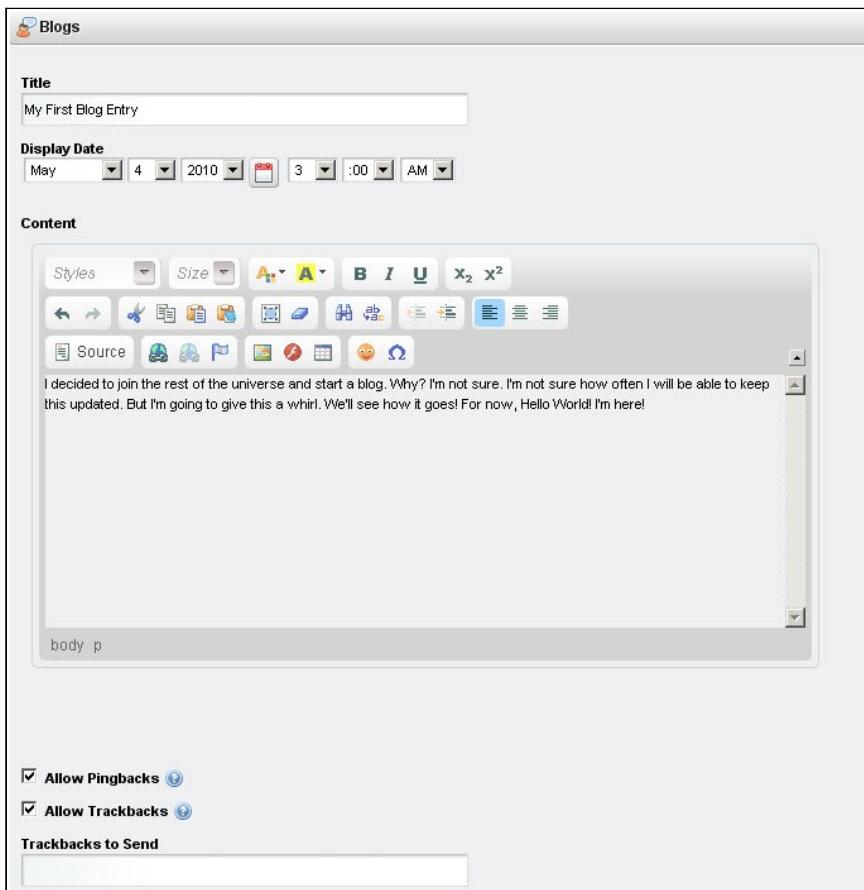


Illustration 76: Adding a Blog entry

There isn't much difference between this screen and any other data entry screen within Liferay Portal. You get a title, a way of scheduling when the entry is to appear, and a rich editor that allows you to format your entry the way you want, complete with embedded images, videos, and the like.

Note also that as you type, the entry is automatically saved as a draft at periodic intervals. This gives you peace of mind in using the portlet from within your browser, since you won't lose your entry in the event of a browser crash or network interruption. You can also tag your entries using the same tagging mechanism found everywhere else in the portal.

The Blogs portlet also supports trackbacks and pingbacks. Trackbacks are special links that let you or another site know if you or if someone else linked to a blog entry. For example, if you wanted to write an entry in your blog and reference someone else's entry, you might put the URL to the other entry in the **Trackbacks to Send** field. Similarly, if you want others who link

to your blog to let you know about the link via trackbacks, leave the **Allow Trackbacks** box checked. This will generate a URL that is displayed with your blog entry. Others who want to link to your entry can use this URL for the link, and every time the link is clicked on, your Liferay-powered site will know about it and will be able to keep track of the clicks.

Note that trackbacks only work when the protocol is supported by both the linker and the linkee. For this reason, the Blogs portlet also supports pingbacks. Pingbacks are XML-RPC requests that are similar to trackbacks except they are automatically sent when you link to another site. So if you link to another site in your blog entry, Liferay will send a pingback to that site to notify that site that you linked to it. Similarly, if someone links to your blog entry, Liferay can receive a pingback from that person's site and record the link.

Once you have finished your blog entry, click *Publish*. You'll go back to the list of entries, and now your entry is displayed. Here is what it looks like when the display style is set to *Abstract* and the number of entries is set to 10:

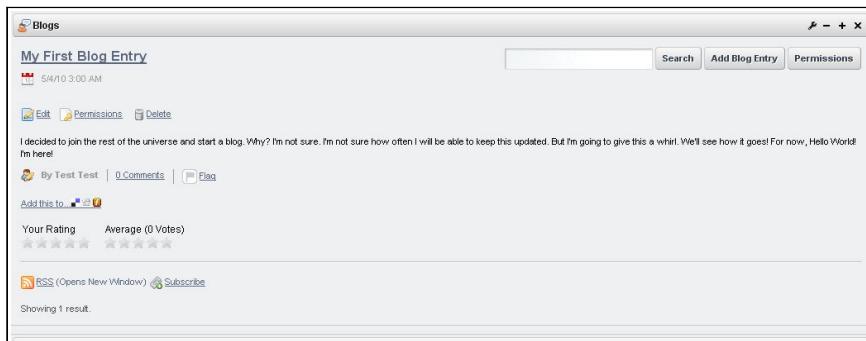


Illustration 77: First blog entry added

You can see that in the summary view, you don't see the trackback / pingback link, and you only see the number of comments which have been added. If you were to click the *Read More* link, you would see the entirety of the article, all of the comments in a threaded view, and the trackback / pingback link which others can use to link back to your blog entry.

Additionally, the full view of a blog contains convenient links to add blog entries to popular aggregating sites such as Digg, del.icio.us, and others. This gives your readers an easy way to submit your blog entries to these aggregators, potentially driving further traffic to your site.

As you can see, the Blogs portlet is a full-featured blogging application which gives you and your users the ability to enter the blogosphere with an application that supports anything a blogger needs.

Aggregating Blog Entries

You can set up a whole web site devoted just to blogging if you wish. The Blogs Aggregator portlet allows you to publish entries from multiple bloggers on one page, giving further visibility to blog entries. This portlet is also very easy and straightforward to set up. You can add it to a page from the Collaboration category in the Add → More menu in the dock.

If you click *Configuration* from the menu button in the title bar of the portlet, you will see the Blogs Aggregator's configuration page. From here, you can set several configuration options.

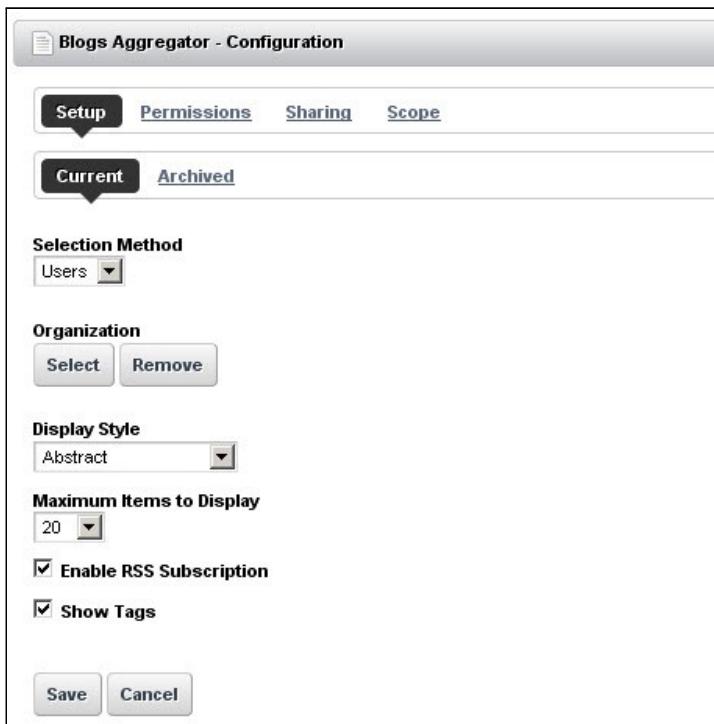


Illustration 78: Blogs Aggregator Configuration

Selection Method: You can select Users or Scope here. If you select Users, the Blogs Aggregator will aggregate the entries of every blogger on your system. If you want to refine the aggregation, you can select an Organization by which to filter the users. If you select Scope, the Blogs Aggregator will contain only entries of users who are in the current scope. This will, in essence, limit the entries to members of the Community or Organization upon which the Blogs Aggregator portlet resides.

Organization: Allows you to determine which organization's blogs you want to aggregate.

Display Style: Just like the Blogs portlet, you can select from several different styles for displaying blog entries.

Maximum Items to Display: Select maximum number of entries the portlet will display.

Enable RSS Subscription: The aggregated entries can themselves be an RSS feed. Leave this box selected if you want people to be able to subscribe to your aggregated blog entries.

Show Tags: This option will display all the tags associated with the blogs.

When you have finished setting the options in the portlet, click *Save*. Then click *Return to Full Page*.

As you will see, the Blogs Aggregator looks very much like the Blogs portlet, except that the entries come from more than one author.

Calendar

Liferay's Calendar portlet is a complete calendaring solution. You can schedule any number of events of different types, receive alarms via email or text message, import and export your calendar, and much more. Additionally, you can import and export the calendar to the popular iCalendar format for use in other applications.

In a similar way to the Blogs portlet, you can use the Calendar portlet as a shared calendar on a community or organization's web site, or you can use the Calendar portlet as a personal calendar—or both.



Illustration 79: The Liferay Calendar Portlet

Configuring the Calendar Portlet

Go to the **Configuration** option from the menu in the portlet's title bar. You have three tabs there which allow you to configure three different options: **Email From**, **Event Reminder Email**, and **Display Settings**.

Email From

In this tab, you can set the **Name** and **Email Address** that will display in system generated reminder emails. The address that you enter in the Email Address field must be properly formatted, but it does not need to be an address that actually exists. This can work well in conjunction with mail rules in your email client that can operate on these messages. By default, the name is set to Joe Bloggs and the email address is set to test@liferay.com.

Event Reminder Email

This tab lets you customize the email message that you receive from the portlet when you have an event for which you have configured a reminder. It contains the same rich text editor that you see everywhere else in Liferay, and this allows you to format your message so that you will be able to easily recognize it. Additionally, there are several variables which allow you to insert runtime values into the message, and these are listed underneath the text editor so that you can use them in the appropriate place in your template. For example, you might want the event start date and time and the event title included in the email reminder that you receive. Inserting the variables that correspond with those values into your template will allow you to do that.

Display Settings

Display Settings for the calendar allows you to define which tab in the calendar is the default when the portlet is first displayed. By default, the summary tab is displayed, but you may want the daily, weekly, or monthly view to be the default.

There are additional settings for the summary tab: you can select whether it has a horizontal or vertical layout, select whether it shows a mini month, select whether it shows today's events, or select to enable comments.

Using the Calendar Portlet

The Calendar portlet generally works the way you would expect a calendar to work. It inherits its interface from the rest of Liferay's portlet library, so you should find common tasks in the same place that you find them in other Liferay portlets.

To get started, you may want to click the *Permissions* button. Here you can find a list of roles with check boxes denoting whether the role has the *Add Event* or the *Export All Events* permission. By default, only the owner has permissions to do these things, which means that, by default, the Calendar portlet is set up to be used in one's personal space. Out of the box, Liferay defaults to putting a Calendar portlet on all users' private pages, so this is expected. If you are going to be using the Calendar as a shared calendar, you may want to modify some things here.

First, create a portal, community, or organization role and add the users to whom you wish to grant access to the calendar to the role. Then come back to the Calendar portlet and click the *Permissions* button. Check the boxes next to the role(s) that should have access to one or both of the functions (*Add Event* or *Export All Events*). Then click *Submit*.

Now you are ready to begin using your calendar. Click the *Add Event* button. You will see a form that allows you to fill out all the information for your event.

Start Date/Time: The date and time the event starts.

Duration: How long the event will last.

All Day Event: Check this box to disassociate time from the event and make it last all day.

Time Zone Sensitive: Leave this box checked to make sure that the portal keeps track of the event regardless of time zone.

Title: The title of the event.

Description: A description of the event.

Type: Select from a number of pre-configured event types. You can change these in the portal-ext.properties file.

Permissions: Determine who can view and edit the event.

Repeat: If the event repeats on a schedule, select the schedule (daily, weekly, monthly, etc.)

End Date: If the event repeats on a schedule but there is an end to the set of meetings, enter the end date.

Reminders: Select whether to send a reminder, how long before the event to send it, and through what medium (email, SMS text message, or instant message) to send it. Note that this feature is integrated with your profile on the portal, so you will need to fill out your mobile phone number and / or instant messenger IDs in order to use those features.

When you have finished adding your event, click *Save*.

You can view calendar events by day, week, month, year, or in a simple list.

Chat

Liferay's Chat portlet gives you a convenient way of allowing your users to send each other instant messages when they are logged into your web site. It appears as a bar at the bottom of every page, showing who is logged on, their statuses, and any chats the logged-in user has had open.



Illustration 80: Liferay's Chat Portlet

The Chat portlet is very simple to use. To change the settings, click *Settings* (found near the lower right corner next to *Online Friends*). Here you can set your status, whether to show if you are online, and whether to play a sound if someone sends you a message while you have the window or tab in the background.

The portlet shows you the number of your friends who are online. To chat with one of them, click the *Online Friends* link and then click the friend's name. You can then begin chatting with him or her. You can have multiple chats open at a time, and even have one or more of them minimized.

The Chat portlet is distributed with the Liferay bundles, but is not included as part of the .war distribution, as it is a separate plugin. If you installed the Liferay .war manually on your application server, you can install the Chat portlet by going to the Control Panel, clicking *Plugins Installation*, and then clicking the *Install More Portlets* button. Find the Chat portlet in the list, click on it, and then click *Install*.

Mail

Liferay's Mail portlet enables your users to interact with their email using an easy to use, ubiquitous web interface. If your mail system supports the IMAP protocol, you can use the Mail portlet to integrate your users' mail with the rest of your web site. You can also connect the Mail portlet to a mail account provided by Google.

The Mail portlet is distributed with the Liferay bundles, but is not included as part of the .war distribution, as it is a separate plugin. If you installed the Liferay .war manually on your application server, you can install the Mail portlet by going to the Control Panel, clicking *Plugins Installation*, and then clicking the *Install More Portlets* button. Find the Mail portlet in the list, click on it, and then click *Install*.



Illustration 81: Liferay Mail portlet

To connect the Mail portlet with an email account, click the *Add a New Email Account* link. From there, you are given a choice between a Custom email Account or a Gmail Account. Choose the option that you wish, and fill out the form that appears.

For a Gmail account, all you need to do is provide your email address and your password, and the portlet will take care of the rest.

For a Custom Mail Account, the following fields are necessary:

Address: The email address which receives mail for this account.

Login: The user name for logging into the account.

Password: The password for logging into the account.

Incoming Settings: The host name for your IMAP (Internet Mail Access Protocol) or POP server.

Incoming Port: The port upon which the IMAP or POP service is running.

Use Secure Incoming Connection: Check this box to use an encrypted connection to the server, if your server supports it.

Outgoing SMTP Server: The host name of your SMTP (Simple Mail Transfer Protocol) server.

Outgoing Port: The port upon which the SMTP service is running.

Use Secure Outgoing Connection: Check this box to use an encrypted connection to the server, if your server supports it.

When finished, click *Save*. Your new email account now appears as a tab at the top of the page along with the button for adding a mail account. In this way, you can add as many mail accounts as you want in order to view them in the portlet.

If you click the tab for the mail account you just configured, you will be brought to an interface which allows you to read your mail and compose new messages. To read a message, click on it. To compose a new message, click the *Compose Email* link on the left side of the portlet. You will be brought to a form which allows you to compose an email message using the same rich text editor that appears everywhere else in Liferay. You can read, reply, and create messages, as well as manage all of your folders in Liferay's Mail portlet.

The Mail portlet is a great way to integrate a familiar service with other collaboration features that Liferay provides.

Message Boards

Liferay's message boards are one of the most widely used applications provided by the product. The application is a state of the art forum application similar to many forums in which you may have participated. The difference, of course, is that Liferay's message boards can inherit the abilities of the Liferay development platform to provide an integrated experience that others cannot match.

There are countless web sites out there where it is clearly evident that there is no link whatsoever between the main site and the message boards. In some cases, users are even required to register twice: once for the main site and once for the message boards. Sometimes it is three times: for the site, for the message boards, and for the shopping cart. By providing a message boards portlet along with all of the other applications, Liferay can provide a unique, integrated approach to building your web site. All of the integration work is done for you, and you can concentrate on building out the site that you want to build.

The Message Boards portlet has a lot of configuration options, but they are straightforward to use and are the reason why this portlet is a full-featured forum application for your web site. To get started, add a Message Boards portlet to your site. Once it is added, click the *Menu* icon in the portlet's title bar and click *Configuration*. There are two rows of tabs. The first tab in the top row is titled simply, *Setup*. This is where you can configure the application the way you want it to behave on your site.

General

The first tab beneath *Setup* is labeled *General*. Here, you can enable anonymous posting, flags and ratings. All three options are selected by default.

Anonymous posting lets those without an account on the system post messages to your message boards. You may or may not want to do this, depending on the type of community you are building. Allowing anonymous posting opens your site to anyone who might want to spam your forums with unwanted or off topic advertising messages. For this reason, most of those who implement message boards turn anonymous posting off by unchecking this box.

Flags allow your users to flag content which they consider to be objectionable. If you are allowing anonymous posting, you might use flags in combination with it if you have someone administering your message boards on a day-to-day basis. That way, any unwanted messages can be flagged by your community, and you can review those flagged messages and take whatever action is necessary. Using flags is also a good practice even if you're not allowing anonymous posting.

Ratings enable your users to give certain posts a score. This score is used by Liferay Portal's social equity system to rank your community members by how helpful their contributions are. You can read more about social equity later in this chapter.

Email From

This tab allows you to configure the email address that messages from the Message Boards portlet come from. By default, the name is Joe Bloggs and the email address is test@liferay.com.

Message Added Email

This tab allows you to customize the email message that users receive when a message is added to a topic to which they are subscribed.

Enabled: Uncheck the box to disable the message added email.

Subject Prefix: Enter a prefix that will be prepended to the subject line of the email. This is usually done so that users can set up message filters to filter the notifications to a specific folder in their email clients.

Body: Enter anything here that should appear in the body of the email.

Signature: Enter anything here that should appear as part of the signature of the email.

Below the fields is a section called **Definition of Terms** which defines certain variables which you can use in the fields above to customize the email message. Some of these variables are for the message board category name, the community name, and more.

Message Updated Email

The Message Updated Email tab is identical to the Message Added Email tab, except it defines the email message that users receive whenever a topic is updated.

Thread Priorities

You can define custom priorities for message threads. These allow administrators to tag certain threads with certain priorities in order to highlight them for users. By default, three priorities are defined: Urgent, Sticky, and Announcement. To define a thread priority, enter its name, a URL to the image icon that represents it, and a priority number which denotes the order in which that priority should appear.

There is also a field on this form that allows you to select a localized language for your priorities. If you need to do this, you can select the language from the selection box.

User Ranks

Users can be ranked according to the number of messages they have posted. You can set up custom ranks here. Defaults have been provided for you, going from zero messages all the way up to 1000.

In addition to ranks, you can also select who is a “moderator” by what roles are held. Defaults are there for you which show you how to do this.

```
Moderator=community-role:Message Boards Administrator  
Moderator=organization:Message Boards Administrator  
Moderator=organization-role:Message Boards Administrator  
Moderator=regular-role:Message Boards Administrator  
Moderator=user-group:Message Boards Administrator
```

As you can see, all you need to do is set the rank, the collection type, and the name of the type. In the example above, anyone who has a Community Role, an Organization Role, a regular Role, or is in a user group called *Message Boards Administrator*, or anyone who is the Organization Owner gets the Moderator rank.

As with thread priorities, on this tab you can define whether your ranks are localized in a particular language.

RSS

Message board threads can be published as RSS feeds. This tab allows you to define how the feeds are generated.

Maximum Items to Display: Select the number of items to display in the feed.

Display Style: Select the style. You can publish the full content, an abstract, or just the title of a thread.

Format: Choose the format among RSS 1.0, RSS 2.0, or Atom 1.0 formats.

Permissions

The default page that the Message Boards portlet displays has three buttons on it. Click the one labeled *Permissions*. This allows you to define which roles have the ability to add a category of threads or to ban abusive users from the message boards. Select the roles and permissions you want to configure and then click *Submit*.

Adding Categories and Mailing Lists

You are now ready to add categories to your message boards. Click the *Add Category* button. You may merge with a Parent Category by enabling the Merge with Parent Category check box and clicking the *Select* button. Enter a name for the category and a description of the category. At the bottom of the form is a check box which allows you to enable the mailing list function.

The mailing list function works in concert with the message notification emails. If a user subscribes to a message board category, he or she will get emails when someone posts messages to that category. Enabling the mailing list function allows those users to simply reply to the notification messages in their email clients, and those replies will be posted to the thread automatically.

To enable this functionality, you will need a mail account for the category. Once you click the check box, a number of other options will appear.

Email Address: Enter the email address of the account that will receive the messages.

Next, there are two sections: *Incoming* and *Outgoing*. These define the mail settings for receiving mail and for sending mail. The *Incoming* tab has the following options:

Protocol: Select POP or IMAP.

Server Name: Enter the host name of the mail server you are using.

Server Port: Enter the port on which your mail service is running.

Use a Secure Network Connection: Check this box to use an encrypted connection if your server supports it.

User Name: The login name on the mail server.

Password: The password for the account on the server.

Read Interval (Minutes): Liferay will poll the server at this interval looking for new messages to post.

The *Outgoing* section has the following options:

Email Address: Enter the email address that messages from this category should come from. If you want your users to be able to reply to the categories using email, this should be the same address configured on the *Incoming* tab.

Use Custom Outgoing Server: If you need to use a different mail server than the one that is configured for the portal, check this box. If you check the box, a number of other options will appear.

Server Name: Enter the host name of the SMTP mail server you are using.

Server Port: Enter the port on which your mail service is running.

Use a Secure Network Connection: Check this box to use an encrypted connection if your server supports it.

User Name: Enter the login name on the mail server.

Password: Enter the password for the account on the mail server.

When finished adding your category, click *Save*. Add as many categories to your message boards as you wish.

Note that categories can have subcategories. You can add a number of top-level categories and then click on each one and add categories under that, to an unlimited level. For usability reasons, you don't want to nest your categories too deep, or your users will have trouble finding them. You can always add more categories as your message boards grow.

Using the Message Boards

Upon seeing Liferay's Message Boards portlet, your users will immediately recognize that the interface is similar to many other implementations they've seen before. Message boards are nothing new to the Internet, and many people have been using them for quite a long time. For that reason, Liferay's message boards will seem very familiar to your users.

Threads can be viewed in many ways. At the top of the portlet is a set of tabs: *Recent posts*, *My Posts*, *My Subscriptions*, and for administrative users, *Statistics* and *Banned Users*. The Recent Posts tab shows all posts from all categories by date, so you can keep up on all the most recent discussions in the message boards. The My Posts tab shows all of the posts for the user that is currently logged in. This is a convenient way to get back to a previous conversation in order to retrieve some pertinent information. The My Subscriptions tab allows a user to manage thread subscriptions. If you lose interest in a particular topic, you may want to visit this tab and unsubscribe from a thread.

For administrators, the Statistics tab shows the number of categories, the number of posts, and the number of participants in your message boards. It also has a list of who the top posters to your message boards are. The Banned Users tab shows all of the users who have been banned from posting on the message boards.

Posting New Threads

To post a new thread simply select the *Post New Thread* button. You will see a message editing form. The body field on this form is different from that of the other portlets in Liferay. The reason for this is to support *BBCode*, which is a standard form of markup used in many message board products. Before BBCode was invented, many message board products would allow users to enter HTML to format their messages. This, however, enabled attackers to insert malicious code into the message board. BBCode was invented to provide users a way of formatting their messages without allowing them to enter HTML. Similarly, Liferay supports BBCode in the message boards portlet because the other editor—which is used for the Content Management System, the Blogs portlet, and other portlets—produces HTML. This is appropriate for those other portlets, as they are only used by privileged users, but it is not appropriate for the message boards. Besides this, many users of message boards are familiar with BBCode and are used to it, and the editor that is provided for Liferay's Message Boards portlet makes it very easy to use.

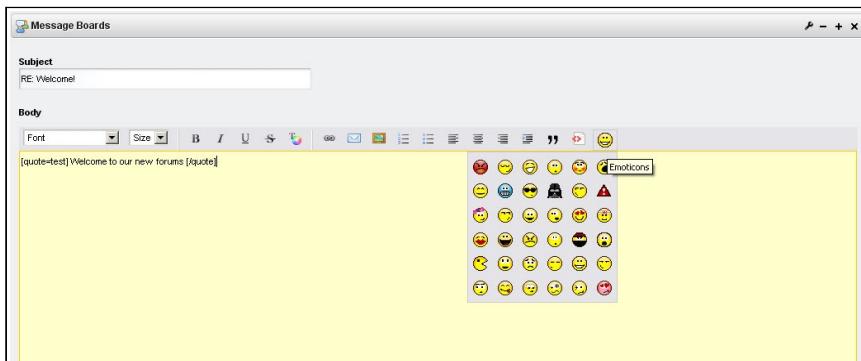


Illustration 82: Editing a message boards post. You can see the emoticons that are available in the editor.

Users who have Moderator access to the board can modify the priority of messages. You can also use the editor to quote from messages that you are replying to, to insert emoticons, to add preformatted text, and more.

Messages that are posted to the message boards are shown by default in a threaded view so that replies are attached to the proper parent message. This makes it easy to follow along with conversations.

When viewing a message board thread, users are given several options. At the top right of the thread are three icons, allowing users to view threads in a flat view, in a tree view, or in a combination view. A flat view shows all of the messages in the order in which they are posted. A tree view shows all of the messages in a threaded view, so that replies are next to the messages they are replying to. A combination view shows the threads at the top as subjects only, with the flat view underneath.

When viewing a thread, users can click links allowing them to post a new thread, subscribe to the thread they are viewing, or if they have administrative access, lock a thread or move a thread to another category. Subscribing to a thread causes Liferay to send the user an email whenever a new message is posted to the thread. If you have enabled the mailing list feature for the category in which the thread resides, users can simply reply to these messages in order to post back to the thread, without having to visit your site.

The Message Boards portlet is also highly integrated with Liferay's user management features. Posts on the message board show users' pictures if they have uploaded one for themselves, as well as the dates that users created an ID on your site.

Message Board Administrative Functions

The Message Boards portlet provides for the day to day administration of the message threads. You may wish to separate this function out by a role, and then delegate that role to one or more of your users. That would free you up to concentrate on other areas of your web site. To do this, you can create a role called Message Board Administrator. This role can be scoped by the portal, an organization, or a community. If you have a portal scoped role, members of this role will be able to administer any Message Boards portlet in the portal. If it is a community or organization scoped role, members of this role will be able to administer a Message Boards portlet in only the community or organization in which they have the role.

Go to the Control Panel and create this role. Once it is created, click *Actions → Define Permissions*. Click the *Portlet Permissions* button. Browse the list until you find the Message Boards portlet and then click on it. You will then see a screen which allows you to configure the various permissions on the portlet.

Roles

View All **Add**

Message Board Administrator [«Back](#)

Edit **Define Permissions** [Assign Members](#)

Add Permissions **Message Boards**

Action	Scope	Limit Scope
Access in Control Panel	Portal	Limit Scope
Add to Page	Portal	Limit Scope
Configuration	Portal	Limit Scope
View	Portal	Limit Scope

Save **Cancel**

Illustration 83: Configuring Message Board Administrators role

Grant the permissions you wish message board administrators to have and then click *Save*. You can then add users to this role and they will inherit the permissions.

Message Board administrators can perform all of the functions we have already presented, including creating and deleting categories and posting threads. In addition to these, a number of other functions are available.

Moving Threads

Many times a user will post a thread in the wrong category. Administrators may in this case want to move a thread to the proper category. This is very easy to do. You can select the *Action* menu to the right of the thread and then select *Move Thread*. Or, if you are already viewing the thread and you have administrative access, there is a link at the top of the thread labeled *Move Thread*. Click this link. You will be presented with a simple form which allows you to select a category to which to move the thread and a check box which allows you to post a message explaining why the thread was moved. This message will be posted as a reply to the thread you are moving. When finished, click the *Move Thread* button and the thread will be moved.

Deleting Threads

Users with administrative access to the message boards can delete threads. Sometimes users begin discussing topics that are inappropriate or which reveal information which should not be revealed. In this case, you can

simply delete the thread from the message boards. This is easy to do. First, view the list of threads. Next to every thread is an *Actions* button. Click *Actions* → *Delete* to delete the thread. This does not prevent users from re-posting the information, so you may need to be vigilant in deleting threads or consider the next option.

Banning Users

Unfortunately, sometimes certain users can become abusive. If you wind up with a user like this, you can certainly make attempts to warn him or her that the behavior he or she is displaying is unacceptable. If this does not work, you can ban the user from posting on the message boards.

Again, this is very easy to do. Find any post which was written by the abusive user. Underneath the user's name / profile picture is a link called *Ban this User*. Click this link to ban the user from the message boards.

If after taking this action the user apologizes and agrees to stop his or her abusive behavior, you can choose to reinstate the user. To do this, click the *Banned Users* tab at the top of the Message Boards portlet. This will show a list of all banned users. Find the user in the list and click the *Unban this User* link.

Splitting Threads

Sometimes a thread will go on for a while and the discussion completely changes into something else. In this case, you can split the thread where the discussion diverges and create a whole new thread for the new topic. Administrative users will see a *Split Thread* link on each post. To split the thread, click the link. You will be brought to a form which allows you to add an explanation post to the split thread. Click *Ok* to split the thread.

Editing Posts

Administrative users can edit not only their own posts, but also everyone else's. Sometimes users will post links to copyrighted material or unsuitable pictures. You can edit these posts, which allows you to redact information that should not be posted or to censor profanity that is not allowed on your message boards.

Permissions

Permissions can be set not only on threads, but also on individual posts. You can choose to limit a particular conversation or a post to only a select group of people. To do this, click the *Permissions* link on the post and then select among the *Delete*, *Permissions*, *Subscribe*, *Update*, and *View* permissions for the particular role to which you want to grant particular access.

This function can be used to make it so some privileged users can post on a certain thread, but others are allowed to view it, or any combination of the above permissions.

Wikis

Liferay's Wiki portlet, like the Message Boards portlet, is a full-featured wiki application which has all of the features you would expect in a state of the art wiki. Again, though, it has the benefit of being able to take advantage of all of the features of the Liferay platform. As such, it is completely integrated with Liferay's user management, tagging, and security features.

So what is a wiki? Put simply, a wiki is an application which allows users to collaborate on information. This, of course, has many applications—the most famous of which is Wikipedia, which is a full encyclopedia developed collaboratively by users from all over the world, using a wiki. Another example would be Liferay's wiki, which is used for collaborative documentation for the Standard Edition of the product.

A wiki application allows users to create and edit documents and link them to each other. To accomplish this, a special form of markup is used which is sometimes called wikitext. Unfortunately, the proliferation of many different wiki applications resulted in slightly different syntax for wikitext in the various products, as each new wiki tried to focus on new features that other wikis did not have. For that reason, a project called WikiCreole was started. This project resulted in the release of WikiCreole 1.0 in 2007, which is an attempt to define a standard wiki markup that all wikis can support.

Rather than define another wikitext syntax, Liferay's Wiki portlet supports WikiCreole as its syntax. This syntax is a best-of-breed wiki syntax and should be familiar to users of other wikis. The portlet provides a handy cheat sheet for the syntax on the page editing form, with a link to the full documentation if you wish to use some of WikiCreole's advanced features.

Getting Started with the Liferay Wiki

The Wiki portlet works just like the other portlets developed by Liferay. Add the portlet to a page using the *Add → More* menu and then click *Configuration* in the portlet menu in the Wiki portlet's title bar. You'll see some op-

tions are likely to be familiar to you by now such as sharing the application with websites, Facebook, Google Gadgets, etc.

You will also notice that the Communication tab has some additional options not seen in the other portlets.

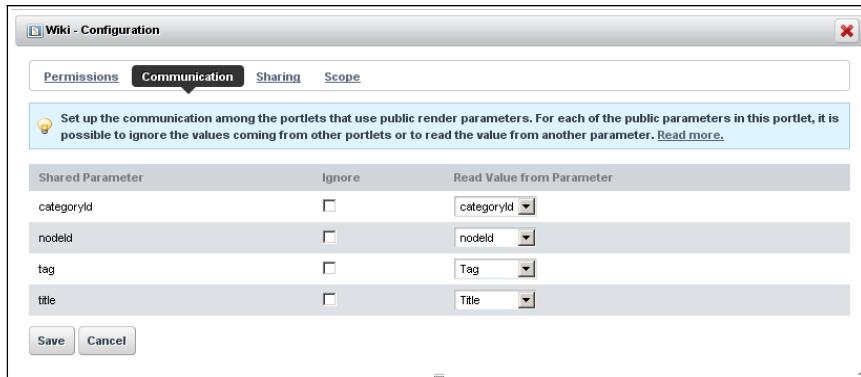


Illustration 84: Communication tab of Wiki portlet

The Communication tab allows you to configure communication across portlets, using predefined public render parameters. From here you can modify four public render parameters: nodeId, title, categoryId and tag. For each parameter you can select:

- Ignore the values for this parameter that come from other portlets. For example, the wiki portlet can be used along with the Tags Navigation portlet. When a user clicks on a tag in the Tags Navigation portlet, the wiki it shows the list of pages with that tag. In some cases an administrator may want the wiki portlet to always show the front page independently of any tag navigation done through other portlets. This can be achieved by checking the ignore check box so that the values of the parameter coming from those other portlets are ignored.
- Read the value of a parameter from another portlet. This is an advanced but very powerful option that allows portlets to communicate without configuring it beforehand. For example, imagine that the wiki portlet is used to publish information about certain countries. Another custom portlet that allows browsing countries for administrative reasons was written and placed on the same page. That second portlet has a public render parameter called country with the name of the country. Using this procedure, we can cause the wiki to show the information from the country being browsed through the other portlet. You can do this here for the wiki by setting the value for the title parameter to be read from the country parameter of the other portlet.

Once you have set the options the way you want them, click *Save*.

Managing Wikis

The Wiki portlet can contain many wikis. By default, it contains only one, called *Main*. To manage Wikis, navigate to the *Control Panel* and select the *Wiki* tab. You will then be brought to a screen that allows you to add, modify, and delete wikis. You will see that the *Main* wiki has already been added for you.

At the top of this screen is a *Permissions* button. Clicking this allows you to define what roles have access to create wikis. If you have created a specific role for creating wikis, you can click the box in the *Add Node* column and then click *Submit*, and that role will have access to create new wikis in this portlet.

Clicking the *Add Wiki* button brings you to a screen which allows you to give the wiki a name and a description. You can also set up some default permissions. When you create a new wiki, it will appear in a list at the top of the main page of the portlet.

Next to each wiki in the list of wiki nodes is an *Actions* button. This button contains several options:

Edit: Lets you edit the name and description of the wiki.

Permissions: Lets you define what roles can add attachments to wiki pages, add pages to the wiki, delete pages, import pages to the wiki, set permissions on the wiki, subscribe to the wiki, update existing pages, and view the wiki.

Import Pages: You can import your data from other wikis. This allows you to migrate off of another wiki which you may be using and use the Liferay wiki instead. You may wish to do this if you are migrating your site from a set of disparate applications (i.e. a separate forum, a separate wiki, a separate content management system) to Liferay, which provides all of these features.

Currently, MediaWiki is the only wiki that is supported, but others are likely to be supported in the future.

Subscribe: A user can subscribe to a wiki node, and any time a page is added or updated Liferay will send an email to the user informing him or her what happened.

Delete: Deletes the wiki node.

To go back to your wiki, click on its name in the list of wikis.

Note that there is also a wrench icon leading to a configuration menu on this portlet in the control panel. This contains several other options which you may have seen on other portlets.

Email From, Page Added Email, and Page Updated Email are similar to the notification email settings for other portlets, allowing you to customize who wiki emails come from and the format and text of the email that is sent when a page is added or updated.

The *Display Settings* tab gives you several options for how the wiki should be displayed. **Enable Page Ratings, Enable Comments, and Enable Comment Ratings** are similar to the same options in other portlets. They give you the ability to set how you want users to interact with wiki documents: a little, a lot, or not at all.

Below this, you can set which wikis are visible in the Wiki portlet by default and which are hidden. You might host two wikis in any given community, exposing one to the public and keeping one of them private for community members.

Finally, the Wiki portlet also supports RSS feeds as the other collaboration portlets do, and you can configure its options here.

Adding and Editing Wiki Pages

By default, there is one page added to your wiki, called *FrontPage*. To get started adding data to your wiki, click the *Edit* link at the top right of the portlet. You will be brought to a blank editing page.

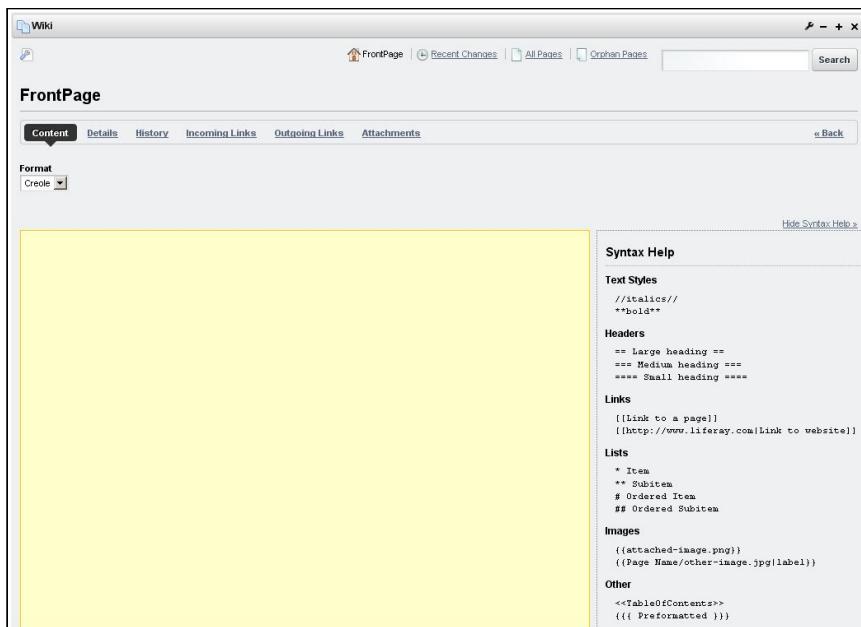


Illustration 85: Editing the default page in the wiki portlet

You can now begin to add content to the page. Notice that there is a very convenient “cheat sheet” which can help with the wiki syntax. You can use this syntax to format your wiki pages. Consider for example the following wiki document:

```
== Welcome to Our Wiki! ==
```

This is our new wiki, which should allow us to collaborate on documentation. Feel free to add pages showing people how to do stuff. Below are links to some sections that have already been added.

```
[[Introduction]]
```

```
[[Getting Started]]
```

```
[[Configuration]]
```

```
[[Development]]
```

```
[[Support]]
```

```
[[Community]]
```

This would produce the following wiki page:

The screenshot shows a Liferay wiki page titled "FrontPage". The main content area contains the text "Welcome to Our Wiki!". Below this, a paragraph of text is displayed, followed by a list of red hyperlinks: "Introduction", "Getting Started", "Configuration", "Development", and "Support". At the bottom of the page, there are buttons for "Add Child Page", "Attachments", and "Views". Below these buttons, there is a rating section with a star icon and the text "Average (0 Votes)". A note at the bottom states "No comments yet. Be the first."

Illustration 86: Wiki text added to front page

This adds a simple heading, a paragraph of text, and several links to the page. Notice that the links are red, instead of the normal blue color in the default Liferay theme. This indicates that the page behind that link does not yet exist, and therefore needs to be created. If you click one of those links, you will be brought immediately to the editing screen you were on previously when you edited the front page, except this time you will be creating the page behind the link you just clicked. Liferay will display a notice at the top of the page stating that the page does not exist yet, and that you are creating it right now. As you can see, it is very easy to create wiki pages. All you have to do is create a link from an existing page.

Note that at the top of the screen you can select from the Creole wiki format and the HTML editor that comes with Liferay. We recommend that

you stick with the Creole format, as it allows for a much cleaner separation of content and code. If you want all of your users to use the Creole format, you can disable the HTML format using the `portal-ext.properties` file. See the next chapter for further information on how to configure this.

At the bottom of the page editing screen, you can select *Categories* for the article. Categories are a hierarchical list of headings under which you can create wiki pages. This allows you to organize your content in a more formal fashion. You can create categories using the Control Panel, in the *Tags and Categories* section.

Page Details

When viewing a page, you can view its details by clicking the *Details* link which appears in the top right of the page. This allows you to view many properties of the page. There are several tabs which organize all of the details into convenient categories.

Details

The Details tab shows various statistics about the page, and also contains a few actions that you can perform on the page.

Title: Displays the title of the page.

Format: Displays the format for the page—either Creole or HTML.

Latest Version: Displays the latest version of the page. The wiki portlet automatically keeps track of page versions whenever a page has been edited.

Created By: Displays the user who created the page.

Last Changed By: Displays the user who last modified the page.

Attachments: Displays the number of attachments to the page.

RSS Subscription: Displays links which allow you to subscribe to the page as an RSS feed in three formats: RSS 1.0, RSS 2.0, and Atom 1.0.

Email Subscription: Contains links allowing you to subscribe to the entire wiki or just to this page.

Advanced Actions: Contains links allowing you to modify the permissions on the page, make a copy of the page, move (rename) the page, or delete the page.

History

This tab shows a list of all of the versions of the wiki page since it was created. You can revert a page back to a previous state and you can also com-

pare the differences between versions by selecting the versions and then clicking the *Compare Versions* button.

Incoming / Outgoing Links

The next two tabs are for incoming and outgoing links. These are wiki links to and from the page. You can use this tab to examine how this page links to other pages and how other pages link back to this page.

Attachments

The last tab is for attachments. You can attach any file to the wiki. This is mostly used to attach images to wiki articles which can then be referenced in the text. Referencing them using the proper WikiCreole syntax renders the image inline, which is a nice way to include illustrations in your wiki documents.

Navigating in the Wiki Portlet

At the top of the portlet is a list of links which allow you to navigate around the wiki. Next to the *Manage Wikis* button is a list of wikis that are currently created in the portlet. Simply click on the wiki's name to begin browsing that wiki. After this is a set of navigation links:

FrontPage: Takes you to the main page of the main wiki.

Recent Changes: Takes you to a page which shows all of the recently updated pages.

All Pages: Takes you to a flat, alphabetical list of all pages currently stored in the wiki.

Orphan Pages: This link takes you to a list of pages that have no links to them. This can happen if you take a link out of a wiki page in an edit without realizing it's the only link to a certain page. This area allows you to review wiki pages that are orphaned in this way so that you can re-link to them or delete them from the wiki if they are no longer relevant.

Search: Enter a term here and click the *Search* button to search for items in the wiki. If the search term is not found, a link will be displayed which allows you to create a new wiki page on the topic for which you searched. In general, categories start off with broad terms and eventually

Tags

Tags are an important tool that you can use to help organize information on your portal and make it easier for your users to find content that they're looking for. Tags are words or phrases that you can attach to any content on

the website. Tagging content will make your search results more accurate, and enable you to use tools like the Asset Publisher to display content in an organized fashion on a web page. There are two ways to create tags: you can do it through the administrative console in the control panel, or on the fly as content is created.

To create tags in the control panel, select the community that you want to create tags for, and select *Tags*. From this screen, you will be able to view any existing tags and make new ones. To create a new tag, simply click *Add Tag*. You then be asked for the name of the tag, and you'll also have the ability to set permissions for viewing or managing the tag.

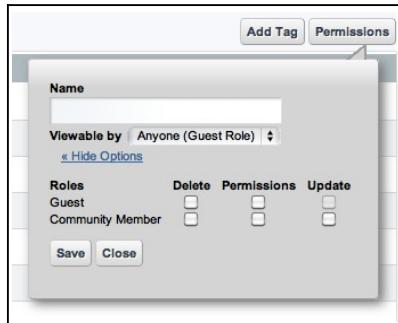


Illustration 87: The Add Tag dialog

From the *Tags* screen on the control panel, you can also edit existing tags. You can change the tag name, change the tag's permissions, delete the tag, or add *Properties*. Properties are essentially tags for your tags; you can use them to add additional information about your tags.

Categories

Categories are similar in concept to tags, but are designed with the administrator in mind, not the end user. Hierarchies of categories can be created, and categories can be grouped together in *Vocabularies*. Where tags represent an ad hoc method for users to group content together, categories exist to allow administrators to organize content in a more official, hierarchical structure. As has been said in Chapter 3, think of tags like the index to a book and categories like the table of contents. Both serve the same purpose: to help the user find the information he or she seeks.

Adding vocabularies and categories is similar to adding tags. Once you've selected the community or organization you want to work on, select *Categories* from the content section of the control panel, and you will be presented with the categories administration screen.

The screenshot shows the 'Categories' administration screen in Liferay. On the left, there's a sidebar titled 'Vocabularies' with a single entry 'Topic'. To the right, under 'Categories', there's a list of items: 'general', 'welcome', 'software', 'productivity', and 'games'. Each item has a small triangle icon before its name, indicating it can be expanded or collapsed. At the top of the main area are search and filter fields, and buttons for 'Add Vocabulary', 'Add Category', and 'Permissions'. At the bottom are buttons for 'Delete Vocabulary' and 'Edit Vocabulary Permissions'.

Illustration 88: Categories administration

Clicking on a vocabulary on the left will display any categories that have been created under that vocabulary. You can create new vocabularies simply by clicking *Add Vocabulary* and providing a name for it. You can create categories in a similar fashion by choosing a vocabulary on the left, and then selecting *Add Category*. Like tags, you can also provide properties for categories.

Once you have created some vocabularies and categories, you can take advantage of the full capabilities of categories by creating a nested hierarchy of categories. To nest categories, select what you want to be the parent category, then drag any category that you want to become a child category onto it. You will see a plus sign appear next to the name of the category you are dragging if you can add it to the selected parent category; if you see a red x that means that you cannot add that category as a subcategory of parent category that you have selected.

This screenshot shows a zoomed-in view of the category structure under the 'Topic' vocabulary. The 'Topic' vocabulary is highlighted. Below it, the 'general' category is expanded, showing its sub-categories: 'welcome', 'software', and 'prod'. The 'prod' category has a green plus sign icon next to it, indicating it can be added as a subcategory of 'software'. The 'games' category is also visible under 'software'.

Illustration 89: Adding a subcategory

Once you have created a hierarchy of categories, your content creators will have them available to apply to content that they create. Under the *Categorization* section of the content creation screen, click the button labeled *Select* under the *Categories* heading. A dialog will appear with your vocabularies

and categories. Select any relevant categories by checking the box next to them, and they will be applied to the content.

Social Equity

When you have a lot of user interaction on your web site, sometimes it is helpful to try to separate the signal from the noise. Liferay contains a lot of applications which end users can make use of to communicate with each other and provide information. Some of this information is good and helpful and some of it can be rather unhelpful. In order to better show which users are making real, valuable contributions, Liferay is introducing the new Social Equity system with Liferay 6.

Social Equity enables you to assign values and weight for each contribution that a user makes. Points are given to users for *Information*, and *Participation*. This way, a user who writes several blogs and wiki articles can be shown as having a higher rank than a user who has only written comments

Wiki Page						
Name	Information Value	Information Lifespan	Daily Limit	Participation Value	Participation Lifespan	Daily Limit
View	1	365	0	1	365	0
Add Page	10	365	0	5	365	0
Add Discussion	2	365	0	2	365	0

Showing 3 results.

Message Boards Message						
Name	Information Value	Information Lifespan	Daily Limit	Participation Value	Participation Lifespan	Daily Limit
Reply to Message	0	0	0	10	30	0
Add Message	0	0	0	20	30	0
View	0	0	0	1	30	0
Add Vote	10	365	0	0	0	0

Showing 4 results.

Blogs Entry						
Name	Information Value	Information Lifespan	Daily Limit	Participation Value	Participation Lifespan	Daily Limit
View	1	365	0	1	365	0
Add Entry	10	365	0	5	365	0
Add Discussion	2	365	0	2	365	0

Showing 3 results.

Save

Illustration 90: The Social Equity control panel

and message board replies. The purpose of this is to better indicate which users on a web site are knowledgeable contributors, and to reward those users.

Currently Social Equity is available for the Blogs, Message Boards, and Wiki applications. Each category contains a rating for *Information Value*, *Information Lifespan*, *Daily Limit*, *Participation Value*, *Participation Lifespan*, and *Daily Limit*.

Information Value: Indicates how many points a user receives for the information content of a specific action.

Information Lifespan: Indicates, in number of days, how long it will be before the information points expire. Setting it to 0 means that they will never expire.

Daily Limit: This is the maximum number of actions of this type that a user will receive information points for in a given day.

Participation Value: Indicates how many points a user receives for the participating in the content creation of the website.

Participation Lifespan: Indicates, in number of days, how long it will be before the participation points expire. Setting it to 0 means that they will never expire.

Daily Limit: This is the maximum number of actions of this type that a user will receive participation points for in a given day.

The actions available that you can provide credit for are wide ranging. Users can receive credit for everything from writing wiki pages to simply viewing content. You can also easily tweak these numbers in the control panel if it becomes clear that certain activities are weighted too high or too low.

If you have a portal that is heavily driven by community created content, Social Equity is an invaluable tool to recognize users who are major contributors, as well as to indicate to new users whose advice is going to be the most sound and helpful. It's easy to set up, and can be configured differently for each community and organization on your portal, giving you increased flexibility across your web site as a whole.

Summary

We have together explored many of the portlets in Liferay's collaboration suite. You have seen how you can configure all of the portlets in a similar fashion using a unified user interface. After this, we went over all of the portlets in succession.

The Blogs and Blogs Aggregation portlets can be used to manage shared blogs or blogs belonging to a group of people at once. These portlets have all the features you would want in a blog, including rich text editing, links to news aggregators, tags, RSS feeds, and more.

The Calendar portlet likewise can be used to manage a shared calendar or a group calendar. It includes features for events, event notification, repeatable events, and import and export to and from the standard iCalendar format.

Integrating mail with your portal is easy with the Mail portlet. You can add as many custom or Gmail mail accounts as you wish, and this portlet can keep them all organized in one place, together with the rest of the things Liferay is aggregating for you.

Discussion becomes easy with Liferay's Message Boards portlet. This portlet can be used to manage heavily trafficked discussion forums with ease. It inherits all of the security features of the Liferay platform and includes administrative functions for thread priorities, moving threads, nested discussion categories, banning users, and more.

Liferay's Wiki portlet is a state of the art wiki application that users can make use of to collaborate on web pages. Again, it inherits the strengths of the Liferay platform in the form of security, interface, and search. You can use the wiki portlet to manage several wiki nodes or use many wiki portlets to manage one node each.

Tying all of these applications together are tags and categories. Tags can be added to any content by end users, and hierarchies of categories can be added to the system by administrators to be applied to content anywhere in the portal. These help your users to find the content that is most relevant to them, empowering the search to be as accurate as possible.

Liferay's collaboration platform is a full suite of integrated applications that empower users to work together. You can use them to great effect to enhance your web site and to build a vibrant, active community.

6. ADVANCED LIFERAY CONFIGURATION

Liferay is configured by a combination of settings which are stored in the database (configured by the use of the Control Panel) and settings which are stored in properties (text) files. These files can be modified to change Liferay's behavior in certain ways. There are a large number of configuration options that can be set, and so this chapter will have a wide-ranging set of topics. We will first go over the main configuration file, which is stored in the Liferay Home directory, and is called `portal-ext.properties`.

There are also some other settings that you may want to further customize. They include changing certain out-of-box defaults, security configuration, adding features to Liferay through plugin management, and accessing Liferay's web services. We will examine specifically these topics:

- *Advanced Liferay Configuration:* This includes the customization of the `portal-ext.properties` file.
- *Plugin Management:* You will learn how to install Plugins (portlets and themes) from Liferay's Official Repository and Liferay's Community Repository, as well as how to create your own plugin repository.
- *Liferay SOA:* Accessing Liferay services remotely, from outside the portal, will be discussed, as well as how to configure the security settings for these services.

The portal-ext.properties File

Liferay's properties files differ from the configuration files of most other products in that changing the default configuration file is discouraged. In fact, the file that contains all of the defaults is stored inside of a .jar file, making it more difficult to customize. Why is it set up this way? Because Liferay uses the concept of *overriding* the defaults in a separate file, rather than going in and customizing the default configuration file. You put just the settings you want to customize in your own configuration file, and then the configuration file for your portal is uncluttered and contains only the settings you need. This makes it far easier to determine whether a particular setting has been customized, and it makes the settings more portable across different installations of Liferay.

The default configuration file is called `portal.properties`, and it resides inside of the `portal-impl.jar` file. This .jar file is located in Liferay Portal's `WEB-INF/lib` folder. The file which is used to override the configuration is `portal-ext.properties`. This file can be created in your Liferay Home folder (please see Chapter 2: Initial Setup for the location of this folder for your application server). By default, the file does not exist at all, unless you are running an older version of Liferay. What follows is a brief description of the options that can be placed there, thus overriding the defaults from the `portal.properties` file. These are presented in a logical order, not an alphabetical one, as many properties relate to other properties in the system.

Properties Override

This property specifies where to get the overridden properties. By default, it is `portal-ext.properties`. Updates should not be made on the original file (`portal.properties`) but on the overridden version of this file. Furthermore, each portal instance can have its own overridden property file following the convention `portal-companyid.properties`.

For example, one read order may be: `portal.properties`, then `portal-ext.properties`, and then `portal-test.properties`.

Examples:

```
include-and-override=portal-ext.properties  
include-and-override=${liferay.home}/portal-ext.properties
```

You can add additional property files that overwrite the default values by using the `external-properties` system property.

A common use case is to keep legacy property values when upgrading to newer versions of Liferay. For example:

```
java ... -Dexternal-properties=portal-legacy-5.1.properties  
include-and-override=${external-properties}
```

Liferay Home

Specify the Liferay home directory.

```
liferay.home=${resource.repositories.root}
```

This property is available for backwards compatibility. Please set the property `liferay.home` instead.

```
resource.repositories.root=${default.liferay.home}
```

Portal Context

This specifies the path of the portal servlet context. This is needed because `javax.servlet.ServletContext` does not have access to the context path until Java EE 5.

Set this property if you deploy the portal to another path besides root.

Examples:

```
portal.ctx=/  
portal.ctx=/portal
```

Resource Repositories Root

Specifies the default root path for various repository and resource paths. Under this path several directories will be created for the hot deploy feature, JCR, etc.

Examples:

```
resource.repositories.root=${user.home}/liferay  
resource.repositories.root=/home/liferay
```

Technology Compatibility Kit

Set the following to true to enable programmatic configuration to let the Portlet TCK obtain a URL for each test. This should never be set to true unless you are running the TCK tests.

```
tck.url=false
```

Schema

Set this to true to automatically create tables and populate with default data if the database is empty.

```
schema.run.enabled=true
```

Set this to true to populate with the minimal amount of data. Set this to false to populate with a larger amount of sample data.

```
schema.run.minimal=true
```

Upgrade

Input a list of comma delimited class names that implement com.liferay.portal.upgrade.UpgradeProcess. These classes will run on startup to upgrade older data to match with the latest version.

```
upgrade.processes=\n    com.liferay.portal.upgrade.UpgradeProcess_4_3_0,\n    com.liferay.portal.upgrade.UpgradeProcess_4_3_1,\n    com.liferay.portal.upgrade.UpgradeProcess_4_3_2,\n    com.liferay.portal.upgrade.UpgradeProcess_4_3_3,\n    com.liferay.portal.upgrade.UpgradeProcess_4_3_4,\n    com.liferay.portal.upgrade.UpgradeProcess_4_3_5,\n    com.liferay.portal.upgrade.UpgradeProcess_4_4_0,\n    com.liferay.portal.upgrade.UpgradeProcess_5_0_0,\n    com.liferay.portal.upgrade.UpgradeProcess_5_1_0,\n    com.liferay.portal.upgrade.UpgradeProcess_5_1_2,\n    com.liferay.portal.upgrade.UpgradeProcess_5_2_0,\n    com.liferay.portal.upgrade.UpgradeProcess_5_2_1,\n    com.liferay.portal.upgrade.UpgradeProcess_5_2_2
```

Verify

Input a list of comma delimited class names that implement com.liferay.portal.integrity.VerifyProcess. These classes will run on startup to verify and fix any integrity problems found in the database.

```
verify.processes=com.liferay.portal.verify.VerifyProcessSuite
```

Specify the frequency for verifying the integrity of the database.

Constants in VerifyProcess:

```
public static final int ALWAYS = -1;\npublic static final int NEVER = 0;\npublic static final int ONCE = 1;
```

```
verify.frequency=1
```

Auto Deploy

Input a list of comma delimited class names that implement com.liferay.portal.kernel.deploy.auto.AutoDeployListener. These classes are used to process the auto deployment of WARS.

```
auto.deploy.listeners=\n    com.liferay.portal.deploy.auto.HookAutoDeployListener,\n    com.liferay.portal.deploy.auto.LayoutTemplateAutoDeployListener,\n
```

```
com.liferay.portal.deploy.auto.PortletAutoDeployListener,\ncom.liferay.portal.deploy.auto.ThemeAutoDeployListener,\ncom.liferay.portal.deploy.auto.WebAutoDeployListener,\ncom.liferay.portal.deploy.auto.exploded.tomcat.LayoutTemplateExplodedTomca\ntListener,\ncom.liferay.portal.deploy.auto.exploded.tomcat.PortletExplodedTomcatListene\nr,\ncom.liferay.portal.deploy.auto.exploded.tomcat.ThemeExplodedTomcatListener
```

Set the following to true to enable auto deploy of layout templates, portlets, and themes.

```
auto.deploy.enabled=true
```

Set the directory to scan for layout templates, portlets, and themes to auto deploy.

```
auto.deploy.deploy.dir=${liferay.home}/deploy
```

Set the directory where auto deployed WARs are copied to. The application server or servlet container must know to listen on that directory.

Different containers have different hot deploy paths. For example, Tomcat listens on \${catalina.base}/webapps whereas JBoss listens on \${jboss.server.home.dir}/deploy. Set a blank directory to automatically use the application server specific directory.

Examples:

```
auto.deploy.dest.dir=\nauto.deploy.default.dest.dir=../webapps\nauto.deploy.geronimo.dest.dir=${org.apache.geronimo.base.dir}/deploy\nauto.deploy.geronimo-jetty.dest.dir=${org.apache.geronimo.base.dir}/deploy\nauto.deploy.geronimo-tomcat.dest.dir=${org.apache.geronimo.base.dir}/deploy\nauto.deploy.glassfish.dest.dir=${com.sun.aas.instanceRoot}/autodeploy\nauto.deploy.glassfish-tomcat.dest.dir=${com.sun.aas.instanceRoot}/autodeploy\nauto.deploy.jboss-tomcat.dest.dir=${jboss.server.home.dir}/deploy\nauto.deploy.jetty.dest.dir=${jetty.home}/webapps\nauto.deploy.jonas-jetty.dest.dir=${jonas.base}/webapps/autoload\nauto.deploy.jonas-tomcat.dest.dir=${jonas.base}/webapps/autoload\nauto.deploy.resin.dest.dir=${resin.home}/webapps\nauto.deploy.tomcat.dest.dir=${catalina.base}/webapps\nauto.deploy.weblogic.dest.dir=${env.DOMAIN_HOME}/autodeploy
```

Set the interval in milliseconds on how often to scan the directory for changes.

```
auto.deploy.interval=10000
```

Set the number of attempts to deploy a file before blacklisting it.

```
auto.deploy.blacklist.threshold=10
```

Set the following to true if deployed WARs are unpacked. Set this to false if your application server has concurrency issues with deploying large WARs.

```
auto.deploy.unpack.war=true
```

Set the following to true if you want the deployer to rename portlet.xml to portlet-custom.xml. This is only needed when deploying the portal on WebSphere 6.1.x with a version before 6.1.0.7 because WebSphere's portlet container will try to process a portlet at the same time that Liferay is trying to process a portlet.

Note that according to IBM, on versions *after* 6.1.0.9, you need to add a context parameter to the web.xml descriptor in your portlet application called com.ibm.websphere.portletcontainer.PortletDeploymentEnabled and set it to false. This parameter causes WebSphere's built-in portlet container to ignore your portlet application when it is deployed, enabling Liferay to pick it up.

```
auto.deploy.custom.portlet.xml=false
```

Set this to 1 if you are using JBoss' PrefixDeploymentSorter. This will append a 1 in front of your WAR name. For example, if you are deploying a portlet called test-portlet.war, it will deploy it to 1test-portlet.war. JBoss now knows to load this portlet after the other WARs have loaded; however, it will remove the 1 from the context path.

Modify /server/default/conf/jboss-service.xml.

See org.jboss.deployment.scanner.PrefixDeploymentSorter.

```
auto.deploy.jboss.prefix=1
```

Set the path to Tomcat's configuration directory. This property is used to auto deploy exploded WARs. Tomcat context XML files found in the auto deploy directory will be copied to Tomcat's configuration directory. The context XML file must have a docBase attribute that points to a valid WAR directory.

```
auto.deploy.tomcat.conf.dir=../conf/Catalina/localhost
```

Set the path to Tomcat's global class loader. This property is only used by Tomcat in a standalone environment.

```
auto.deploy.tomcat.lib.dir=../common/lib/ext
```

Set the URLs of Libraries that might be needed to download during the auto deploy process

```
library.download.url.quercus.jar=http://lportal.svn.sourceforge.net/viewvc/*checkout*/lportal/portal/trunk/lib/development/quercus.jar  
library.download.url.resin-util.jar=http://lportal.svn.sourceforge.net/viewvc/*checkout*/lportal/portal/trunk/lib/development/resin-util.jar
```

```
library.download.url.script-
10.jar=http://lportal.svn.sourceforge.net/viewvc/*checkout*/lportal/portal/t
runk/lib/development/script-10.jar
```

Set the Glassfish settings to enable JSR 88 application deployment.

```
auto.deploy.glassfish-tomcat.jee.deployment.enabled=false
auto.deploy.glassfish-
tomcat.jee.dm.id=deployer:Sun:AppServer::localhost:4848
auto.deploy.glassfish-tomcat.jee.dm.user=admin
auto.deploy.glassfish-tomcat.jee.dm.passwd=adminadmin
auto.deploy.glassfish-
tomcat.jee.df.classname=com.sun.enterprise.deployapi.SunDeploymentFactory
```

Hot Deploy

Input a list of comma delimited class names that implement `com.liferay.portal.kernel.deploy.hot.HotDeployListener`. These classes are used to process the deployment and undeployment of WARs at runtime.

Note: `PluginPackageHotDeployListener` must always be first.

```
hot.deploy.listeners=\
    com.liferay.portal.deploy.hot.PluginPackageHotDeployListener,\ \
    com.liferay.portal.deploy.hot.ExtHotDeployListener,\ \
    com.liferay.portal.deploy.hot.HookHotDeployListener,\ \
    com.liferay.portal.deploy.hot.LayoutTemplateHotDeployListener,\ \
    com.liferay.portal.deploy.hot.PortletHotDeployListener,\ \
    com.liferay.portal.deploy.hot.ThemeHotDeployListener,\ \
    com.liferay.portal.deploy.hot.ThemeLoaderHotDeployListener,\ \
    com.liferay.portal.deploy.hot.MessagingHotDeployListener
```

Hot Undeploy

Set the following to true to enable undeploying plugins.

```
hot.undeploy.enabled=true
```

Set the undeploy interval in milliseconds on how long to wait for the undeploy process to finish.

```
hot.undeploy.interval=0
```

Set the following to true to undeploy a plugin before deploying a new version. This property will only be used if the property `hot.undeploy.enabled` is set to true.

```
hot.undeploy.on.redeploy=false
```

Sandbox Deploy

Input a list of comma delimited class names that implement com.liferay.portal.kernel.deploy.sandbox.SandboxDeployListener. These classes are used to process sandbox style plugins.

```
sandbox.deploy.listeners=\com.liferay.portal.deploy.sandbox.ThemeSandboxDeployListener
```

Set this to true to enable sandbox style plugin development.

```
sandbox.deploy.enabled=false
```

Set the directory to scan for sand box style plugins.

```
sandbox.deploy.dir=${liferay.home}/sandbox
```

Set the interval in milliseconds on how often to scan the directory for

```
sandbox.deploy.interval=10000
```

Plugin

Input a list of comma delimited supported plugin types.

```
plugin.types=portlet,theme,layout-template,hook,web
```

Input a list of Liferay plugin repositories separated by \n characters.

```
plugin.repositories.trusted=http://plugins.liferay.com/official  
plugin.repositories.untrusted=http://plugins.liferay.com/community
```

Set this property to false to avoid receiving on screen notifications when there is a new version of an installed plugin.

```
plugin.notifications.enabled=true
```

Input a list of plugin packages ids separated by \n characters. Administrators won't be notified when a new version of these plugins are available. The ids are of the form groupId/artifactId. You can also end the id with an asterisk to match any id that starts with the previous character.

```
plugin.notifications.packages.ignored=liferay/sample-jsp-portlet
```

Portlet

Set this property to define the default virtual path for all hot deployed portlets. See liferay-portlet-app_5_1_0.dtd and the virtual-path element for more information.

```
portlet.virtual.path=
```

Set this property to true to validate portlet.xml against the portlet schema.

```
portlet.xml.validate=true
```

Set this property to add a security check when portlets are dynamically added to a page.

```
portlet.add.default.resource.check.enabled=true
```

Set a list of comma delimited list of portlet ids that will bypass this security check set in the previous property.

```
portlet.add.default.resource.check.whitelist=58,86,87,88,103,113,145
```

Input a list of comma delimited struts actions that will bypass the security check set in the property portlet.add.default.resource.check.enabled.

```
portlet.add.default.resource.check.whitelist.actions=\
/journal/rss,\
/language/view
```

Persistence

Set the provider for ORM persistence. If this property is set to jpa, then the properties with the prefix jpa. will be read. If this property is set to hibernate, then the properties with the prefix hibernate will be read.

```
persistence.provider=hibernate
persistence.provider=jpa
```

JPA

Input a list of comma delimited JPA configurations.

```
jpa.configs=\
META-INF/mail-orm.xml,\
META-INF/portal-orm.xml
```

Set the name of the JPA provider.

```
jpa.provider=eclipselink
jpa.provider=hibernate
jpa.provider=openjpa
jpa.provider=toplink
```

Specify provider specific properties prefixed with jpa.provider.property.

```
jpa.provider.property.eclipselink.allow-zero-id=true
jpa.provider.property.eclipselink.logging.level=FINEST
jpa.provider.property.eclipselink.logging.timestamp=true
```

The LoadTimeWeaver interface is a Spring class that allows JPA ClassTransformer instances to be plugged in a specific manner depending on

the environment.

Not all JPA providers require a JVM agent (Hibernate is an example). If your provider does not require an agent or you have other alternatives (such as applying enhancements at build time through a custom compiler or an Ant task), then the loadtime weaver should not be used.

```
jpa.load.time.weaver=org.springframework.instrument.classloading.ReflectiveLoadTimeWeaver  
jpa.load.time.weaver=org.springframework.instrument.classloading.glassfish.GlassFishLoadTimeWeaver  
jpa.load.time.weaver=org.springframework.instrument.classloading.oc4j.OC4JLoadTimeWeaver  
jpa.load.time.weaver=org.springframework.instrument.classloading.weblogic.WebLogicLoadTimeWeaver
```

Specify a specific database platform setting if the JPA provider is not able to detect the database platform.

Valid values for the Hibernate and OpenJPA providers are: DB2, DERBY, HSQL, INFORMIX, MYSQL, ORACLE, POSTGRESQL, SQL_SERVER, and SYBASE.

Valid values for the EclipseLink provider are:

```
org.eclipse.persistence.platform.database.DB2MainframePlatform,  
org.eclipse.persistence.platform.database.DB2Platform,  
org.eclipse.persistence.platform.database.DerbyPlatform,  
org.eclipse.persistence.platform.database.HSQLPlatform,  
org.eclipse.persistence.platform.database.InformixPlatform,  
org.eclipse.persistence.platform.database.MySQLPlatform,  
org.eclipse.persistence.platform.database.OraclePlatform,  
org.eclipse.persistence.platform.database.PostgreSQLPlatform,  
org.eclipse.persistence.platform.database.SQLServerPlatform, or  
org.eclipse.persistence.platform.database.SybasePlatform.
```

Check with JPA provider's documentation for details and all possible values.

```
jpa.database.platform=
```

Liferay will automatically detect the database type by initializing DBUtil. You can override the value here if needed. Expected values are: db2, derby, firebird, hypersonic, informix, ingres, interbase, jdatastore, mysql, oracle, postgresql, sap, sqlserver, and sybase.

```
jpa.database.type=
```

Transaction Manager

Set the transaction manager. It must be a class that extends org.springframework.transaction.support.AbstractPlatformTransactionManager.

The application server specific transaction managers provide XA transactions by leveraging application server specific data sources and thus require additional application server specific configuration. You should not modify this unless you know what you're doing.

```
transaction.manager.impl=org.springframework.orm.hibernate3.HibernateTransactionManager  
transaction.manager.impl=org.springframework.transaction.jta.JtaTransactionManager  
transaction.manager.impl=org.springframework.transaction.jta.OC4JJtaTransactionManager  
transaction.manager.impl=org.springframework.transaction.jta.WebLogicJtaTransactionManager  
transaction.manager.impl=org.springframework.transaction.jta.WebSphereUowTransactionManager
```

Additional properties that follow the pattern `transaction.manager.property.*` will be read to call the setters on the transaction manager. For example, the property `transaction.manager.property.globalRollbackOnParticipationFailure`, will call the setter `setGlobalRollbackOnParticipationFailure` on the transaction manager. The list of available setters depends on the implementation specified in the property `transaction.manager.impl.allowCustomIsolationLevels` should be set to true when using the `JtaTransactionManager`.

```
transaction.manager.property.allowCustomIsolationLevels=true  
transaction.manager.property.globalRollbackOnParticipationFailure=false
```

Portlet Coordination

Set this property to specify how events are distributed. If the value is `layout-set`, then events will be distributed to all portlets contained in a layout set. If the value is `layout`, then events will be distributed to all portlets that are present in a layout.

```
portlet.event.distribution=layout
```

Set this property to specify how public render parameters are distributed. If the value is `layout-set`, then public render parameters will be distributed to all portlets contained in a layout set. This will only work correctly if the property `layout.default.p_l_reset` is set to false. If the value is `layout`, then public render parameters will be distributed to all portlets that are present in a layout.

```
portlet.public.render.parameter.distribution=layout
```

Theme

Set this property to true to load the theme's merged CSS files for faster loading for production.

Set this property to false for easier debugging for development. You can also disable fast loading by setting the URL parameter `css_fast_load` to 0.

```
theme.css.fast.load=true
```

Set this property to true to load the theme's merged image files for faster loading for production.

Set this property to false for easier debugging for development. You can also disable fast loading by setting the URL parameter `images_fast_load` to 0.

```
theme.images.fast.load=true
```

Set the theme's shortcut icon.

```
theme.shortcut.icon=liferay.ico
```

Set this property to set the default virtual path for all hot deployed themes. See `liferay-look-and-feel_5_1_0.dtd` and the `virtual-path` element for more information.

```
theme.virtual.path=
```

Set this with an absolute path to specify where imported theme files from a LAR will be stored. This path will override the file-storage path specified in `liferay-theme-loader.xml`.

```
theme.loader.storage.path=
```

Themes can be imported via LAR files. Set this to true if imported themes should use a new theme id on every import. This will ensure that a copy of the old theme is preserved in the theme loader storage path. However, this also means that a lot of themes that are no longer used remain in the file system. It is recommended that you set this to false.

```
theme.loader.new.theme.id.on.import=false
```

Set this to true to decorate portlets by default.

```
theme.portlet.decorate.default=true
```

Set this to true to exposing sharing icons for portlets by default.

```
theme.portlet.sharing.default=false
```

Resource Actions

Input a list of comma delimited resource action configurations that will be read from the class path.

```
resource.actions.configs=resource-actions/default.xml
```

Model Hints

Input a list of comma delimited model hints configurations.

```
model.hints.configs=\n    META-INF/portal-model-hints.xml,\n    META-INF/workflow-model-hints.xml,\n    META-INF/ext-model-hints.xml,\n    META-INF/portlet-model-hints.xml
```

Service Builder

Input a list of common delimited method prefixes designated for read-only transactions. Service Builder will use these prefixes to annotate methods that are to run in read-only transactions.

```
service.builder.service.read.only.prefixes=\n    get,\n    has,\n    is,\n    reindex,\n    search
```

Spring

Input a list of comma delimited Spring configurations. These will be loaded after the bean definitions specified in the contextConfigLocation parameter in web.xml.

Note that there is a special case for hibernate-spring.xml and jpa-spring.xml. Even though both files are specified, only one will actually load at runtime based on the property persistence.provider.

```
spring.configs=\n    META-INF/base-spring.xml,\n    \\\n    META-INF/hibernate-spring.xml,\n    META-INF/infrastructure-spring.xml,\n    META-INF/management-spring.xml,\n    \\\n    META-INF/util-spring.xml,\n    \\\n    META-INF/jpa-spring.xml,\n    \\\n    META-INF/audit-spring.xml,\n    META-INF/cluster-spring.xml,\n    META-INF/editor-spring.xml,\n    META-INF/jcr-spring.xml,\n    META-INF/ldap-spring.xml,\n    META-INF/messaging-core-spring.xml,\n    \\\n    META-INF/
```

```
META-INF/messaging-misc-spring.xml,\  
META-INF/poller-spring.xml,\  
META-INF/rules-spring.xml,\  
META-INF/scheduler-spring.xml,\  
META-INF/scripting-spring.xml,\  
META-INF/search-spring.xml,\  
META-INF/workflow-spring.xml,\  
\  
META-INF/counter-spring.xml,\  
META-INF/document-library-spring.xml,\  
META-INF/mail-spring.xml,\  
META-INF/portal-spring.xml,\  
META-INF/portlet-container-spring.xml,\  
\  
#META-INF/dynamic-data-source-spring.xml,\  
#META-INF/shard-data-source-spring.xml,\  
#META-INF/memcached-spring.xml,\  
#META-INF/monitoring-spring.xml,\  
\  
META-INF/ext-spring.xml
```

Set the bean name for the Liferay data source.

```
spring.hibernate.data.source=liferayDataSource
```

Set the bean name for the Liferay session factory.

```
spring.hibernate.session.factory=&liferaySessionFactory
```

Hibernate

Many of the following properties should only be customized if you have advanced knowledge of Hibernate. They map to various Hibernate configuration options which themselves have detailed documentation. Please see <http://www.hibernate.org> for more information.

Input a list of comma delimited Hibernate configurations.

```
hibernate.configs=\  
  
META-INF/mail-hbm.xml,\  
META-INF/portal-hbm.xml,\  
META-INF/ext-hbm.xml
```

Liferay will automatically detect the Hibernate dialect in `com.liferay.portal.spring.PortalHibernateConfiguration`. Set this property to manually override the automatically detected dialect.

```
#hibernate.dialect=
```

Set the Hibernate connection release mode. You should not modify this unless you know what you're doing. The default setting works best for Spring

managed transactions. See the method `buildSessionFactory` in class `org.springframework.orm.hibernate3.LocalSessionFactoryBean` and search for the phrase "on_close" to understand how this works.

```
hibernate.connection.release_mode=on_close
```

Set the Hibernate cache provider. Ehcache is recommended in a clustered environment. See the property `net.sf.ehcache.configurationResourceName` for detailed configuration.

Examples:

```
hibernate.cache.provider_class= \
com.liferay.portal.dao.orm.hibernate.EhCacheProvider
hibernate.cache.provider_class= \
net.sf.hibernate.cache.HashtableCacheProvider
hibernate.cache.provider_class= \
com.liferay.portal.dao.orm.hibernate.OSCacheProvider
#hibernate.cache.provider_class= \
com.liferay.portal.dao.orm.hibernate.TerracottaCacheProvider
```

This property is used if Hibernate is configured to use Ehcache's cache provider.

```
net.sf.ehcache.configurationResourceName=/ehcache/hibernate.xml
```

Use the following ehcache configuration in a clustered environment.

```
net.sf.ehcache.configurationResourceName=/ehcache/hibernate-clustered.xml
```

Uncomment the following in a Terracotta environment.

```
#net.sf.ehcache.configurationResourceName=/ehcache/hibernate-terracotta.xml
```

Set other Hibernate cache settings.

```
hibernate.cache.use_query_cache=true
hibernate.cache.use_second_level_cache=true
hibernate.cache.use_minimal_puts=true
hibernate.cache.use_structured_entries=false
```

Use these properties to disable Hibernate caching. This may be a performance hit; you may only want to use these properties for diagnostic purposes.

```
hibernate.cache.provider_class=org.hibernate.cache.NoCacheProvider
hibernate.cache.use_query_cache=false
hibernate.cache.use_second_level_cache=false
```

Set the JDBC batch size to improve performance. If you're using Oracle 9i, however, you must set the batch size to 0 as a workaround for a hanging bug in the Oracle driver. See LEP-1234 for more information.

Examples:

```
hibernate.jdbc.batch_size=20  
hibernate.jdbc.batch_size=0
```

Set other miscellaneous Hibernate properties.

```
hibernate.jdbc.use_scrollable_resultset=true  
hibernate.bytecode.use_reflection_optimizer=true  
hibernate.show_sql=false
```

Use the classic query factory until WebLogic and Hibernate 3 can get along. See <http://www.hibernate.org/250.html#A23> for more information.

```
hibernate.query.factory_class=org.hibernate.hql.classic.ClassicQueryTranslat  
orFactory
```

Set this property to true to enable Hibernate cache monitoring. See LPS-2056 for more information.

```
hibernate.generate_statistics=false
```

When using DB2, set:

```
hibernate.dialect=com.liferay.portal.dao.orm.hibernate.
```

JDBC

Set the JNDI name to lookup the JDBC data source. If none is set, then the portal will attempt to create the JDBC data source based on the properties prefixed with `jdbc.default`.

```
#jdbc.default.jndi.name=jdbc/LiferayPool
```

Set the properties used to create the JDBC data source. These properties will only be read if the property `jdbc.default.jndi.name` is not set.

The default settings are configured for an in-memory database called Hyersonic that is not recommended for production use. Please change the properties to use another database.

Add `dynamic-data-source-spring.xml` to the property `spring.configs` to configure the portal to use one database cluster for read calls and another database cluster for write calls. The convention is to create a set of properties prefixed with `jdbc.read`. to handle read calls and another set of properties prefixed with `jdbc.write`. to handle write calls. These data sources can also be created via JNDI by setting the properties `jdbc.read.jndi.name` and `jdbc.write.jndi.name`.

DB2

```
jdbc.default.driverClassName=com.ibm.db2.jcc.DB2Driver  
jdbc.default.url=jdbc:db2:lportal  
jdbc.default.username=db2admin  
jdbc.default.password=lportal
```

Derby

```
jdbc.default.driverClassName=org.apache.derby.jdbc.EmbeddedDriver  
jdbc.default.url=jdbc:derby:lportal  
jdbc.default.username=  
jdbc.default.password=
```

Hypersonic

```
jdbc.default.driverClassName=org.hsqldb.jdbcDriver  
jdbc.default.url=jdbc:hsqldb:${liferay.home}/data/hsq/lportal  
jdbc.default.username=sa  
jdbc.default.password=
```

MySQL

```
jdbc.default.driverClassName=com.mysql.jdbc.Driver  
jdbc.default.url=jdbc:mysql://localhost/lportal?  
useUnicode=true&characterEncoding=UTF-8&useFastDateParsing=false  
jdbc.default.username=  
jdbc.default.password=
```

Oracle

```
jdbc.default.driverClassName=oracle.jdbc.driver.OracleDriver  
jdbc.default.url=jdbc:oracle:thin:@localhost:1521:xe  
jdbc.default.username=lportal  
jdbc.default.password=lportal
```

P6Spy

```
jdbc.default.driverClassName=com.p6spy.engine.spy.P6SpyDriver  
jdbc.default.url=jdbc:mysql://localhost/lportal?  
useUnicode=true&characterEncoding=UTF-8&useFastDateParsing=false  
jdbc.default.username=  
jdbc.default.password=
```

PostgreSQL

```
jdbc.default.driverClassName=org.postgresql.Driver  
jdbc.default.url=jdbc:postgresql://localhost:5432/lportal  
jdbc.default.username=sa  
jdbc.default.password=
```

SQL Server

```
jdbc.default.driverClassName=net.sourceforge.jtds.jdbc.Driver  
jdbc.default.url=jdbc:jtds:sqlserver://localhost/lportal  
jdbc.default.username=sa  
jdbc.default.password=
```

Sybase

```
jdbc.default.driverClassName=net.sourceforge.jtds.jdbc.Driver  
jdbc.default.url=jdbc:jtds:sybase://localhost:5000/lportal  
jdbc.default.username=sa  
jdbc.default.password=
```

Liferay uses C3PO by default for connection pooling. The data source factory can be configured to use JNDI or another pooling implementation by modifying `infrastructure-spring.xml`. See <http://www.mchange.com/projects/c3p0/index.html> configuration for a list of additional fields used by C3PO for configuring the database connection.

```
jdbc.default.maxPoolSize=50  
jdbc.default.minPoolSize=5
```

Custom SQL

Input a list of comma delimited custom SQL configurations. Liferay Administrators should never need to customize this; this is more of an option for developers who are customizing Liferay's behavior.

```
custom.sql.configs=custom-sql/default.xml
```

Some databases do not recognize a `NULL IS NULL` check. Set the `custom.sql.function.isnull` and `custom.sql.function.isnotnull` properties for your specific database.

There is no need to manually set these properties because `com.liferay.portal.spring.PortalHibernateConfiguration` already sets it. These properties are available, however, so that you can see how you can override it for a database that `PortalHibernateConfiguration` does not yet know how to auto configure.

DB2

```
custom.sql.function.isnull=CAST(? AS VARCHAR(32672)) IS NULL  
custom.sql.function.isnotnull=CAST(? AS VARCHAR(32672)) IS NOT NULL
```

Database

Specify any database vendor specific settings.

MySQL

```
database.mysql.engine=InnoDB
```

Ehcache

Set the classpath to the location of the Ehcache config file for internal caches. Edit the file specified in the property `ehcache.multi-vm.config.location` to enable clustered cache.

```
ehcache.single.vm.config.location=/ehcache/liferay-single-vm.xml  
ehcache.multi.vm.config.location=/ehcache/liferay-multi-vm.xml
```

Use the following in a clustered environment.

```
ehcache.multi.vm.config.location=/ehcache/liferay-multi-vm-clustered.xml
```

JavaScript

Set a list of JavaScript files that will be loaded programmatically in `/html/common/themes/top_js.jsp`.

There are two lists of files specified in the properties `javascript.barebone.files` and `javascript.everything.files`.

As the name suggests, the barebone list is the minimum list of JavaScript files required for most cases. The everything list includes everything else not listed in the barebone list.

The two lists of files exist for performance reasons because unauthenticated users usually do not utilize all the JavaScript that is available. See the property `javascript.barebone.enabled` for more information on the logic of when the barebone list is used and when the everything list is used and how to customize that logic.

The list of files are also merged and packed for further performance improvements. See the property `javascript.fast.load` for more details.

Specify the list of barebone files.

The ordering of the JavaScript files is important.

The Liferay scripts are grouped in such a way that the first grouping denotes utility scripts that are used by the second and third groups. The second grouping denotes utility classes that rely on the first group, but does not rely on the second or third group. The third grouping denotes modules that rely on the first and second group.

```
javascript.barebone.files=\  
  \  
  #  
  # YUI core  
  #  
  \  
  aui/yui/yui.js,\
```

```
\  
#  
# YUI modules  
#  
\  
aui/attribute/attribute.js,\  
aui/event-custom/event-custom.js,\  
aui/loader/loader.js,\  
aui/oop/oop.js,\  
\  
#  
# Alloy core  
#  
\  
aui/aui-base/aui-base.js,\  
\  
#  
# Liferay base utility scripts  
#  
\  
liferay/dependency.js,\  
liferay/language.js,\  
liferay/liferay.js,\  
liferay/modules.js,\  
liferay/util.js,\  
\  
#  
# Liferay utility scripts  
#  
\  
liferay/events.js,\  
liferay/portal.js,\  
liferay/portlet.js,\  
liferay/portlet_sharing.js
```

Specify the list of everything files (everything else not already in the list of barebone files).

```
javascript.everything.files=\  
\  
#  
# Liferay modules  
#  
\  
liferay/address.js,\  
liferay/layout_configuration.js,\  
liferay/layout_exporter.js,\  
liferay/workflow.js,\
```

```
\n
#\n
# Deprecated JS\n
#\n
\\n
liferay/deprecated.js
```

Set this property to false to always load JavaScript files listed in the property `javascript.everything.files`. Set this to true to sometimes load `javascript.barebone.files` and sometimes load `javascript.everything.-files`.

The default logic is coded in `com.liferay.portal.events.ServicePreAction` in such a way that unauthenticated users get the list of barebone JavaScript files whereas authenticated users get both the list of barebone JavaScript files and the list of everything JavaScript files.

`javascript.barebone.enabled=true`

Set this property to true to load the packed version of files listed in the properties `javascript.barebone.files` or `javascript.everything.files`.

Input a list of comma delimited properties that are valid bundle ids for the JavaScript minifier.

```
javascript.bundle.ids=\n    javascript.barebone.files,\n    javascript.everything.files
```

Define a bundle directory for each property listed in `javascript.bundle.ids`.

`javascript.bundle.dir[javascript.barebone.files]=/html/js`
`javascript.bundle.dir[javascript.everything.files]=/html/js`

Define the bundle dependencies using any property listed in `javascript.bundle.ids`.

`javascript.bundle.dependencies[javascript.everything.files]=javascript.barebone.files`

Set this property to false for easier debugging for development. You can also disable fast loading by setting the URL parameter `js_fast_load` to 0.

`javascript.fast.load=true`

Set the following to true to enable the display of JavaScript logging.

`javascript.log.enabled=false`

Combo

The combo servlet combines multiple JavaScript files into a bundle based on shared dependencies. This makes loading JavaScript files much faster. Set this to false if the combination should refresh when one of its

JavaScript files has changed. This property should be set to true during development for easier debugging but set to false during production for faster performance.

```
combo.check.timestamp=false
```

SQL Data

Set the default SQL IDs for common objects.

```
sql.data.com.liferay.portal.model.Country.country.id=19  
sql.data.com.liferay.portal.model.Region.region.id=5  
sql.data.com.liferay.portal.model.ListType.account.address=10000  
sql.data.com.liferay.portal.model.ListType.account.email.address=10004  
sql.data.com.liferay.portal.model.ListType.contact.email.address=11003  
sql.data.com.liferay.portal.model.ListType.organization.status=12017
```

Company

This sets the default web id. Omni admin users must belong to the company with this web id.

```
company.default.web.id=liferay.com
```

This sets the default home URL of the portal.

```
company.default.home.url=/web/guest
```

The portal can authenticate users based on their email address, screen name, or user id.

```
company.security.auth.type=emailAddress  
company.security.auth.type=screenName  
company.security.auth.type=userId
```

Set this to true to ensure users login with https.

```
company.security.auth.requires.https=false
```

Set the following to true to allow users to select the *remember me* feature to automatically login to the portal.

```
company.security.auto.login=true
```

Set the following to the maximum age (in number of seconds) of the browser cookie that enables the *remember me* feature. A value of 31536000 signifies a lifespan of one year. A value of -1 signifies a lifespan of a browser session.

Rather than setting this to 0, set the property `company.security.auto.-login` to false to disable the *remember me* feature.

```
company.security.auto.login.max.age=31536000
```

Set the following to true to allow users to ask the portal to send them their password.

```
company.security.send.password=true
```

Set the following to true to allow strangers to create accounts and register themselves on the portal.

```
company.security.strangers=true
```

Enter a friendly URL of a page that will be used to create new accounts whenever the user clicks the *create account* link in the login portlet. This allows providing custom portlets to create accounts. By default, the portal's *create account* will be used.

```
#company.security.strangers.url=/create_account
```

Set the following to true if strangers can create accounts with email addresses that match the company mail suffix. This property is not used unless `company.security.strangers` is also set to true.

```
company.security.strangers.with.mx=true
```

Set the following to true if strangers who create accounts need to be verified via email.

```
company.security.strangers.verify=false
```

Set the following to true to allow community administrators to use their own logo instead of the enterprise logo.

```
company.security.community.logo=true
```

Input a list of sections that will be included as part of the company settings form.

```
company.settings.form.configuration=general,authentication,default-user-associations,reserved-credentials,mail-host-names,email-notifications  
company.settings.form.identification=addresses,phone-numbers,additional-email-addresses,websites  
company.settings.form.miscellaneous=display-settings
```

Users

Set the following to false if users cannot be deleted.

```
users.delete=true
```

Set the following to true to always autogenerate user screen names even if the user gives a specific user screen name.

```
users.screen.name.always.autogenerate=false
```

Input a class name that extends `com.liferay.portal.security.auth.ScreenNameGenerator`. This class will be called to generate user screen names.

```
users.screen.name.generator=com.liferay.portal.security.auth.ScreenNameGenerator
```

Set this to true when you want the validation to allow for creation of numeric screen names.

```
users.screen.name.allow.numeric=false
```

Input a class name that extends com.liferay.portal.security.auth.ScreenNameValidator. This class will be called to validate user ids.

Examples:

```
users.screen.name.validator=com.liferay.portal.security.auth.ScreenNameValidator
```

```
users.screen.name.validator=com.liferay.portal.security.auth.LiberalScreenNameValidator
```

Input a class name that implements com.liferay.portal.security.auth.FullNameGenerator. This class will be called to generate a full name from the user's first, middle and last names.

```
users.full.name.generator=com.liferay.portal.security.auth.DefaultFullNameGenerator
```

Set this to false if you want to be able to create users without an email address. Note that not requiring an email address disables some features that depend on an email address.

```
users.email.address.required=true
```

Set the suffix of the email address that will be automatically generated for a user that does not have an email address. This property is not used unless the property users.email.address.required is set to false. The autogenerated email address will be the user id plus the specified suffix.

```
users.email.address.auto.suffix=@no-emailaddress.com
```

Input a class name that implements com.liferay.portal.security.auth.EmailAddressGenerator. This class will be called to generate an email address for a user that does not specify an email address. This class will only be used if the property users.email.address.required is set to false.

```
users.email.address.generator=com.liferay.portal.security.auth.DefaultEmailAddressGenerator
```

Input a class name that implements com.liferay.portal.security.auth.FullNameGenerator. This class will be called to generate a full name from the user's first, middle and last names.

```
users.full.name.generator=com.liferay.portal.security.auth.DefaultFullNameGenerator
```

Input a class name that implements com.liferay.portal.security.auth.FullNameValidator. This class will be called to validate user first, middle and last names.

```
users.full.name.validator=com.liferay.portal.security.auth.DefaultFullNameValidator
```

Set the maximum file size for user portraits. A value of 0 for the maximum file size can be used to indicate unlimited file size. However, the maximum file size allowed is set in property com.liferay.portal.upload.UploadServletRequestImpl.max.size found in system.properties.

```
users.image.max.size=307200
```

Set the maximum user portrait height and width in pixels. A value of 0 indicates no restrictions on user portrait dimensions.

```
users.image.max.height=120  
users.image.max.width=100
```

Set this to true to record last login information for a user.

```
users.update.last.login=true
```

Input a list of sections that will be included as part of the user form when adding a user.

```
users.form.add.main=details,organizations  
users.form.add.identification=  
users.form.add.miscellaneous=
```

Input a list of sections that will be included as part of the user form when updating a user.

```
users.form.update.main=details,password,organizations,communities,user-groups,roles,categorization  
users.form.update.identification=addresses,phone-numbers,additional-email-addresses,websites,instant-messenger,social-network,sms,open-id  
users.form.update.miscellaneous=announcements,display-settings,comments,custom-attributes
```

Set this to true to enable reminder queries that are used to help reset a user's password.

```
users.reminder.queries.enabled=true  
users.reminder.queries.custom.question.enabled=true
```

Input a list of questions used for reminder queries.

```
users.reminder.queries.questions=what-is-your-primary-frequent-flyer-number,what-is-your-library-card-number,what-was-your-first-phone-number,what-was-your-first-teacher's-name,what-is-your-father's-middle-name
```

Set this to true to search users from the index. Set this to false to search users from the database. Note that setting this to false will disable the ability to search users based on Expando attributes.

```
users.search.with.index=true
```

Set a property with the prefix `users.update.user.name.` and a suffix with the class name that should be updated whenever a user's name has changed.

```
users.update.user.name.com.liferay.portlet.messageboards.model.MBMessage  
=true
```

Input a list of user attributes that will be included when exporting users to a CSV file. You can include custom fields by adding the prefix `expando:` to the attribute name.

```
users.export.csv.fields=fullName,emailAddress
```

Set this to false to enable users without a reminder query to reset their password.

```
users.reminder.queries.required=true
```

Set a property with the prefix `users.update.user.name.` and a suffix with the class name that should be updated whenever a user's name has changed.

```
users.update.user.name.com.liferay.portlet.messageboards.model.MBMessage  
=true
```

Input a list of user attributes that will be included when exporting users to a CSV file. You can include custom fields by adding the prefix `expando:` to the attribute name.

```
users.export.csv.fields=fullName,emailAddress
```

When importing and exporting users, the portal will use this mapping to connect LDAP user attributes and portal contact attributes.

See `com.liferay.portal.model.ContactModel` for a list of attributes.

```
ldap.contact.mappings=
```

When importing and exporting users, the portal will use this mapping to connect LDAP user attributes and portal contact's custom attributes.

```
ldap.contact.custom.mappings=
```

See `com.liferay.portal.model.UserModel` for a list of attributes.

```
ldap.user.mappings=uuid=uuid\ncn=cn\npassword=userPassword\nemailAddress=mail\nfirstName=givenName\nlastName=sn\njobTitle=title\ngroupName=groupMembership
```

When importing and exporting users, the portal will use this mapping to connect LDAP user attributes and portal user's custom attributes.

```
ldap.user.custom.mappings=
```

Set this to true if the portal should automatically create a role per group imported from LDAP. The role will be assigned to the group so that users can automatically inherit that role when they are assigned to the group.

```
ldap.import.create.role.per.group=false
```

Facebook Connection

```
facebook.connect.auth.enabled=false
facebook.connect.app.id=
facebook.connect.app.secret=
facebook.connect.graph.url=https://graph.facebook.com
facebook.connect.oauth.auth.url=https://graph.facebook.com/oauth/authorize
facebook.connect.oauth.redirect.url=http://localhost:8080/c/login/facebook_connect_oauth
facebook.connect.oauth.token.url=https://graph.facebook.com/oauth/access_token
```

NTLM

Set this to true to enable NTLM single sign on. NTLM will work only if LDAP authentication is also enabled and the authentication is made by screen name. If set to true, then the property auto.login.hooks must contain a reference to the class com.liferay.portal.security.auth.NtlmAutoLogin and the filter com.liferay.portal.servlet.filters.sso.ntlm.NtlmFilter must be referenced in web.xml.

```
ntlm.auth.enabled=false
ntlm.auth.domain.controller=127.0.0.1
ntlm.auth.domain.controller.name=EXAMPLE
ntlm.auth.domain=EXAMPLE
ntlm.auth.service.account=LIFERAY$@EXAMPLE.COM
ntlm.auth.service.password=test
```

Request Header Authentication

Set this to true to automatically import users from LDAP if they do not exist in the portal. The property auto.login.hooks must contain a reference to the class com.liferay.portal.security.auth.RequestHeaderAutoLogin to enable request header authentication.

```
request.header.auth.import.from.ldap=false
```

Authentication Token

Set this to true to enable authentication token security checks. The checks can be disabled for specific actions via the property auth.token.ignore.actions or for specific portlets via the init parameter check-auth-token in portlet.xml.

```
auth.token.check.enabled=true
```

Set the authentication token class. This class must implement com.liferay.portal.security.auth.AuthToken. This class is used to prevent CSRF attacks. See <http://issues.liferay.com/browse/LPS-8399> for more information.

```
auth.token.impl=com.liferay.portal.security.auth.SessionAuthToken
```

Input a list of comma delimited struts actions that will not be checked for an authentication token.

```
auth.token.ignore.actions=\
/asset/rss, \
\
/blogs/rss, \
\
/document_library/edit_file_entry, \
\
/journal/rss, \
\
/image_gallery/edit_image, \
\
/login/login, \
\
/message_boards/rss, \
\
/wiki/edit_page_attachment, \
/wiki/rss
```

Set the character sets for password validation.

```
passwords.passwordpolicytoolkit.charset.lowercase=abcdefghijklmnopqrstuvwxyz
```

```
z
```

```
passwords.passwordpolicytoolkit.charset.numbers=23456789
```

```
passwords.passwordpolicytoolkit.charset.symbols=_.!@#$*=-?
```

```
YZ
```

Groups and Roles

Input a list of comma delimited system group names that will exist in addition to the standard system groups. When the server starts, the portal checks to ensure all system groups exist. Any missing system group will be created by the portal.

```
system.groups=
```

Input a list of comma delimited system role names that will exist in addition to the standard system roles. When the server starts, the portal checks to ensure all system roles exist. Any missing system role will be created by the portal.

The standard system roles are: Administrator, Guest, Power User, and User. These roles cannot be removed or renamed.

```
system.roles=
```

Set the description of the Administrator system role.

```
system.role.Administrator.description=Administrators are super users who can do anything.
```

Set the description of the Guest system role.

```
system.role.Guest.description=Unauthenticated users always have this role.
```

Set the description of the Owner system role.

```
system.role.Owner.description=This is an implied role with respect to the objects users create.
```

Set the description of the Power User system role.

```
system.role.Power.User.description=Power Users have their own public and private pages.
```

Set the description of the User system role.

```
system.role.User.description=Authenticated users should be assigned this role.
```

Input a list of comma delimited system community role names that will exist in addition to the standard system community roles. When the server starts, the portal checks to ensure all system community roles exist. Any missing system community role will be created by the portal.

The standard system community roles are: Community Administrator, Community Member, and Community Owner. These roles cannot be removed or renamed.

```
system.community.roles=
```

Set the description of the Community Administrator system community role.

```
system.community.role.Community.Administrator.description=Community Administrators are super users of their community but cannot make other users into Community Administrators.
```

Set the description of the Community Member system community role.

```
system.community.role.Community.Member.description>All users who belong to a community have this role within that community.
```

Set the description of the Community Owner system community role.

```
system.community.role.Community.Owner.description=Community Owners are super users of their community and can assign community roles to users.
```

Input a list of comma delimited system organization role names that will exist in addition to the standard system organization roles. When the server

starts, the portal checks to ensure all system organization roles exist. Any missing system organization role will be created by the portal.

The standard system organization roles are: Organization Administrator, Organization Member, and Organization Owner. These roles cannot be removed or renamed.

```
system.organization.roles=
```

Set the description of the Organization Administrator system organization role.

```
System.organization.role.Organization.Administrator.description=Organization Administrators are super users of their organization but cannot make other users into Organization Administrators.
```

Set the description of the Organization Member system organization role.

```
System.organization.role.Organization.Member.description>All users who belong to a organization have this role within that organization.
```

Set the description of the Organization Owner system organization role.

```
System.organization.role.Organization.Owner.description=Organization Owners are super users of their organization and can assign organization roles to users.
```

Omni admin users can administer the portal's core functionality: gc, shutdown, etc. Omni admin users must belong to the default company.

Multiple portal instances might be deployed on one application server, and not all of the administrators should have access to this core functionality. Input the ids of users who are omniadmin users.

Leave this field blank if users who belong to the right company and have the Administrator role are allowed to administer the portal's core functionality.

```
omniadmin.users=
```

Set the following to true if all users are required to agree to the terms of use.

```
terms.of.use.required=true
```

Specify the group id and the article id of the Web Content article that will be displayed as the terms of use. The default text will be used if no Web Content article is specified.

```
terms.of.use.journal.article.group.id=
terms.of.use.journal.article.id=
```

Specify subtypes of roles if you want to be able to search for roles using your custom criteria.

```
roles.community.subtypes=
```

```
roles.organization.subtypes=
roles.regular.subtypes=
```

Organizations

Specify the names of your organization(s). For example, you could use Teams, Clubs, Parishes, or anything which describes your hierarchical structure.

```
organizations.types=regular-organization,location
```

Specify which organizations can be at the top of the hierarchy.

```
organizations.rootable[regular-organization]=true
```

Specify which organizations can be children.

```
organizations.children.types[regular-organization]=regular-
organization,location
```

Set this to true if organizations can have an associated country.

```
organizations.country.enabled[regular-organization]=false
```

Set this to true if organizations must have an associated country.

```
organizations.country.required[regular-organization]=false
```

By default, Locations cannot be at the top of the hierarchy, because they cannot have children. You must specify the following properties for each organization type you create.

Example:

```
organizations.rootable[location]=false
#organizations.children.types[location]=
organizations.country.enabled[location]=true
organizations.country.required[location]=true
```

Input a list of sections that will be included as part of the organization form when adding an organization.

```
organizations.form.add.main=details
organizations.form.add.identification=
organizations.form.add.miscellaneous=
```

Input a list of sections that will be included as part of the organization form when updating an organization.

```
organizations.form.update.main=details
organizations.form.update.identification=addresses,phone-numbers,additional-
email-addresses,websites,services
organizations.form.update.miscellaneous=comments,reminder-queries,custom-
attributes
```

Set this property to true if you want any administrator that creates an organization to be automatically assigned to that organization.

```
organizations.assignment.auto=false
```

Set this property to false if you want any administrator of an organization to be able to assign any user to that organization. By default, he will only be able to assign the users of the organizations and suborganizations that he can manage.

```
organizations.assignment.strict=true
```

Set this property to true if you want users to only be members of the organizations to which they are assigned explicitly. By default they will also become implicit members of the ancestors of those organizations. for example if a user belongs to Liferay Spain he will implicitly be a member of the ancestors Liferay Europe and Liferay Global and will be able to access their private pages.

```
organizations.membership.strict=false
```

Security Manager

Set this to true to use Liferay's `java.lang.SecurityManager` implementation. This should never be set to true except for debugging purposes.

```
portal.security.manager.enable=false
```

Basic Authentication

Set this to true to require a password when using basic authentication. Only set this to false if additional security measures are in place to ensure users have been properly authenticated.

```
basic.auth.password.required=true
```

Languages and Time Zones

Specify the available locales. Messages corresponding to a specific language are specified in properties files with file names matching that of `content/Language_*.properties`. These values can also be overridden in properties files with file names matching that of `content/Language-ext_*.properties`. Use a comma to separate each entry.

All locales must use UTF-8 encoding.

See the following links to specify language and country codes:

<http://ftp.ics.uci.edu/pub/ietf/http/related/iso639.txt>

http://userpage.chemie.fu-berlin.de/diverse/doc/ISO_3166.html

```
locales=ar_SA,ca_AD,ca_ES,zh_CN,zh_TW,cs_CZ,nl_NL,en_US,fi_FI,fr_FR,de_DE,el_GR,hu_HU,it_IT,ja_JP,ko_KR,nb_NO,fa_IR,p1_PL,pt_BR,pt_PT,ru_RU,es_ES,sv_SE,tr_TR,vi_VN
```

Set the following to true if unauthenticated users get their preferred language from the Accept-Language header. Set the following to false if unauthenticated users get their preferred language from their company.

```
locale.default.request=false
```

Specify the available time zones. The specified ids must match those from the class java.util.TimeZone.

```
time.zones=\n    Pacific/Midway,\n    Pacific/Honolulu,\n    America/Anchorage,\n    America/Los_Angeles,\n    America/Denver,\n    America/Chicago,\n    America/New_York,\n    America/Puerto_Rico,\n    America/St_Johns,\n    America/Sao_Paulo,\n    America/Noronha,\n    Atlantic/Azores,\n    UTC,\n    Europe/Lisbon,\n    Europe/Paris,\n    Europe/Istanbul,\n    Asia/Jerusalem,\n    Asia/Baghdad,\n    Asia/Tehran,\n    Asia/Dubai,\n    Asia/Kabul,\n    Asia/Karachi,\n    Asia/Calcutta,\n    Asia/Katmandu,\n    Asia/Dhaka,\n    Asia/Rangoon,\n    Asia/Saigon,\n    Asia/Shanghai,\n    Asia/Tokyo,\n    Asia/Seoul,\n    Australia/Darwin,\n    Australia/Sydney,\n    Pacific/Guadalcanal,\n    Pacific/Auckland,\n    Pacific/Enderbury,\n    Pacific/Kiritimati
```

Look and Feel

Set the following to false if the system does not allow users to modify the look and feel.

```
look.and.feel.modifiable=true
```

Set the default layout template id.

```
default.layout.template.id=2_columns_iI
```

Set the default theme id for regular themes.

```
default.regular.theme.id=classic
```

Set the default color scheme id for regular themes.

```
default.regular.color.scheme.id=01
```

Set the default theme id for wap themes.

```
default.wap.theme.id=mobile
```

Set the default color scheme id for wap themes.

```
default.wap.color.scheme.id=01
```

Set the following to true if you want a change in the theme selection of the public or private group to automatically be applied to the other (i.e. if public and private group themes should always be the same).

```
theme.sync.on.group=false
```

Layouts

Set this to true to remember maximized window states across different pages.

```
layout.remember.maximized.window.state=false
```

Set this to true to enable comments for pages.

```
layout.comments.enabled=true
```

Set this to true to remember maximized window states across different pages.

```
layout.remember.maximized.window.state=false
```

Editors

You can configure individual JSP pages to use a specific implementation of the available WYSIWYG editors: ckeditor, fckeditor, liferay, simple, tinymce, or tinymceSimple.

```
editor.wysiwyg.default=ckeditor
```

```
editor.wysiwyg.portal-web.docroot.html.portlet.blogs.edit_entry.jsp=ckeditor
```

```
editor.wysiwyg.portal-
web.docroot.html.portlet.calendar.edit_configuration.jsp=ckeditor
editor.wysiwyg.portal-
web.docroot.html.portlet.enterprise_admin.view.jsp=ckeditor
editor.wysiwyg.portal-
web.docroot.html.portlet.invitation.edit_configuration.jsp=ckeditor
editor.wysiwyg.portal-
web.docroot.html.portlet.journal.edit_article_content.jsp=ckeditor
editor.wysiwyg.portal-
web.docroot.html.portlet.journal.edit_article_content_xsd_el.jsp=
ckeditor
editor.wysiwyg.portal-
web.docroot.html.portlet.journal.edit_configuration.jsp=ckeditor
editor.wysiwyg.portal-
web.docroot.html.portlet.login.configuration.jsp=ckeditor
editor.wysiwyg.portal-web.docroot.html.portlet.mail.edit.jsp=ckeditor
editor.wysiwyg.portal-
web.docroot.html.portlet.mail.edit_message.jsp=ckeditor
editor.wysiwyg.portal-
web.docroot.html.portlet.message_boards.edit_configuration.jsp=
ckeditor
editor.wysiwyg.portal-
web.docroot.html.portlet.shopping.edit_configuration.jsp=ckeditor
editor.wysiwyg.portal-web.docroot.html.portlet.wiki.edit.html.jsp=ckeditor
```

Fields

Input a list of comma delimited user types who can edit their own fields. Valid types are administrator, user-mx, and user-without-mx.

Set a value of administrator if an administrator can edit the specified field. An administrator is anyone who has the Administrator role.

Set a value of user-mx if a user who has an email address that matches the company mail suffix can edit the specified field.

Set a value of user-without-mx if a user who does not have an email address that matches the company mail suffix can edit the specified field.

Set all three values if all users can edit the specified field. Set a combination of the three values if only a combination of the users can edit the specified field.

```
field.editable.com.liferay.portal.model.User.screenName=administrator,user-
with-mx,user-without-mx
field.editable.com.liferay.portal.model.User.emailAddress=administrator,user-
with-mx,user-without-mx
```

Request

Portlets that have been configured to use private request attributes in liferay-portlet.xml may still want to share some request attributes. This property allows you to configure which request attributes will be shared.

Set a comma delimited list of attribute names that will be shared when the attribute name starts with one of the specified attribute names. For example, if you set the value to `hello_,world_`, then all attribute names that start with `hello_` or `world_` will be shared.

```
request.shared.attributes=LIFERAY_SHARED
```

Session

Specify the number of minutes before a session expires. This value is always overridden by the value set in `web.xml`.

```
session.timeout=30
```

Specify the number of minutes before a warning is sent to the user informing the user of the session expiration. Specify `0` to disable any warnings.

```
session.timeout.warning=1
```

Set the auto-extend mode to true to avoid having to ask the user whether to extend the session or not. Instead it will be automatically extended. The purpose of this mode is to keep the session open as long as the user browser is open and with a portal page loaded. It is recommended to use this setting along with a smaller `session.timeout`, such as 5 minutes for better performance.

```
session.timeout.auto.extend=false
```

Set this to true if the user is redirected to the default page when the session expires.

```
session.timeout.redirect.on.expire=false
```

Portlets that have been configured to use private session attributes in `liferay-portlet.xml` may still want to share some session attributes. This property allows you to configure which session attributes will be shared. Set a comma delimited list of attribute names that will be shared when the attribute name starts with one of the specified attribute names. For example, if you set the value to `hello_,world_`, then all attribute names that start with `hello_` or `world_` will be shared.

Note that this property is used to specify the sharing of session attributes from the portal to the portlet. This is not used to specify session sharing between portlet WARs or from the portlet to the portal.

```
session.shared.attributes=org.apache.struts.action.LOCALE,COMPANY_,USER_,LIFERAY_SHARED
```

Set this to false to disable all persistent cookies. Features like automatically logging in will not work.

```
session.enable.persistent.cookies=true
```

The login process sets several cookies if persistent cookies are enabled. Set this property to set the domain of those cookies.

```
session.cookie.domain=
```

Set the following to true to invalidate the session when a user logs into the portal. This helps prevent phishing. Set this to false if you need the guest user and the authenticated user to have the same session.

```
session.enable.phishing.protection=true
```

Set the following to true to test whether users have cookie support before allowing them to sign in. This test will always fail if tck.url is set to true because that property disables session cookies.

```
session.test.cookie.support=true
```

Set the following to true to disable sessions. Doing this will use cookies to remember the user across requests. This is useful if you want to scale very large sites where the user may be sent to a different server for each request. The drawback to this approach is that you must not rely on the API for sessions provided by the servlet and portlet specs.

This feature is only available for Tomcat and requires that you set Tomcat's Manager class to com.liferay.support.tomcat.session.SessionLessManagerBase.

```
session.disabled=false
```

Input a list of comma delimited class names that extend com.liferay.portal.struts.SessionAction. These classes will run at the specified event.

```
#  
# Servlet session create event  
#  
servlet.session.create.events=com.liferay.portal.events.SessionCreateAction  
  
#  
# Servlet session destroy event  
#  
servlet.session.destroy.events=com.liferay.portal.events.SessionDestroyAction
```

Set the following to true to track user clicks in memory for the duration of a user's session. Setting this to true allows you to view all live sessions in the Admin portlet.

```
session.tracker.memory.enabled=true
```

Set the following to true to track user clicks in the database after a user's session is invalidated. Setting this to true allows you to generate usage reports from the database. Use this cautiously because this will store a lot of usage data.

```
session.tracker.persistence.enabled=false
```

Set the following to true to convert the tracked paths to friendly URLs.

```
session.tracker.friendly.paths.enabled=false
```

Enter a list of comma delimited paths that should not be tracked.

```
session.tracker.ignore.paths=\n    /portal/render_portlet,\n    \n    /document_library/get_file
```

HTTP

Set the maximum number of connections.

```
com.liferay.portal.util.HttpImpl.max.connections.per.host=2\ncom.liferay.portal.util.HttpImpl.max.total.connections=20
```

Set the proxy authentication type.

```
com.liferay.portal.util.HttpImpl.proxy.auth.type=username-password\ncom.liferay.portal.util.HttpImpl.proxy.auth.type=ntlm
```

Set user name and password used for HTTP proxy authentication.

```
com.liferay.portal.util.HttpImpl.proxy.username=\ncom.liferay.portal.util.HttpImpl.proxy.password=
```

Set additional properties for NTLM authentication.

```
com.liferay.portal.util.HttpImpl.proxy.ntlm.domain=\ncom.liferay.portal.util.HttpImpl.proxy.ntlm.host=
```

Set the connection timeout when fetching HTTP content.

```
com.liferay.portal.util.HttpImpl.timeout=10000
```

JAAS

Set the following to false to disable JAAS security checks. Disabling JAAS speeds up login. JAAS must be disabled if administrators are to be able to impersonate other users.

```
portal.jaas.enable=false
```

By default, `com.liferay.portal.security.jaas.PortalLoginModule` loads the correct JAAS login module based on what application server or servlet container the portal is deployed on. Set a JAAS implementation class to override this behavior.

```
portal.jaas.impl=
```

The JAAS process may pass in an encrypted password and the authentication will only succeed if there is an exact match. Set this property to false to relax that behavior so the user can input an unencrypted password.

```
portal.jaas.strict.password=false
```

Set the following to true to enable administrators to impersonate other users.

```
portal.impersonation.enable=true
```

LDAP

Set the values used to connect to a LDAP store.

```
ldap.factory.initial=com.sun.jndi.ldap.LdapCtxFactory  
ldap.base.provider.url=ldap://localhost:10389  
ldap.base.dn=dc=example,dc=com  
ldap.security.principal=uid=admin,ou=system  
ldap.security.credentials=secret  
ldap.referral=follow
```

Settings for `com.liferay.portal.security.auth.LDAPAuth` can be configured from the Admin portlet. It provides out of the box support for Apache Directory Server, Microsoft Active Directory Server, Novell eDirectory, and OpenLDAP. The default settings are for Apache Directory Server.

The `LDAPAuth` class must be specified in the property `auth.pipeline.pre` to be executed.

Encryption is implemented by `com.liferay.util.Encryptor.provider` - class in `system.properties`.

```
ldap.auth.enabled=false  
ldap.auth.required=false
```

Set either bind or password-compare for the LDAP authentication method. Bind is preferred by most vendors so that you don't have to worry about encryption strategies.

```
ldap.auth.method=bind  
ldap.auth.method=password-compare
```

Set the password encryption to used to compare passwords if the property `ldap.auth.method` is set to `password-compare`.

```
ldap.auth.password.encryption.algorithm=  
ldap.auth.password.encryption.algorithm.types=MD5, SHA
```

Active Directory stores information about the user account as a series of bit fields in the `UserAccountControl` attribute.

If you want to prevent disabled accounts from logging into the portal you need to use a search filter similar to the following:

```
(&(objectclass=person)(userprincipalname=@email_address@)(!  
(UserAccountControl:1.2.840.113556.1.4.803:=2)))
```

See the following links:

<http://support.microsoft.com/kb/305144/>

<http://support.microsoft.com/?kbid=269181>

```
ldap.auth.search.filter=(mail=@email_address@)
```

You can write your own class that extends `com.liferay.portal.security.ldap.AttributesTransformer` to transform the LDAP attributes before a user or group is imported to the LDAP store.

```
ldap.attrs.transformer.impl=com.liferay.portal.security.ldap.AttributesTransformer
```

You can write your own class that extends `com.liferay.portal.security.ldap.LDAPUser` to customize the behavior for exporting portal users to the LDAP store.

```
ldap.user.impl=com.liferay.portal.security.ldap.LDAPUser
```

When a user is exported to LDAP and the user does not exist, the user will be created with the following default object classes.

```
ldap.user.default.object.classes=top, person, inetOrgPerson, organizationalPerson
```

When importing and exporting users, the portal will use this mapping to connect LDAP user attributes and portal user variables.

```
ldap.user.mappings=screenName=cn\npassword=userPassword\nemailAddress=mail\nfirstName=givenName\nlastName=sn\njobTitle=title\ngroupName=groupMembership
```

When importing groups, the portal will use this mapping to connect LDAP group attributes and portal user group variables.

```
ldap.group.mappings=groupName=cn\ndescription=description\nuser=uniqueMember
```

Settings for importing users and groups from LDAP to the portal.

```
ldap.import.enabled=false
ldap.import.on.startup=false
ldap.import.interval=10
ldap.import.user.search.filter=(objectClass=inetOrgPerson)
ldap.import.group.search.filter=(objectClass=groupOfUniqueNames)
```

Set either user or group for import method. If set to user, portal will import all users and the groups associated with those users. If set to group, the portal import all groups and the users associated those groups.

This value should be set based on how your LDAP server stores group membership information.

```
ldap.import.method=user
ldap.import.method=group
```

Settings for exporting users from the portal to LDAP. This allows a user to modify his first name, last name, etc. in the portal and have that change

get pushed to the LDAP server. This will only be active if the property `ldap.auth.enabled` is also set to true. New users and groups will be created at the specified DN.

```
ldap.export.enabled=true  
ldap.users_dn=ou=users,dc=example,dc=com  
ldap.groups_dn=ou=groups,dc=example,dc=com
```

Set this to true to use the LDAP's password policy instead of the portal password policy.

```
ldap.password.policy.enabled=false
```

Set these values to be a portion of the error message returned by the appropriate directory server to allow the portal to recognize messages from the LDAP server. The default values will work for Fedora DS.

```
ldap.error.password.age=age  
ldap.error.password.expired=expired  
ldap.error.password.history=history  
ldap.error.password.not.changeable=not allowed to change  
ldap.error.password.syntax=syntax  
ldap.error.password.trivial=trivial  
ldap.error.user.lockoutretry limit
```

CAS

Set this to true to enable CAS single sign on. NTLM will work only if LDAP authentication is also enabled and the authentication is made by screen name. If set to true, then the property `auto.login.hooks` must contain a reference to the class `com.liferay.portal.security.auth.CASAutoLogin` and the filter `com.liferay.portal.servlet.filters.sso.cas.CASFilter` must be referenced in `web.xml`.

```
cas.auth.enabled=false
```

A user may be authenticated from CAS and not yet exist in the portal. Set this to true to automatically import users from LDAP if they do not exist in the portal.

```
cas.import.from.ldap=false
```

Set the default values for the required CAS URLs. Set either `cas.server.name` or `cas.service.url`. Setting `cas.server.name` allows deep linking. See LEP-4423.

```
cas.login.url=https://localhost:8443/cas-web/login  
cas.logout.url=https://localhost:8443/cas-web/logout  
cas.server.name=localhost:8080  
cas.service.url=  
#cas.service.url=http://localhost:8080/c/portal/login  
cas.service.url=http://localhost:8080/c/portal/login
```

```
cas.validate.url=https://localhost:8443/cas-web/proxyValidate
```

NTLM

Set this to true to enable NTLM single sign on. NTLM will work only if LDAP authentication is also enabled and the authentication is made by screen name. If set to true, then the property auto.login.hooks must contain a reference to the class com.liferay.portal.security.auth.NtlmAutoLogin and the filter com.liferay.portal.servlet.filters.sso.ntlm.NtlmFilter must be referenced in web.xml.

```
ntlm.auth.enabled=false  
ntlm.auth.domain.controller=127.0.0.1  
ntlm.auth.domain=EXAMPLE
```

OpenID

Set this to true to enable OpenId authentication. If set to true, then the property auto.login.hooks must contain a reference to the class com.liferay.portal.security.auth.OpenIdAutoLogin.

```
open.id.auth.enabled=true
```

OpenSSO

These properties control Liferay's integration with OpenSSO.

Set this to true to enable OpenSSO authentication.

```
open.sso.auth.enabled=false
```

Set the log in URL and log out URL. The first URL is the link to your OpenSSO server (which can be the same server as the one running Liferay); the second URL is the link to your Liferay Portal.

```
open.sso.login.url=http://openssoshost.example.com:8080/opensso/UI/Login?  
goto=http://portalhost.example.com:8080/c/portal/login  
open.sso.logout.url=http://openssoshost.example.com:8080/opensso/UI/Logout?  
goto=http://portalhost.example.com:8080/web/guest/home
```

Set the URL to the OpenSSO service.

```
open.sso.service.url=http://openssoshost.example.com:8080/opensso
```

Set the HTTP attribute name for the user's screen name.

```
open.sso.screen.name.attr=uid
```

Set the HTTP attribute name for the user's email address.

```
open.sso.email.address.attr=mail
```

Set the HTTP attribute name for the user's Common Name.

```
open.sso.first.name.attr=cn
```

Set the HTTP attribute name for the user's Surname.

```
open.sso.last.name.attr=sn
```

SiteMinder

Set this to true to enable CA SiteMinder single sign on. If set to true, then the property auto.login.hooks must contain a reference to the class com.liferay.portal.security.auth.SiteMinderAutoLogin and the logout.events.post must have a reference to com.liferay.portal.event.SiteMinderLogoutAction for logout to work.

```
siteminder.auth.enabled=false
```

A user may be authenticated from SiteMinder and not yet exist in the portal. Set this to true to automatically import users from LDAP if they do not exist in the portal.

```
siteminder.import.from.ldap=false
```

Set this to the name of the user header that SiteMinder passes to the portal.

```
siteminder.user.header=SM_USER
```

Authentication Pipeline

Input a list of comma delimited class names that implement com.liferay.portal.security.auth.Authenticator. These classes will run before or after the portal authentication begins.

The Authenticator class defines the constant values that should be used as return codes from the classes implementing the interface. If authentication is successful, return SUCCESS; if the user exists but the passwords do not match, return FAILURE; and if the user does not exist on the system, return DNE.

Constants in Authenticator:

```
public static final int SUCCESS = 1;  
public static final int FAILURE = -1;  
public static final int DNE = 0;
```

In case you have several classes in the authentication pipeline, all of them have to return SUCCESS if you want the user to be able to login. If one of the authenticators returns FAILURE or DNE, the login fails.

Under certain circumstances, you might want to keep the information in the portal database in sync with an external database or an LDAP server. This

can easily be achieved by implementing a class via `LDAPAuth` that updates the information stored in the portal user database whenever a user signs in.

Each portal instance can be configured at run time to either authenticate based on user ids or email addresses. See the Admin portlet for more information.

Available authenticators are:

```
com.liferay.portal.security.auth.LDAPAuth
```

See the LDAP properties to configure the behavior of the `LDAPAuth` class.

```
auth.pipeline.pre=com.liferay.portal.security.auth.LDAPAuth  
auth.pipeline.post=
```

Set this to true to enable password checking by the internal portal authentication. If set to false, you're essentially delegating password checking is delegated to the authenticators configured in `auth.pipeline.pre` and `auth.pipeline.post` settings.

```
auth.pipeline.enable.liferay.check=true
```

Input a list of comma delimited class names that implement `com.liferay.portal.security.auth.AuthFailure`. These classes will run when a user has a failed login or when a user has reached the maximum number of failed logins.

```
auth.failure=com.liferay.portal.security.auth.LoginFailure  
auth.max.failures=com.liferay.portal.security.auth.LoginMaxFailures  
auth.max.failures.limit=5
```

Set the following to true if users are forwarded to the last visited path upon successful login. If set to false, users will be forwarded to their default layout page.

```
auth.forward.by.last.path=true
```

The login page reads a redirect by a parameter named `redirect`. If this property is set to true, then users will be redirected to the given redirect path upon successful login. If the user does not have permission to view that page, then the rule set by the property `auth.forward.by.last.path` will apply.

You can set the redirect manually from another application, by appending the `redirect` parameter in a url that looks like this: `/c/portal/login?redirect=%2Fgroup%2Femployees%2Fcalendar`. This url will redirect the user to the path `/group/employees/calendar` upon successful login.

```
auth.forward.by.redirect=true
```

Enter a list of comma delimited paths that can be considered part of the last visited path.

```
auth.forward.last.paths=/document_library/get_file
```

Enter a URL that will be used to login portal users whenever needed. By default, the portal's login page is used.

```
#auth.login.url=/web/guest/home
```

Enter a friendly URL of a page that will be used to login portal users whenever the user is navigating a community and authentication is needed. By default, the portal's login page or the URL set in the property auth.login.url is used.

```
auth.login.community.url=/login
```

Enter the name of the login portlet used in a page identified by the URL of the previous property (if one has been set). This will allow the portlet to have access to the redirect parameter and thus forward the user to the page where he was trying to access when necessary. You should leave the default value unless you have your own custom login portlet.

```
auth.login.portlet.name=58
```

Enter a list of comma delimited paths that do not require authentication.

```
auth.public.paths=\n    /blogs/find_entry,\n    /blogs/rss,\n    /blogs/trackback,\n    \n    /bookmarks/open_entry,\n    \n    /document_library/get_file,\n    \n    /journal/get_article,\n    /journal/get_articles,\n    /journal/get_latest_article_content,\n    /journal/get_structure,\n    /journal/get_template,\n    /journal/view_article_content,\n    /journal_articles/view_article_content,\n    \n    /layout_management/sitemap,\n    \n    /message_boards/find_category,\n    /message_boards/find_message,\n    /message_boards/find_thread,\n    /message_boards/get_message_attachment,\n    /message_boards/rss,\n    \n    /my_places/view,\n    \n
```

```
/polls/view_chart,\  
\\  
/portal/emoticons,\  
/portal/expire_session,\  
/portal/extend_session,\  
/portal/extend_session_confirm,\  
/portal/json_service,\  
/portal/logout,\  
/portal/open_id_request,\  
/portal/open_id_response,\  
/portal/session_click,\  
/portal/session_tree_js_click,\  
/portal/status,\  
\\  
/search/open_search,\  
/search/open_search_description.xml,\  
\\  
/shopping/notify,\  
\\  
/tags/rss,\  
\\  
/wiki/get_page_attachment,\  
/wiki/rss
```

Auto Login

Input a list of comma delimited class names that implement `com.liferay.portal.security.auth.AutoLogin`. These classes will run in consecutive order for all unauthenticated users until one of them return a valid user id and password combination. If no valid combination is returned, then the request continues to process normally. If a valid combination is returned, then the portal will automatically login that user with the returned user id and password combination.

For example, `com.liferay.portal.security.auth.RememberMeAutoLogin` reads from a cookie to automatically log in a user who previously logged in while checking the *Remember Me* box.

This interface allows deployers to easily configure the portal to work with other SSO servers. See `com.liferay.portal.security.auth.CASAutoLogin` for an example of how to configure the portal with Yale's SSO server.

```
auto.login.hooks=com.liferay.portal.security.auth.CASAutoLogin,com.liferay.portal.security.auth.NtlmAutoLogin,com.liferay.portal.security.auth.OpenIdAutoLogin,com.liferay.portal.security.auth.OpenSSOAutoLogin,com.liferay.portal.security.auth.RememberMeAutoLogin,com.liferay.portal.security.auth.SiteMindeAutoLogin
```

Set the hosts that will be ignored for auto login.

```
auto.login.ignore.hosts=
```

Set the paths that will be ignored for auto login.

```
auto.login.ignore.paths=
```

SSO with MAC (Message Authentication Code)

To use SSO with MAC, post to an URL like:

```
http://localhost:8080/c/portal/login?cmd=already-registered&login=<userId|emailAddress>&password=<MAC>
```

Pass the MAC in the password field. Make sure the MAC gets URL encoded because it might contain characters not allowed in a URL.

SSO with MAC also requires that you set the following property in system.properties:

```
com.liferay.util.servlet.SessionParameters=false
```

See the following links:

<http://issues.liferay.com/browse/LEP-1288>

http://en.wikipedia.org/wiki/Message_authentication_code

Set the following to true to enable SSO with MAC.

```
auth.mac.allow=false
```

Set the algorithm to use for MAC encryption.

```
auth.mac.algorithm=MD5
```

Set the shared key used to generate the MAC.

```
auth.mac.shared.key=
```

Passwords

Set the following encryption algorithm to encrypt passwords. The default algorithm is SHA (SHA-1). If set to NONE, passwords are stored in the database as plain text. The SHA-512 algorithm is currently unsupported.

Examples:

```
passwords.encryption.algorithm=CRYPT  
passwords.encryption.algorithm=MD2  
passwords.encryption.algorithm=MD5  
passwords.encryption.algorithm=NONE  
passwords.encryption.algorithm=SHA  
passwords.encryption.algorithm=SHA-256  
passwords.encryption.algorithm=SHA-384  
passwords.encryption.algorithm=SSHA
```

Digested passwords are encoded via base64 or hex encoding. The default is base64.

```
passwords.digest.encoding=base64  
#passwords.digest.encoding=hex
```

Input a class name that extends `com.liferay.portal.security.pwd.BasicToolkit`. This class will be called to generate and validate passwords.

Examples:

```
passwords.toolkit=com.liferay.portal.security.pwd.PasswordPolicyToolkit  
passwords.toolkit=com.liferay.portal.security.pwd.RegExpToolkit
```

If you choose to use `com.liferay.portal.security.pwd.PasswordPolicyToolkit` as your password toolkit, you can choose either static or dynamic password generation. Static is set through the property `passwords.passwordpolicytoolkit.static` and dynamic uses the class `com.liferay.util.PwdGenerator` to generate the password. If you are using LDAP password syntax checking, you will also have to use the static generator so that you can guarantee that passwords obey its rules.

Examples:

```
passwords.passwordpolicytoolkit.generator=static  
passwords.passwordpolicytoolkit.generator=dynamic  
passwords.passwordpolicytoolkit.static=iheartliferay
```

If you choose to use `com.liferay.portal.security.pwd.RegExpToolkit` as your password toolkit, set the regular expression pattern that will be used to generate and validate passwords.

Note that \ is replaced with \\ to work in Java.

The first pattern ensures that passwords must have at least 4 valid characters consisting of digits or letters.

The second pattern ensures that passwords must have at least 8 valid characters consisting of digits or letters.

Examples:

```
passwords.regexptoolkit.pattern=(?=.{4})(?:[a-zA-Z0-9]*)  
passwords.regexptoolkit.pattern=(?=.{8})(?:[a-zA-Z0-9]*)
```

Set the length and key for generating passwords.

Examples:

```
passwords.regexptoolkit.charset=0123456789  
passwords.regexptoolkit.charset=0123456789ABCDEFGHIJKLMNPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
```

Examples:

```
passwords.regexToolkit.length=4  
passwords.regexToolkit.length=8
```

Set the name of the default password policy.

```
passwords.default.policy.name=Default Password Policy
```

Permissions

Set the default permission checker class used by `com.liferay.portal.security.permission.PermissionCheckerFactory` to check permissions for actions on objects. This class can be overridden with a custom class that extends `com.liferay.portal.security.permission.PermissionCheckerImpl`.

```
permissions.checker=com.liferay.portal.security.permission.PermissionCheckerImpl
```

Set the algorithm used to check permissions for a user. This is useful so that you can optimize the search for different databases. See `com.liferay.portal.service.impl.PermissionLocalServiceImpl`. The default is method two.

The first algorithm uses several *if* statements to query the database for these five things in order. If it finds any one of them, it returns *true*:

- Is the user connected to one of the permissions via group or organization roles?
- Is the user associated with groups or organizations that are directly connected to one of the permissions?
- Is the user connected to one of the permissions via user roles?
- Is the user connected to one of the permissions via user group roles?
- Is the user directly connected to one of the permissions?

The second algorithm (the default) does a database join and checks the permissions in one step, by calling `countByGroupsRoles`, `countByGroupsPermissions`, `countByUsersRoles`, `countByUserGroupRole`, and `countByUsersPermissions` in one method.

The third algorithm checks the permissions by checking for three things. It combines the role check into one step. If it finds any of the following items, it returns *true*:

- Is the user associated with groups or organizations that are directly connected to one of the permissions?
- Is the user associated with a role that is directly connected to one of the permissions?
- Is the user directly connected to one of the permissions?

The fourth algorithm does a database join and checks the permissions that algorithm three checks in one step, by calling `countByGroupsPermissions`, `countByRolesPermissions`, and `countByUsersPermissions` in one method.

Algorithm 5 moves to a completely role-based permissions check for better performance. Permissions by users are no longer supported, yet it uses the same table structure as Algorithms 1-4.

Algorithm 6 is the current algorithm for Liferay 6 and above. It supports role-based permissions like Algorithm 5, but does so by using only one table and bitwise operations. This makes it perform far better than the other algorithms.

```
permissions.user.check.algorithm=1  
permissions.user.check.algorithm=2  
permissions.user.check.algorithm=3  
permissions.user.check.algorithm=4  
permissions.user.check.algorithm=5  
permissions.user.check.algorithm=6
```

Set the default permissions list filter class. This class must implement `com.liferay.portal.kernel.security.permission.PermissionsListFilter`. This is used if you want to filter the list of permissions before it is actually persisted. For example, if you want to make sure that all users who create objects never have the UPDATE action, then you can filter that list and remove any permissions that have the UPDATE action before it is persisted.

```
permissions.list.filter=com.liferay.portal.security.permission.PermissionsListFilterImpl
```

Set this to true to configure permission caching to block. See the property `ehcache.blocking.cache.allowed` for more information.

```
permissions.object.blocking.cache=false
```

The permissions cache uses a thread local map to store the most frequently accessed items to lower the number of queries to the underlying cache. Set the maximum map size to 0 to disable the thread level cache.

```
permissions.thread.local.cache.max.size=100
```

Set the following to true to automatically check the view permission on parent categories or folders when checking the permission on an specific item.

For example, if set to true, to be able to have access to a document, a user must have the view permission on the document's folder and all its parent folders. Or, to have access to a comment, a user must have the view permission on the comments's category and all its parent categories.

```
permissions.view.dynamic.inheritance=true
```

Captcha

Set the maximum number of captcha checks per portlet session. Set this value to 0 to always check. Set this value to a number less than 0 to never check. Unauthenticated users will always be checked on every request if captcha checks is enabled.

```
captcha.max.challenges=1
```

Set whether or not to use captcha checks for the following actions.

```
captcha.check.portal.create_account=true  
captcha.check.portal.send_password=true  
captcha.check.portlet.message_boards.edit_category=false  
captcha.check.portlet.message_boards.edit_message=false
```

Set the engine used to generate captchas. reCAPTCHA uses an external service that must be configured independently but provides an audible alternative which makes the captcha accessible to the visually impaired.

```
captcha.engine.impl=com.liferay.portal.captcha.recaptcha.ReCaptchaImpl  
captcha.engine.impl=com.liferay.portal.captcha.simplecaptcha.SimpleCaptchaImpl  
captcha.engine.recaptcha.key.private=  
captcha.engine.recaptcha.key.public=  
captcha.engine.recaptcha.url.script=http://api.recaptcha.net/challenge?k=  
captcha.engine.recaptcha.url.noscript=http://api.recaptcha.net/noscript?k=  
captcha.engine.recaptcha.url.verify=http://api-verify.recaptcha.net/verify
```

SimpleCaptcha

Set the height and width for captcha images generated by SimpleCaptcha.

```
captcha.engine.simplecaptcha.height=50  
captcha.engine.simplecaptcha.width=150
```

Input a list of comma delimited class names that implement `n1.captcha.backgrounds.BackgroundProducer`. These classes will be randomly used by SimpleCaptcha to generate a background for a captcha image.

```
captcha.engine.simplecaptcha.background.producers=n1.captcha.backgrounds.FlatColorBackgroundProducer,n1.captcha.backgrounds.GradiatedBackgroundProducer,  
n1.captcha.backgrounds.SquigglesBackgroundProducer,n1.captcha.backgrounds.TransparentBackgroundProducer
```

Input a list of comma delimited class names that implement `n1.captcha.gimpy.GimpyRenderer`. These classes will be randomly used by SimpleCaptcha to gimp a captcha image.

```
captcha.engine.simplecaptcha.gimpy.renderers=nlcaptcha.gimpy.RippleGimpyRen  
derer  
captcha.engine.simplecaptcha.gimpy.renderers=nlcaptcha.gimpy.BlockGimpyRen  
derer,nlcaptcha.gimpy.DropShadowGimpyRenderer,nlcaptcha.gimpy.FishEyeGimpyR  
enderer,nlcaptcha.gimpy.RippleGimpyRenderer,nlcaptcha.gimpy.ShearGimpyRend  
erer
```

Input a list of comma delimited class names that implement `nl.-captcha.noise.NoiseProducer`. These classes will be randomly used by SimpleCaptcha to add noise to a captcha image.

```
captcha.engine.simplecaptcha.noise.producers=nlcaptcha.noise.CurvedLineNois  
eProducer  
captcha.engine.simplecaptcha.noise.producers=nlcaptcha.noise.CurvedLineNois  
eProducer,nlcaptcha.noise.StraightLineNoiseProducer
```

Input a list of comma delimited class names that implement `nl.-captcha.text.producer.TextProducer`. These classes will be randomly used by SimpleCaptcha to generate text for a captcha image.

```
captcha.engine.simplecaptcha.text.producers=com.liferay.portal.captcha.simpl  
ecaptcha.PinNumberTextProducer  
captcha.engine.simplecaptcha.text.producers=com.liferay.portal.captcha.simpl  
ecaptcha.DictionaryWordTextProducer,com.liferay.portal.captcha.simplecaptcha  
.PinNumberTextProducer,nlcaptcha.text.producer.DefaultTextProducer,nlcapt  
cha.text.producer.FiveLetterFirstNameTextProducer
```

Input a list of comma delimited class names that implement `nl.-captcha.text.renderer.WordRenderer`. These classes will be randomly used by SimpleCaptcha to render text for a captcha image.

```
captcha.engine.simplecaptcha.word.renderers=nlcaptcha.text.renderer.Default  
WordRenderer  
captcha.engine.simplecaptcha.word.renderers=nlcaptcha.text.renderer.Colored  
EdgesWordRenderer,nlcaptcha.text.renderer.DefaultWordRenderer
```

Startup Events

Input a list of comma delimited class names that extend `com.liferay.-portal.struts.SimpleAction`. These classes will run at the specified event.

The following is a global startup event that runs once when the portal initializes.

```
global.startup.events=com.liferay.portal.events.GlobalStartupAction
```

The following is an application startup event that runs once for every web site instance of the portal that initializes.

```
application.startup.events=com.liferay.portal.events.AppStartupAction  
#application.startup.events=com.liferay.portal.events.AppStartupAction,com.lif  
eray.portal.events.SampleAppStartupAction
```

Shutdown Events

Input a list of comma delimited class names that extend com.liferay.-portal.struts.SimpleAction. These classes will run at the specified event.

Global shutdown event that runs once when the portal shuts down.

```
global.shutdown.events=com.liferay.portal.events.GlobalShutdownAction
```

Application shutdown event that runs once for every web site instance of the portal that shuts down.

```
application.shutdown.events=com.liferay.portal.events.AppShutdownAction
```

Programmatically kill the Java process on shutdown. This is a work-around for a bug in Tomcat and Linux where the process hangs on forever.

See <http://issues.liferay.com/browse/LEP-2048> for more information.

```
shutdown.programmatically.exit=false
```

Portal Events

Input a list of comma delimited class names that extend com.liferay.-portal.struts.Action. These classes will run before or after the specified event.

Servlet service event: the pre-service events have an associated error page and will forward to that page if an exception is thrown during execution of the events. The pre-service events process before Struts processes the request.

Examples:

```
servlet.service.events.pre=com.liferay.portal.events.ServicePreAction  
servlet.service.events.pre=com.liferay.portal.events.LogMemoryUsageAction,com.liferay.portal.events.LogThreadCountAction,com.liferay.portal.events.ServicePreAction  
servlet.service.events.pre=com.liferay.portal.events.LogSessionIdAction,com.liferay.portal.events.ServicePreAction  
servlet.service.events.pre=com.liferay.portal.events.ServicePreAction,com.liferay.portal.events.RandomLayoutAction  
servlet.service.events.pre=com.liferay.portal.events.ServicePreAction,com.liferay.portal.events.RandomLookAndFeelAction
```

Use the following to define the error page.

```
servlet.service.events.pre.error.page=/common/error.jsp
```

The post-service events process after Struts processes the request.

```
servlet.service.events.post=com.liferay.portal.events.ServicePostAction
```

Login event

Define events that can occur pre-login and post-login.

```
login.events.pre=com.liferay.portal.events.LoginPreAction  
login.events.post=com.liferay.portal.events.LoginPostAction,com.liferay.portal.events.DefaultLandingPageAction
```

Logout event

Similarly, events can be defined for the log out event.

```
logout.events.pre=com.liferay.portal.events.LogoutPreAction
```

Example post events:

```
logout.events.post=com.liferay.portal.events.LogoutPostAction  
logout.events.post=com.liferay.portal.events.LogoutPostAction,com.liferay.portal.events.DefaultLogoutPageAction,com.liferay.portal.events.SiteMinderLogoutAction  
#logout.events.post=com.liferay.portal.events.LogoutPostAction,com.liferay.portal.events.GarbageCollectorAction
```

Default Landing Page

Set the default landing page path for logged in users relative to the server path. This is the page users are automatically redirected to after logging in. For example, if you want the default landing page to be `http://localhost:8080/web/guest/login`, set this to `/web/guest/login`. To activate this feature, set `auth.forward.by.last.path` to true. To customize the behavior, see `com.liferay.portal.events.DefaultLandingPageAction` in the `login.events.post` property above.

```
#default.landing.page.path=/web/guest/login
```

Default Logout Page

Set the default logout page path for users relative to the server path. This is the page users are automatically redirected to after logging out. For example, if you want the default logout page to be `http://localhost:8080/web/guest/logout`, set this to `/web/guest/logout`. To activate this feature, set `auth.forward.by.last.path` to true. To customize the behavior, see `com.liferay.portal.events.DefaultLogoutPageAction` in the `logout.events.post` property above.

```
#default.logout.page.path=/web/guest/logout
```

Default Guest Public Layouts

The Guest group must have at least one public page. The settings for the initial public page are specified in the following properties.

If you need to add more than one page, set the property `default.guest.public.layout.lar` to specify a LAR file instead.

For even more complex behavior, override the `addDefaultGuestPublicLayouts` method in `com.liferay.portal.service.impl.GroupLocalServiceImpl`.

Set the name of the public layout.

```
default.guest.public.layout.name=Welcome
```

Set the layout template id of the public layout.

```
default.guest.public.layout.template.id=2_columns_ii
```

Set the portlet ids for the columns specified in the layout template.

```
default.guest.public.layout.column-1=58  
default.guest.public.layout.column-2=47  
default.guest.public.layout.column-3=  
default.guest.public.layout.column-4=
```

Set the friendly url of the public layout.

```
default.guest.public.layout.friendly.url=/home
```

Set the regular theme id for the public layout.

```
#default.guest.public.layout.regular.theme.id=classic
```

Set the regular color scheme id for the public layout.

```
#default.guest.public.layout.regular.color.scheme.id=01
```

Set the wap theme id for the public layout.

```
#default.guest.public.layout.wap.theme.id=mobile
```

Set the wap color scheme for the public layout.

```
#default.guest.public.layout.wap.color.scheme.id=01
```

Specify a LAR file that can be used to create the guest public layouts. If this property is set, the previous layout properties will be ignored.

```
#default.guest.public.layouts.lar=$  
{liferay.home}/deploy/default_guest_public.lar
```

Default User Private Layouts

If the properties `layout.user.private.layouts.enabled` and `layout.user.private.layouts.auto.create` are both set to true, then users will have private layouts and they will be automatically created. The settings below are used for the creation of the initial private pages.

If you need to add more than one page, set the property `default.user.private.layout.lar` to specify a LAR file instead.

For even more complex behavior, override the `addDefaultUserPrivateLayouts` method in `com.liferay.portal.events.ServicePreAction`.

Set the name of the private layout.

```
default.user.private.layout.name=Welcome
```

Set the layout template id of the private layout.

```
default.user.private.layout.template.id=2_columns_i1
```

Set the portlet ids for the columns specified in the layout template.

```
default.user.private.layout.column-1=71_INSTANCE_OY0d,82,23,61  
default.user.private.layout.column-2=11,29,8,19  
default.user.private.layout.column-3=  
default.user.private.layout.column-4=
```

Set the friendly url of the private layout.

```
default.user.private.layout.friendly.url=/home
```

Set the regular theme id for the private layout.

```
#default.user.private.layout.regular.theme.id=classic
```

Set the regular color scheme id for the private layout.

```
#default.user.private.layout.regular.color.scheme.id=01
```

Set the wap theme id for the private layout.

```
#default.user.private.layout.wap.theme.id=mobile
```

Set the wap color scheme for the private layout.

```
#default.user.private.layout.wap.color.scheme.id=01
```

Specify a LAR file that can be used to create the user private layouts. If this property is set, the previous layout properties will be ignored.

```
#default.user.private.layouts.lar= \  
 ${liferay.home}/deploy/default_user_private.lar
```

Default User Public Layouts

If the properties `layout.user.public.layouts.enabled` and `layout.user.public.layouts.auto.create` are both set to true, then users will have public layouts and they will be automatically created. The settings below are used for the creation of for the initial public pages.

If you need to add more than one page, set the property `default.user.public.layout.lar` to specify a LAR file instead.

For even more complex behavior, override the `addDefaultUserPublicLayouts` method in `com.liferay.portal.events.ServicePreAction`.

Set the name of the public layout.

```
default.user.public.layout.name=Welcome
```

Set the layout template id of the public layout.

```
default.user.public.layout.template.id=2_columns_ii
```

Set the portlet ids for the columns specified in the layout template.

```
default.user.public.layout.column-1=82,23  
default.user.public.layout.column-2=8,19  
default.user.public.layout.column-3=  
default.user.public.layout.column-4=
```

Set the friendly url of the public layout.

```
default.user.public.layout.friendly.url=/home
```

Set the regular theme id for the public layout.

```
#default.user.public.layout.regular.theme.id=classic
```

Set the regular color scheme id for the public layout.

```
#default.user.public.layout.regular.color.scheme.id=01
```

Set the wap theme id for the public layout.

```
#default.user.public.layout.wap.theme.id=mobile
```

Set the wap color scheme for the public layout.

```
#default.user.public.layout.wap.color.scheme.id=01
```

Specify a LAR file that can be used to create the user public layouts. If this property is set, the previous layout properties will be ignored.

```
#default.user.public.layouts.lar=${liferay.home}/deploy/default_user_public.lar
```

Sanitizer

Set the name of a class that implements com.liferay.portal.kernel.sanitizer.Sanitizer. This class is used to sanitize content.

```
sanitizer.impl=com.liferay.portal.sanitizer.DummySanitizerImpl
```

Social Equity

Set the interval on which the CheckEquityLogMessageListener will run. The value is set in one minute increments.

```
social.equity.equity.log.check.interval=1440
```

Set this to true to enable social equity logs.

```
social.equity.equity.log.enabled=true
```

Vaadin

Specify the location of the portal wide Vaadin themes and widget set (client side JavaScript).

```
vaadin.resources.path=/html
```

Specify the base Vaadin theme to load automatically for all Vaadin portlets. A portlet can include an additional theme that is loaded after the shared theme.

```
vaadin.theme=reindeer
```

Specify the shared widget set (client side JavaScript) that is used by all Vaadin portlets running in the portal.

```
vaadin.widgetset=com.vaadin.portal.gwt.PortalDefaultWidgetSet
```

Default Admin

Set the default admin password.

```
default.admin.password=test
```

Set the default admin screen name prefix.

```
default.admin.screen.name=test
```

Set the default admin email address prefix.

```
default.admin.email.address.prefix=test
```

Set the default admin first name.

```
default.admin.first.name=Test
```

Set the default admin middle name.

```
default.admin.middle.name=
```

Set the default admin last name.

```
default.admin.last.name=Test
```

Layouts

Set the list of layout types. The display text of each of the layout types is set in *content/Language.properties* and prefixed with *layout.types*. You can create new layout types and specify custom settings for each layout type. End users input dynamic values as designed in the edit page. End users see the layout as designed in the view page. The generated URL can reference properties set in the edit page. Parentable layouts can contain child layouts. You can also specify a comma delimited list of configuration actions that will be called for your layout when it is updated or deleted.

```
layout.types=portlet,panel,embedded,article,url,link_to_layout
```

Set whether or not private layouts are enabled. Set whether or not private layouts are modifiable. Set whether or not private layouts should be auto created if a user has no private layouts. If private layouts are not enabled, the other two properties are assumed to be false.

```
layout.user.private.layouts.enabled=true  
layout.user.private.layouts.modifiable=true  
layout.user.private.layouts.auto.create=true
```

Set whether or not public layouts are enabled. Set whether or not public layouts are modifiable. Set whether or not public layouts should be auto created if a user has no public layouts. If public layouts are not enabled, the other two properties are assumed to be false.

```
layout.user.public.layouts.enabled=true  
layout.user.public.layouts.modifiable=true  
layout.user.public.layouts.auto.create=true
```

Default Settings Layouts

These settings allow you to define several attributes on layouts. You can also specify which JSPs are responsible for editing and viewing layouts. You will likely never need to change these settings.

```
layout.edit.page=/portal/layout/edit/portlet.jsp  
layout.view.page=/portal/layout/view/portlet.jsp  
layout.url=${liferay:mainPath}/portal/layout?p_l_id=${liferay:plid}  
layout.url.friendliable=true  
layout.parentable=true  
layout.sitemapable=true  
layout.first.pageable=true  
layout.configuration.action.update=  
layout.configuration.action.delete=
```

Settings for portlet layouts are inherited from the default settings.

```
layout.edit.page[portlet]=/portal/layout/edit/portlet.jsp  
layout.view.page[portlet]=/portal/layout/view/portlet.jsp  
layout.url[portlet]=${liferay:mainPath}/portal/layout?p_l_id=${liferay:plid}  
layout.url.friendliable[portlet]=true  
layout.parentable[portlet]=true  
layout.configuration.action.update[portlet]=  
layout.configuration.action.delete[portlet]=
```

Settings for panel layouts.

```
layout.edit.page[panel]=/portal/layout/edit/panel.jsp  
layout.view.page[panel]=/portal/layout/view/panel.jsp  
layout.url[panel]=${liferay:mainPath}/portal/layout?p_l_id=${liferay:plid}  
layout.url.friendliable[panel]=true
```

Advanced Liferay Configuration

```
layout.parentable[panel]=true  
layout.first.pageable[panel]=true
```

Settings for control_panel layouts.

```
layout.edit.page[control_panel]=/portal/layout/edit/control_panel.jsp  
layout.view.page[control_panel]=/portal/layout/view/control_panel.jsp  
layout.url[control_panel]="${liferay:mainPath}/portal/layout?p_l_id=${liferay:plid}  
layout.url.friendliable[control_panel]=true  
layout.parentable[control_panel]=true  
layout.first.pageable[control_panel]=true
```

Settings for embedded layouts.

```
layout.edit.page[embedded]=/portal/layout/edit/embedded.jsp  
layout.view.page[embedded]=/portal/layout/view/embedded.jsp  
layout.url[embedded]="${liferay:mainPath}/portal/layout?p_l_id=${liferay:plid}  
layout.url.friendliable[embedded]=true  
layout.parentable[embedded]=false  
layout.sitemapable[embedded]=true  
layout.first.pageable[embedded]=true  
layout.configuration.action.update[embedded]=  
layout.configuration.action.delete[embedded]=
```

Settings for article layouts.

```
layout.edit.page[article]=/portal/layout/edit/article.jsp  
layout.view.page[article]=/portal/layout/view/article.jsp  
layout.url.friendliable[article]=true  
layout.url[article]="${liferay:mainPath}/portal/layout?p_l_id=${liferay:plid}  
layout.parentable[article]=false  
layout.sitemapable[article]=true  
layout.first.pageable[article]=true  
layout.configuration.action.update[article]=com.liferay.portal.model.LayoutTypeArticleConfigurationUpdateAction  
layout.configuration.action.delete[article]=com.liferay.portal.model.LayoutTypeArticleConfigurationDeleteAction
```

Settings for URL layouts.

```
layout.edit.page[url]=/portal/layout/edit/url.jsp  
layout.view.page[url]=  
layout.url[url]="${url}"  
layout.url.friendliable[url]=true  
layout.parentable[url]=false  
layout.sitemapable[url]=false  
layout.first.pageable[url]=false  
layout.configuration.action.update[url]=  
layout.configuration.action.delete[url]=
```

Settings for page layouts.

```

layout.edit.page[link_to_layout]=/portal/layout/edit/link_to_layout.jsp
layout.view.page[link_to_layout]=
layout.url[link_to_layout]="${liferay:mainPath}/portal/layout?p_l_id=${linkToPlid}
layout.url.friendliable[link_to_layout]=true
layout.parentable[link_to_layout]=true
layout.sitemapable[link_to_layout]=false
layout.first.pageable[link_to_layout]=false
layout.configuration.action.update[link_to_layout]=
layout.configuration.action.delete[link_to_layout]=

```

Specify static portlets that cannot be moved and will always appear on every layout. Static portlets will take precedence over portlets that may have been dynamically configured for the layout.

For example, if you want the Hello World portlet to always appear at the start of the iteration of the first column for user layouts, set the property *layout.static.portlets.start.column-1[user]* to 47. If you want the Hello World portlet to always appear at the end of the second column for user layouts, set the property *layout.static.portlets.end.column-2[user]* to 47. You can input a list of comma delimited portlet ids to specify more than one portlet. If the portlet is instanceable, add the suffix *_INSTANCE_abcd* to the portlet id, where *abcd* is any random alphanumeric string.

The static portlets are fetched based on the properties controlled by custom filters using EasyConf. By default, the available filters are user, community, and organization.

```

layout.static.portlets.start.column-1[user]=3,6
layout.static.portlets.end.column-1[user]=14
layout.static.portlets.start.column-2[user]=71_INSTANCE_abcd,7
layout.static.portlets.end.column-2[user]=34,70
layout.static.portlets.start.column-3[user]=
layout.static.portlets.end.column-3[user]=

```

It is also possible to set static portlets based on the layout's friendly URL.

```

layout.static.portlets.start.column-1[user][/home]=3,6
layout.static.portlets.end.column-2[community][/home]=14

```

Set the static portlets for community layouts.

```

layout.static.portlets.start.column-1[community]=
layout.static.portlets.end.column-1[community]=
layout.static.portlets.start.column-2[community]=
layout.static.portlets.end.column-2[community]=
layout.static.portlets.start.column-3[community]=
layout.static.portlets.end.column-3[community]=

```

Set the static portlets for organization layouts.

```

layout.static.portlets.start.column-1[organization]=

```

```
layout.static.portlets.end.column-1[organization]=  
layout.static.portlets.start.column-2[organization]=  
layout.static.portlets.end.column-2[organization]=  
layout.static.portlets.start.column-3[organization]=  
layout.static.portlets.end.column-3[organization]=
```

Set the static portlets that will appear for every layout. See [/html/portal/layout/view/portlet.jsp](#) in the Liferay source code for the logic of when these portlets will be shown. For example, these portlets will only show for layouts that can contain portlets and are not in a pop up state.

```
layout.static.portlets.all=1_WAR_chatportlet
```

Set the private group, private user, and public servlet mapping for `com.liferay.portal.servlet.FriendlyURLServlet`. This value must match the servlet mapping set in web.xml.

For example, if the private group pages are mapped to `/group` and the group's friendly URL is set to `/guest` and the layout's friendly URL is set to `/company/community`, then the friendly URL for the page will be `http://www.liferay.com/group/guest/company/community`. Private group pages map to a community's private pages and are only available to authenticated users with the proper permissions.

For example, if the public pages are mapped to `/web` and the group or user's friendly URL is set to `/guest` and the layout's friendly URL is set to `/company/community`, then the friendly URL for the page will be `http://www.liferay.com/web/guest/company/community`. Public pages are available to unauthenticated users.

The friendly URLs for users, groups, and layouts can be set during runtime.

```
layout.friendly.url.private.group.servlet.mapping=/group  
layout.friendly.url.private.user.servlet.mapping=/user  
layout.friendly.url.public.servlet.mapping=/web
```

Redirect to this resource if the user requested a friendly URL that does not exist. Leave it blank to display nothing.

Note: For backward compatibility, this overrides the property `layout.show.http.status` for the 404 status code.

```
layout.friendly.url.page.not.found=/html/portal/404.html
```

Set the reserved keywords that cannot be used in a friendly URL.

```
layout.friendly.url.keywords=c,group,web,image,wsrp,page,public,private,rss,  
tags
```

Set the following to true if layouts should remember (across requests) that a window state was set to maximized.

```
layout.remember.request.window.state.maximized=false
```

Set the following to true if guest users should see the maximize window icon.

```
layout.guest.show.max.icon=false
```

Set the following to true if guest users should see the minimize window icon.

```
layout.guest.show.min.icon=false
```

Set the following to true if users are shown that they do not have access to a portlet. The portlet init parameter *show-portlet-access-denied* will override this setting.

```
layout.show.portlet.access.denied=true
```

Set the following to true if users are shown that a portlet is inactive. The portlet init parameter *show-portlet-inactive* will override this setting.

```
layout.show.portlet.inactive=true
```

Set the following to true if the portal should show HTTP status codes like 404 if the requested page is not found.

```
layout.show.http.status=true
```

Set the default layout template id used when creating layouts.

```
layout.default.template.id=2_columns_ll
```

Set the following to false to disable parallel rendering. You can also disable it on a per request basis by setting the attribute key *com.liferay.portal.util.WebKeys.PORTLET_PARALLEL_RENDER* to the Boolean.FALSE in a pre service event or by setting the URL parameter *p_p_parallel* to 0.

```
layout.parallel.render.enable=true
```

Set the name of a class that implements *com.liferay.portal.util.LayoutClone*. This class is used to remember maximized and minimized states on shared pages. The default implementation persists the state in the browser session.

```
layout.clone.impl=com.liferay.portal.util.SessionLayoutClone
```

Set the following to true to cache the content of layout templates. This is recommended because it improves performance for production servers. Setting it to false is useful during development if you need to make a lot of changes.

```
layout.template.cache.enabled=true
```

Set the default value for the *p_l_reset* parameter. If set to true, then render parameters are cleared when different pages are hit. This is not the behavior promoted by the portlet specification, but is the one that most end users seem to prefer.

```
layout.default.p_1_reset=true
```

Portlet URL

Set the following to true if calling `setParameter` on a portlet URL appends the parameter value versus replacing it. There is some disagreement in the interpretation of the JSR 168 spec among portlet developers over this specific behavior. Liferay Portal successfully passes the portlet TCK tests whether this value is set to true or false.

See <http://issues.liferay.com/browse/LEP-426> for more information.

```
portlet.url.append.parameters=false
```

Set the following to true to allow portlet URLs to generate with an anchor tag.

```
portlet.url.anchor.enable=false
```

JSR 286 specifies that portlet URLs are escaped by default. Set this to false to provide for better backwards compatibility.

If this is set to true, but a specific portlet application requires that its portlet URLs not be escaped by default, then modify `portlet.xml` and set the container runtime option `javax.portlet.escapeXml` to false.

```
portlet.url.escape.xml=false
```

Preferences

Set the following to true to validate portlet preferences on startup.

```
preference.validate.on.startup=false
```

Struts

Input the custom Struts request processor that will be used by Struts based portlets. The custom class must extend `com.liferay.portal.struts.PortletRequestProcessor` and have the same constructor.

```
struts.portlet.request.processor=com.liferay.portal.struts.PortletRequestProcessor
```

Redirect

Set this property to `ip` or `domain` for the redirect security method. If set to `domain`, the portal will only redirect users to domains listed in the property `redirect.url.domain.allowed`. If set to `ip`, the portal will only redirect to domains whose IP address resolve to an IP address listed in the property `redirect.url.ip.allowed`.

```
redirect.url.security.mode=domain  
redirect.url.security.mode=ip
```

Input a list of comma delimited domains which the portal is allowed to redirect to. Input a blank list to allow any domain.

```
redirect.url.domains.allowed=
```

Input a list of comma delimited IPs which the portal is allowed to redirect to. Input a blank list to allow any IP. SERVER_IP will be replaced with the IP of the host server.

```
redirect.url.ips.allowed=127.0.0.1, SERVER_IP
```

Images

Set the location of the default spacer image that is used for missing images. This image must be available in the class path.

```
image.default.spacer=com/liferay/portal/dependencies/spacer.gif
```

Set the location of the default company logo image that is used for missing company logo images. This image must be available in the class path.

```
image.default.company.logo=com/liferay/portal/dependencies/company_logo.png
```

Set the location of the default organization logo image that is used for missing organization logo images. This image must be available in the class path.

```
image.default.organization.logo=com/liferay/portal/dependencies/organization_logo.png
```

Set the locations of the default user portrait images that are used for missing user portrait images. This image must be available in the class path.

```
image.default.user.female.portrait=com/liferay/portal/dependencies/user_female_portrait.png  
image.default.user.male.portrait=com/liferay/portal/dependencies/user_male_portrait.png
```

Set the name of a class that implements com.liferay.portal.image.Hook. The portal will use this persist images.

Available hooks are:

- com.liferay.portal.image.DatabaseHook
- com.liferay.portal.image.DLHook
- com.liferay.portal.image.FileSystemHook

```
image.hook.impl=com.liferay.portal.image.DatabaseHook  
#image.hook.impl=com.liferay.portal.image.DLHook  
#image.hook.impl=com.liferay.portal.image.FileSystemHook
```

FileSystemHook

```
image.hook.file.system.root.dir=${liferay.home}/data/images
```

Editors

You can configure individual JSP pages to use a specific implementation of the available WYSIWYG editors: liferay, fckeditor, simple, tinymce, or tinymcesimple.

```
editor.wysiwyg.default=fckeditor
editor.wysiwyg.portal-
web.docroot.html.portlet.blogs.edit_entry.jsp=fckeditor
editor.wysiwyg.portal-
web.docroot.html.portlet.calendar.edit_configuration.jsp=fckeditor
editor.wysiwyg.portal-
web.docroot.html.portlet.enterprise_admin.view.jsp=fckeditor
editor.wysiwyg.portal-
web.docroot.html.portlet.invitation.edit_configuration.jsp=fckeditor
editor.wysiwyg.portal-
web.docroot.html.portlet.journal.edit_article_content.jsp=fckeditor
editor.wysiwyg.portal-
web.docroot.html.portlet.journal.edit_article_content_xsd_el.jsp=fckeditor
editor.wysiwyg.portal-
web.docroot.html.portlet.journal.edit_configuration.jsp=fckeditor
editor.wysiwyg.portal-
web.docroot.html.portlet.login.configuration.jsp=fckeditor
editor.wysiwyg.portal-web.docroot.html.portlet.mail.edit.jsp=fckeditor
editor.wysiwyg.portal-
web.docroot.html.portlet.mail.edit_message.jsp=fckeditor
editor.wysiwyg.portal-
web.docroot.html.portlet.message_boards.edit_configuration.jsp=fckeditor
editor.wysiwyg.portal-
web.docroot.html.portlet.shopping.edit_configuration.jsp=fckeditor
editor.wysiwyg.portal-web.docroot.html.portlet.wiki.edit_html.jsp=fckeditor
```

Fields

Set the following fields to false so users cannot see them. Some company policies require gender and birthday information to always be hidden.

```
field.enable.com.liferay.portal.model.Contact.male=true
field.enable.com.liferay.portal.model.Contact.birthday=true
field.enable.com.liferay.portal.model.Organization.status=false
```

Input a list of comma delimited user types who can edit their own fields.

Valid types are administrator, user-mx, and user-without-mx.

Set a value of administrator if an administrator can edit the specified field. An administrator is anyone who has the Administrator role.

Set a value of user-mx if a user who has an email address that matches the company mail suffix can edit the specified field.

Set a value of user-without-mx if a user who does not have an email address that matches the company mail suffix can edit the specified field.

Set all three values if all users can edit the specified field. Set a combination of the three values if only a combination of the users can edit the specified field.

```
field.editable.com.liferay.portal.model.User.screenName=administrator,user-with-mx,user-without-mx  
field.editable.com.liferay.portal.model.User.emailAddress=administrator,user-with-mx,user-without-mx
```

Mime Types

Input a list of comma delimited mime types that are not available by default from javax.activation.MimetypesFileTypeMap.

```
mime.types=\n        application/msword doc,\n        application/pdf pdf,\n        application/vnd.ms-excel xls,\n        application/vnd.ms-powerpoint ppt,\n        application/x-ms-wmp wmv,\n        application/x-shockwave-flash swf flv
```

Input a list of comma delimited extensions for which the content disposition header has to be set to *inline*.

```
mime.types.content.disposition.inline=flv,pdf,swf,wmv
```

Amazon

Enter an Amazon access key ID and an Amazon associate tag. This is made available only for personal use. Please see the Amazon's license at <http://www.amazon.com> for more information.

```
#amazon.access.key.id=\n#amazon.associate.tag=
```

Browser Launcher

Enter a URL to automatically launch a browser to that URL when the portal has fully initialized. Enter a blank URL to disable this feature.

```
browser.launcher.url=http://localhost:8080
```

Control Panel

Set the name of the layout.

```
control.panel.layout.name=Control Panel
```

Set the friendly URL of the layout.

```
control.panel.layout.friendly.url=/manage
```

Set the theme of the layout.

```
control.panel.layout.regular.theme.id=controlpanel
```

Set the maximum number of communities that will be shown in the navigation menus. A large value might cause performance problems if the number of communities that the user can administer is very large.

```
control.panel.navigation.max.communities=50
```

Set the maximum number of organizations that will be shown in the navigation menus. A large value might cause performance problems if the number of organizations that the user can administer is very large.

```
control.panel.navigation.max.organizations=50
```

Set the name of a class that implements `com.liferay.portlet.ControlPanelEntry`. This class denotes the default value of the element `control-panel-entry-class` in `liferay-portlet.xml` and is called by the Control Panel to decide whether the portlet should be shown to a specific user in a specific context.

```
control.panel.default.entry.class=com.liferay.portlet.DefaultControlPanelEntry
```

Instant Messenger

Set the AIM login and password which the system will use to communicate with users.

```
aim.login=
aim.password=
```

Due to a bug in JOScarLib 0.3b1, you must set the full path to the ICQ jar.

See the following posts:

http://sourceforge.net/forum/message.php?msg_id=1972697

http://sourceforge.net/forum/message.php?msg_id=1990487

```
icq.jar=C:/Java/orion-2.0.7/lib/icq.jar
```

Set the ICQ login and password which the system will use to communicate with users.

```
icq.login=
```

```
icq.password=
```

Set the MSN login and password which the system will use to communicate with users.

```
msn.login=
```

```
msn.password=
```

Set the YM login and password which the system will use to communicate with users.

```
ym.login=
```

```
ym.password=
```

Lucene Search

Set the limit for results used when performing index searches that are subsequently filtered by permissions.

```
index.filter.search.limit=5000
```

Set the following to true if you want to avoid any writes to the index. This is useful in some clustering environments where there is a shared index and only one node of the cluster updates it.

```
index.read.only=false
```

Set the following to true if you want to index your entire library of files on startup.

```
index.on.startup=false
```

Set this to true to add a delay before indexing on startup. A delay may be necessary if a lot of plugins need to be loaded and reindexed. This property is only valid if *index.on.startup* is set to true.

```
index.on.startup.delay=60
```

Set this to true if you want to index your entire library of files after an upgrade. Only set this property to false if you are running a small upgrade and you do not need to reindex everything.

```
index.on.upgrade=true
```

Set the following to true if you want the indexing on startup to be executed on a separate thread to speed up execution.

```
index.with.thread=true
```

Designate whether Lucene stores indexes in a database via JDBC, file system, or in RAM.

Examples:

```
lucene.store.type=jdbc
```

```
lucene.store.type=file
```

```
lucene.store.type=ram
```

Lucene's storage of indexes via JDBC has a bug where temp files are not removed. This can eat up disk space over time. Set the following property to true to automatically clean up the temporary files once a day. See LEP-2180.

```
lucene.store.jdbc.auto.clean.up=true
```

Set the JDBC dialect that Lucene uses to store indexes in the database. This is only referenced if Lucene stores indexes in the database. Liferay will attempt to load the proper dialect based on the URL of the JDBC connection. For example, the property *lucene.store.jdbc.dialect.mysql* is read for the JDBC connection URL *jdbc:mysql://localhost/lportal*.

```
lucene.store.jdbc.dialect.db2=org.apache.lucene.store.jdbc.dialect.DB2Dialect  
lucene.store.jdbc.dialect.derby=org.apache.lucene.store.jdbc.dialect.DerbyDialect  
lucene.store.jdbc.dialect.hsqldb=org.apache.lucene.store.jdbc.dialect.HSQLDialect  
lucene.store.jdbc.dialect.jtds=org.apache.lucene.store.jdbc.dialect.SQLServerDialect  
lucene.store.jdbc.dialect.microsoft=org.apache.lucene.store.jdbc.dialect.SQLServerDialect  
lucene.store.jdbc.dialect.mysql=org.apache.lucene.store.jdbc.dialect.MySQLDialect  
#lucene.store.jdbc.dialect.mysql=org.apache.lucene.store.jdbc.dialect.MySQLNoDBDialect  
#lucene.store.jdbc.dialect.mysql=org.apache.lucene.store.jdbc.dialect.MySQLMyISAMDialect  
lucene.store.jdbc.dialect.oracle=org.apache.lucene.store.jdbc.dialect.OracleDialect  
lucene.store.jdbc.dialect.postgresql=org.apache.lucene.store.jdbc.dialect.PostgreSQLDialect
```

Set the directory where Lucene indexes are stored. This is only referenced if Lucene stores indexes in the file system.

```
lucene.dir=${liferay.home}/lucene/
```

Input a class name that extends *com.liferay.portal.search.lucene.LuceneFileExtractor*. This class is called by Lucene to extract text from complex files so that they can be properly indexed.

```
lucene.file.extractor=com.liferay.portal.search.lucene.LuceneFileExtractor
```

The file extractor can sometimes return text that is not valid for Lucene. This property expects a regular expression. Any character that does not match the regular expression will be replaced with a blank space. Set an empty regular expression to disable this feature.

Examples:

```
lucene.file.extractor.regexp.strip=  
lucene.file.extractor.regexp.strip=[\\d\\w]
```

Set the default analyzer used for indexing and retrieval.

Examples:

```
lucene.analyzer=org.apache.lucene.analysis.br.BrazilianAnalyzer  
lucene.analyzer=org.apache.lucene.analysis.cn.ChineseAnalyzer  
lucene.analyzer=org.apache.lucene.analysis.cjk.CJKAnalyzer  
lucene.analyzer=org.apache.lucene.analysis.cz.CzechAnalyzer  
lucene.analyzer=org.apache.lucene.analysis.nl.DutchAnalyzer  
lucene.analyzer=org.apache.lucene.analysis.fr.FrenchAnalyzer  
lucene.analyzer=org.apache.lucene.analysis.de.GermanAnalyzer  
lucene.analyzer=org.apache.lucene.analysis.KeywordAnalyzer  
lucene.analyzer=org.apache.lucene.index.memory.PatternAnalyzer  
lucene.analyzer=org.apache.lucene.analysis.PerFieldAnalyzerWrapper  
lucene.analyzer=org.apache.lucene.analysis.ru.RussianAnalyzer  
lucene.analyzer=org.apache.lucene.analysis.SimpleAnalyzer  
lucene.analyzer=org.apache.lucene.analysis.snowball.SnowballAnalyzer  
lucene.analyzer=org.apache.lucene.analysis.standard.StandardAnalyzer  
lucene.analyzer=org.apache.lucene.analysis.StopAnalyzer  
lucene.analyzer=org.apache.lucene.analysis.WhitespaceAnalyzer
```

Set how often index updates will be committed. Set the batch size to configure how many consecutive updates will trigger a commit. If the value is 0, then the index will be committed on every update. Set the time interval in milliseconds to configure how often to commit the index. The time interval is not read unless the batch size is greater than 0 because the time interval works in conjunction with the batch size to guarantee that the index is committed after a specified time interval. Set the time interval to 0 to disable committing the index by a time interval.

```
lucene.commit.batch.size=0  
lucene.commit.time.interval=0
```

Set Lucene's buffer size in megabytes. Higher numbers mean indexing goes faster but uses more memory.

```
lucene.buffer.size=16
```

Set Lucene's merge factor. Higher numbers mean indexing goes faster but uses more memory. The default value from Lucene is 10. This should never be set to a number lower than 2.

```
lucene.merge.factor=10
```

Set how often to run Lucene's optimize method. Optimization speeds up searching but slows down writing. Set this property to 0 to always optimize. Set this property to an integer greater than 0 to optimize every X writes.

```
lucene.optimize.interval=1
```

Set this to true if you want to index your entire library of files after an upgrade. Only set this property to false if you are running a small upgrade

and you do not need to reindex everything.

```
index.on.upgrade=true
```

Set the interval on which the lucene automatic clean up is set run. The value is set in one minute increments.

```
lucene.store.jdbc.auto.clean.up.interval=1440
```

Set this to true if you want the portal to replicate an index write across all members of the cluster. This is useful in some clustered environments where you wish each server instance to have its own copy of the Lucene search index. This is only relevant when using the default Lucene indexing engine.

```
lucene.replicate.write=false
```

SourceForge

```
source.forge.mirrors=\n    http://downloads.sourceforge.net,\\" # Redirect\n    http://internap.dl.sourceforge.net,\\" # San Jose, CA\n    http://superb-east.dl.sourceforge.net,\\" # McLean, Virginia\n    http://superb-west.dl.sourceforge.net,\\" # Seattle, Washington\n    http://easynews.dl.sourceforge.net,\\" # Phoenix, AZ\n    http://kent.dl.sourceforge.net,\\" # Kent, UK\n    http://ufpr.dl.sourceforge.net,\\" # Curitiba, Brazil\n    http://belnet.dl.sourceforge.net,\\" # Brussels, Belgium\n    http://switch.dl.sourceforge.net,\\" # Lausanne, Switzerland\n    http://mesh.dl.sourceforge.net,\\" # Duesseldorf, Germany\n    http://ovh.dl.sourceforge.net,\\" # Paris, France\n    http://dfn.dl.sourceforge.net,\\" # Berlin, Germany\n    http://heanet.dl.sourceforge.net,\\" # Dublin, Ireland\n    http://garr.dl.sourceforge.net,\\" # Bologna, Italy\n    http://surfnet.dl.sourceforge.net \\" # Amsterdam, The Netherlands\n    http://jaist.dl.sourceforge.net,\\" # Ishikawa, Japan\n    http://nchc.dl.sourceforge.net,\\" # Tainan, Taiwan\n    http://optusnet.dl.sourceforge.net \\" # Sydney, Australia
```

Value Object

You can add a listener for a specific class by setting the property `value.object.listener` with a list of comma delimited class names that implement `com.liferay.portal.model.ModelListener`. These classes are pooled and reused and must be thread safe.

```
value.object.listener.com.liferay.portal.model.Contact=com.liferay.portal.mo\ndl.ContactListener\nvalue.object.listener.com.liferay.portal.model.Layout=com.liferay.portal.mod\el.LayoutListener\nvalue.object.listener.com.liferay.portal.model.LayoutSet=com.liferay.portal.\model.LayoutSetListener
```

```
value.object.listener.com.liferay.portal.model.PortletPreferences=com.liferay.portal.model.PortletPreferencesListener  
value.object.listener.com.liferay.portal.model.User=com.liferay.portal.model.UserListener  
value.object.listener.com.liferay.portlet.journal.model.JournalArticle=com.liferay.portlet.journal.model.JournalArticleListener  
value.object.listener.com.liferay.portlet.journal.model.JournalTemplate=com.liferay.portlet.journal.model.JournalTemplateListener
```

The finder level cache stores the many paths that return a value object and the many paths that return a list of value objects. The finder level cache only caches primary keys and is further helped by the entity level cache that caches the value object to the primary key.

The Hibernate level cache is provided by the `hibernate.cache.provider_class` property. Set this to true to enable entity level caching.

```
value.object.entity.cache.enabled=true
```

Set this to true to configure entity level caching to block. See the property `ehcache.blocking.cache.allowed` for more information.

```
value.object.entity.blocking.cache=true
```

The entity level cache uses a thread local map to store the most frequently accessed items to lower the number of queries to the underlying cache. Set the maximum map size to 0 to disable the thread level cache.

```
value.object.entity.thread.local.cache.max.size=100
```

Entity level caching for a specific type of value object can be configured by using a property name that includes the value object's class name.

```
value.object.entity.cache.enabled.com.liferay.portal.model.Layout=true  
value.object.entity.cache.enabled.com.liferay.portal.model.User=true  
value.object.entity.cache.enabled.com.liferay.portlet.social.model.SocialEquityAssetEntry=false  
value.object.entity.cache.enabled.com.liferay.portlet.social.model.SocialEquityLog=false  
value.object.entity.cache.enabled.com.liferay.portlet.social.model.SocialEquityUser=false
```

Set this to true to enable finder level caching.

```
value.object.finder.cache.enabled=true
```

Set this to true to configure finder level caching to block. See the property `ehcache.blocking.cache.allowed` for more information.

```
value.object.finder.blocking.cache=true
```

The finder level cache uses a thread local map to store the most frequently accessed items to lower the number of queries to the underlying cache. Set the maximum map size to 0 to disable the thread level cache.

```
value.object.finder.thread.local.cache.max.size=100
```

Finder level caching for a specific type of value object can be configured by using a property name that includes the value object's class name. Mapping tables can also be specified to configure the caching of value object relationships.

```
value.object.finder.cache.enabled.com.liferay.portal.model.Layout=true  
value.object.finder.cache.enabled.com.liferay.portal.model.User=true  
value.object.finder.cache.enabled.com.liferay.portlet.social.model.SocialEquityAssetEntry=false  
value.object.finder.cache.enabled.com.liferay.portlet.social.model.SocialEquityLog=false  
value.object.finder.cache.enabled.com.liferay.portlet.social.model.SocialEquityUser=false  
value.object.finder.cache.enabled.Users_Roles=true
```

Communication Link

Set the JGroups properties used by the portal to communicate with other instances of the portal. This is only needed if the portal is running in a clustered environment. The JGroups settings provide a mechanism for the portal to broadcast messages to the other instances of the portal. The specified multi-cast address should be unique for internal portal messaging only. You will still need to set the Hibernate and Ehcache settings for database clustering.

```
comm.link.properties=UDP(bind_addr=127.0.0.1;mcast_addr=231.12.21.102;mcast_port=45566;ip_ttl=32;mcast_send_buf_size=150000;mcast_recv_buf_size=80000):PING(timeout=2000;num_initial_members=3):MERGE2(min_interval=5000;max_interval=10000):FD_SOCK:VERIFY_SUSPECT(timeout=1500):pbcast.NAKACK(gc_lag=50,retransmit_timeout=300,600,1200,2400,4800;max_xmit_size=8192):UNICAST(timeout=300,600,1200,2400):pbcast.STABLE(desired_avg_gossip=20000):FRAG(frag_size=8096;down_thread=false;up_thread=false):pbcast.GMS(join_timeout=5000;join_retry_timeout=2000;shun=false;print_local_addr=true)
```

Cluster Link

Set this to true to enable the cluster link. This is required if you want to cluster indexing and other features that depend the cluster link.

```
cluster.link.enabled=false
```

Set the JGroups properties for each channel, we support up to 10 transport channels and 1 single required control channel. Use as few transport channels as possible for best performance. By default, only one UDP control channel and one UDP transport channel are enabled. Channels can be configured by XML files that are located in the class path or by inline properties.

```
cluster.link.channel.properties.control=UDP(bind_addr=localhost;mcast_address=${multicast.group.address["cluster-link-control"]};mcast_port=${multicast.group.port["cluster-link-control"]};ip_ttl=8;mcast_send_buf_size=150000;mcast_recv_buf_size=80000):PING(timeout=2000;num_initial_members=3):MERGE2(min_interval=5000;max_interval=10000):FD_SOCK:VERIFY_SUSPECT(timeout=1500):pbcast.NAKACK(gc_lag=50,retrans
```

```

mit_timeout=300,600,1200,2400,4800;max_xmit_size=8192):UNICAST(timeout=300,6
00,1200,2400):pbcast.STABLE(desired_avg_gossip=20000):FRAG(frag_size=8096;do
wn_thread=false;up_thread=false):pbcast.GMS(join_timeout=5000;join_retry_tim
eout=2000;shun=false;print_local_addr=true)
    cluster.link.channel.properties.transport.0=UDP(bind_addr=localhost;mca
st_addr=${multicast.group.address["cluster-link-udp"]};mcast_port=$
{multicast.group.port["cluster-link-
udp"]};ip_ttl=8;mcast_send_buf_size=150000;mcast_recv_buf_size=80000):PING(t
imeout=2000;num_initial_members=3):MERGE2(min_interval=5000;max_interval=100
00):FD_SOCK:VERIFY_SUSPECT(timeout=1500):pbcast.NAKACK(gc_lag=50;retransmit_
timeout=300,600,1200,2400,4800;max_xmit_size=8192):UNICAST(timeout=300,600,1
200,2400):pbcast.STABLE(desired_avg_gossip=20000):FRAG(frag_size=8096;down_t
hread=false;up_thread=false):pbcast.GMS(join_timeout=5000;join_retry_timeou
t=2000;shun=false;print_local_addr=true)
#cluster.link.channel.properties.transport.1=udp.xml
#cluster.link.channel.properties.transport.2=mping.xml

```

Set JGroups' system properties. System properties have higher priority than individual properties given to each channel. That means system properties will override individual properties.

```

cluster.link.channel.system.properties=\
#
# Common
#
\
jgroups.bind_addr:localhost,\
#jgroups.bind_interface:eth0,\
\
#
# Multicast
#
\
jgroups.mping.mcast_addr:${multicast.group.address["cluster-link-
mping"]},\
    jgroups.mping.mcast_port:${multicast.group.port["cluster-link-
mping"]},\
        jgroups.mping.ip_ttl:8,\
\
\

```

Cluster Executor

Set this to true to enable the cluster executor debugging. This will attach a debugging listener which will log every cluster event it receives.

```
cluster.executor.debug.enabled=false
```

Minifier

The strip filter will attempt to cache inline minified CSS and JavaScript content. Set this property configure the maximum pieces of cached content. Set this property to 0 to disable caching of inline minified content.

```
minifier.inline.content.cache.size=10000
```

Input a list of comma delimited values that will cause the minified CSS to not be cached if those values are contained in the content.

```
minifier.inline.content.cache.skip.css=
```

Input a list of comma delimited values that will cause the minified JavaScript to not be cached if those values are contained in the content.

```
minifier.inline.content.cache.skip.javascript=getSessionId,encryptedUserId
```

Monitoring

Configure the appropriate level for monitoring Liferay. Valid values are: HIGH, LOW, MEDIUM, OFF.

```
monitoring.level.com.liferay.monitoring.Portal=HIGH  
monitoring.level.com.liferay.monitoring.Portlet=HIGH
```

Set this to true to store data samples of the current request as a thread local variable. This allows you to obtain each request's statistics for further processing

```
monitoring.data.sample.thread.local=false
```

Set this to true to monitor portal requests.

```
monitoring.portal.request=false
```

Set this to true to monitor portlet action requests.

```
monitoring.portlet.action.request=false
```

Set this to true to monitor portlet event requests.

```
monitoring.portlet.event.request=false
```

Set this to true to monitor portlet render requests.

```
monitoring.portlet.render.request=false
```

Set this to true to monitor portlet resource requests.

```
monitoring.portlet.resource.request=false
```

Set this to true to show data samples at the bottom of each portal page. In order for data to show, the property `monitoring.data.sample.thread.local` must be set to true.

```
monitoring.show.per.request.data.sample=false
```

Multicast

Consolidate multicast address and port settings in one location for easier maintenance. These settings must correlate to your physical network configuration (i.e. firewall, switch, and other network hardware matter) to

ensure speedy and accurate communication across a cluster.

Each address and port combination represent a conversation that is made between different nodes. If they are not unique or correctly set, there will be a potential of unnecessary network traffic that may cause slower updates or inaccurate updates.

See the property `cluster.link.channel.properties.control`.

```
multicast.group.address["cluster-link-control"] = 233.0.0.1  
multicast.group.port["cluster-link-control"] = 23301
```

See the properties `cluster.link.channel.properties.transport.0` and `cluster.link.channel.system.properties`.

```
multicast.group.address["cluster-link-udp"] = 233.0.0.2  
multicast.group.port["cluster-link-udp"] = 23302
```

See the property `cluster.link.channel.system.properties`.

```
multicast.group.address["cluster-link-mping"] = 233.0.0.3  
multicast.group.port["cluster-link-mping"] = 23303
```

See the properties `net.sf.ehcache.configurationResourceName` and `net.sf.ehcache.configurationResourceName.peerProviderProperties`.

```
multicast.group.address["hibernate"] = 233.0.0.4  
multicast.group.port["hibernate"] = 23304
```

See the properties `ehcache.multi.vm.config.location` and `ehcache.multi.vm.config.location.peerProviderProperties`.

```
multicast.group.address["multi-vm"] = 233.0.0.5  
multicast.group.port["multi-vm"] = 23305
```

Content Delivery Network

Set the hostname that will be used to serve static content via a CDN. This property can be overridden dynamically at runtime by setting the HTTP parameter `cdn_host`.

```
cdn.host=
```

Counter

The counter operates with its own data source to prevent deadlocks. By default, the data source created for the counter uses the same settings as those used to create the data source used for the rest of the portal. That happens because the counter service will look up the properties prefixed with `jdb.default`. to create its data source. See the JDBC properties prefixed with `jdb.default`. for more information.

Setting a different value for the counter JDBC prefix allows you to better fine tune the counter data source with its own set of configuration settings for high availability installations. Note that these settings, though separate, are a copy of the default settings with the newly overridden values.

```
counter.jdbc.prefix=jdbc.default.
```

Set the number of increments between database updates to the Counter table. Set this value to a higher number for better performance.

```
counter.increment=100
```

Set the interval in minutes for the ConnectionHeartbeatJob. This will determine how often the database is polled for long running connections and will prevent the database from disconnecting the socket prematurely.

```
counter.connection.heartbeat.job.interval=60
```

Lock

Set the lock expiration time for each class.

Example: 1 Day

```
lock.expiration.time.com.liferay.portlet.documentlibrary.model.DLFileEntry=8  
6400000
```

Example: 20 Minutes

```
lock.expiration.time.com.liferay.portlet.wiki.model.WikiPage=1200000
```

JBI

Connect to either Mule or ServiceMix as your ESB.

Examples:

```
jbi.workflow.url=http://localhost:8080/mule-web/workflow  
jbi.workflow.url=http://localhost:8080/servicemix-web/workflow
```

JCR

Liferay includes Jackrabbit (<http://jackrabbit.apache.org>) by default as its JSR-170 Java Content Repository.

```
jcr.initialize.on.startup=false  
jcr.workspace.name=liferay  
jcr.node.documentlibrary=documentlibrary  
jcr.jackrabbit.repository.root=${liferay.home}/jackrabbit  
jcr.jackrabbit.config.file.path=${jcr.jackrabbit.repository.root}/repository.xml  
jcr.jackrabbit.repository.home=${jcr.jackrabbit.repository.root}/home  
jcr.jackrabbit.credentials.username=none  
jcr.jackrabbit.credentials.password=none
```

Live Users

Set this to true to enable tracking via Live Users.

```
live.users.enabled=false
```

Lock

Set the lock expiration time for each class.

1 day:

```
lock.expiration.time.com.liferay.portlet.documentlibrary.model.DLFolder=8640  
0000  
lock.expiration.time.com.liferay.portlet.documentlibrary.model.DLFileEntry=8  
6400000
```

20 minutes:

```
lock.expiration.time.com.liferay.portlet.wiki.model.WikiPage=1200000
```

Mail

Set the JNDI name to lookup the Java Mail session. If none is set, then the portal will attempt to create the Java Mail session based on the properties prefixed with *mail.session*.

```
#mail.session.jndi.name=mail/MailSession
```

Set the properties used to create the Java Mail session. The property prefix *mail.session.* will be removed before it is used to create the session object. These properties will only be read if the property *mail.session.jndi.name* is not set.

```
mail.session.mail.imap.host=localhost  
mail.session.mail.pop3.host=localhost  
#mail.session.mail.smtp.auth=true  
mail.session.mail.smtp.host=localhost  
#mail.session.mail.smtp.socketFactory.class=javax.net.ssl.SSLSocketFactory  
#mail.session.mail.smtp.socketFactory.fallback=false  
#mail.session.mail.smtp.socketFactory.port=465  
#mail.session.mail.smtp.starttls.enable=true  
#mail.session.mail.smtp.password=  
#mail.session.mail.smtp.port=465  
#mail.session.mail.smtp.user=  
mail.session.mail.store.protocol=localhost  
mail.session.mail.transport.protocol=smtp
```

Set this to false if administrator should not be allowed to change the mail domain via the Admin portlet.

```
mail.mx.update=true
```

Input a list of comma delimited email addresses that will receive a BCC of every email sent through the mail server.

```
mail.audit.trail=
```

Set the name of a class that implements com.liferay.mail.util.Hook. The mail server will use this class to ensure that the mail and portal servers are synchronized on user information. The portal will not know how to add, update, or delete users from the mail server except through this hook.

Available hooks are:

```
mail.hook.impl=com.liferay.mail.util.CyrusHook  
mail.hook.impl=com.liferay.mail.util.DummyHook  
mail.hook.impl=com.liferay.mail.util.FuseMailHook  
mail.hook.impl=com.liferay.mail.util.GoogleHook  
mail.hook.impl=com.liferay.mail.util.SendmailHook  
mail.hook.impl=com.liferay.mail.util.ShellHook
```

CyrusHook

Set the commands for adding, updating, and deleting a user where %1% is the user id. Replace the password with the password for the cyrus user.

```
mail.hook.cyrus.add.user=cyrusadmin password create %1%  
#mail.hook.cyrus.add.user=cyrus_adduser password %1%  
mail.hook.cyrus.delete.user=cyrusadmin password delete %1%  
#mail.hook.cyrus.delete.user=cyrus_userdel password %1%  
mail.hook.cyrus.home=/home/cyrus
```

FuseMailHook

See <http://www.fusemail.com/support/api.html> for more information. You must also update the *mail.account.finder* property.

```
mail.hook.fusemail.url=https://www.fusemail.com/api/request.html  
mail.hook.fusemail.username=  
mail.hook.fusemail.password=  
mail.hook.fusemail.account.type=group_subaccount  
mail.hook.fusemail.group.parent=
```

SendmailHook

Set the commands for adding, updating, and deleting a user where %1% is the user id and %2% is the password. Set the home and virtual user table information.

```
mail.hook.sendmail.add.user=adduser %1% -s /bin/false  
mail.hook.sendmail.change.password=autopasswd %1% %2%  
mail.hook.sendmail.delete.user=userdel -r %1%  
mail.hook.sendmail.home=/home  
mail.hook.sendmail.virtusertable=/etc/mail/virtusertable  
mail.hook.sendmail.virtusertable.refresh=bash -c "makemap hash  
/etc/mail/virtusertable < /etc/mail/virtusertable"
```

ShellHook

Set the location of the shell script that will interface with any mail server.

```
mail.hook.shell.script=/usr/sbin/mailadmin.ksh
```

OpenOffice

Enabling OpenOffice integration allows the Document Library portlet to provide document conversion functionality. To start OpenOffice as a service, run the command:

```
soffice -headless -accept="socket,host=127.0.0.1,port=8100;urp;"  
-nofirststartwizard
```

This is tested with OpenOffice 2.3.x.

```
openoffice.server.enabled=false  
openoffice.server.host=127.0.0.1  
openoffice.server.port=8100
```

Poller

Specify the poller request timeout in milliseconds. This prevents the poller from locking up the application server.

```
poller.request.timeout=1000
```

POP

Set this to true to enable polling of email notifications from a POP server. The user credentials are the same used for SMTP authentication and is specified in the *mail/MailSession* configuration for each application server.

```
pop.server.notifications.enabled=false
```

Set the interval on which the POPNotificationsJob will run. The value is set in one minute increments.

```
pop.server.notifications.interval=1
```

Set this property to create a special MX subdomain to receive all portal related email (e.g. events.liferay.com). This means configuring a default inbox for the domain and receiving all emails into that inbox.

This approach may not be allowed for some organizations. If you cannot use the subdomain approach, unset this value and Liferay will use the replyTo address specified in the portlet preferences.

```
pop.server.subdomain=events
```

Quartz

These properties define the connection to the built-in Quartz job scheduling engine.

```
org.quartz.dataSource.ds.connectionProvider.class=com.liferay.portal.scheduler.quartz.QuartzConnectionProviderImpl  
org.quartz.jobStore.class=org.quartz.impl.jdbcjobstore.JobStoreTX  
org.quartz.jobStore.dataSource=ds  
org.quartz.jobStore.driverDelegateClass=com.liferay.portal.scheduler.quartz.DynamicDriverDelegate  
org.quartz.jobStore.isClustered=false  
org.quartz.jobStore.misfireThreshold=60000  
org.quartz.jobStore.tablePrefix=QUARTZ_  
org.quartz.jobStore.useProperties=true  
org.quartz.scheduler.instanceId=AUTO  
org.quartz.scheduler.instanceName=QuartzSchedulerEngineInstance  
org.quartz.threadPool.class=org.quartz.simpl.SimpleThreadPool  
org.quartz.threadPool.threadCount=5  
org.quartz.threadPool.threadPriority=5
```

Scheduler

Set this to false to disable all scheduler classes defined in *liferay-portlet.xml* and in the property *scheduler.classes*.

```
scheduler.enabled=true
```

Input a list of comma delimited class names that implement *com.liferay.portal.kernel.job.Scheduler*. These classes allow jobs to be scheduled on startup. These classes are not associated to any one portlet.

```
scheduler.classes=
```

Search Container

Set the available values for the number of entries to display per page. An empty value, or commenting out the value, will disable delta resizing.

The default of 20 will apply in all cases.

Always include 20, since it is the default page size when no delta is specified. The absolute maximum allowed delta is 200.

```
search.container.page.delta.values=5,10,20,30,50,75
```

Sharepoint

Set the tokens for supported Sharepoint storage paths.

```
sharepoint.storage.tokens=document_library
```

Set the class names for supported Sharepoint storage classes.

```
sharepoint.storage.class[document_library]=com.liferay.portlet.documentlibrary.sharepoint.DLSharepointStorageImpl
```

Social Bookmarks

The Blogs portlet allows for the posting of entries to various popular social bookmarking sites. The example ones are the defaults; to configure more, just add the site in the format below.

```
social.bookmark.types=blinklist,delicious,digg,furl,newsvine,reddit,technorati
social.bookmark.post.url[blinklist]=http://blinklist.com/index.php?Action=Blink/addblink.php&url=${liferay:social-bookmark:url}&Title=${liferay:social-bookmark:title}
social.bookmark.post.url[delicious]=http://del.icio.us/post?url=${liferay:social-bookmark:url}&title=${liferay:social-bookmark:title}
social.bookmark.post.url[digg]=http://digg.com/submit?phase=2&url=${liferay:social-bookmark:url}
social.bookmark.post.url[furl]=http://furl.net/storeIt.jsp?u=${liferay:social-bookmark:url}&t=${liferay:social-bookmark:title}
social.bookmark.post.url[newsvine]=http://www.newsvine.com/_tools/seed&save?u=${liferay:social-bookmark:url}&h=${liferay:social-bookmark:title}
social.bookmark.post.url[reddit]=http://reddit.com/submit?url=${liferay:social-bookmark:url}&title=${liferay:social-bookmark:title}
social.bookmark.post.url[technorati]=http://technorati.com/cosmos/search.html?url=${liferay:social-bookmark:url}
```

Velocity Engine

Input a list of comma delimited class names that extend *com.liferay.util.velocity.VelocityResourceListener*. These classes will run in sequence to allow you to find the applicable ResourceLoader to load a Velocity template.

```
velocity.engine.resource.listeners=com.liferay.portal.velocity.ServletVelocityResourceListener,com.liferay.portal.velocity.JournalTemplateVelocityResourceListener,com.liferay.portal.velocity.ThemeLoaderVelocityResourceListener,com.liferay.portal.velocity.ClassLoaderVelocityResourceListener
```

Set the Velocity resource managers. We extend the Velocity's default resource managers for better scalability.

Note that the modification check interval is not respected because the resource loader implementation does not know the last modified date of a resource. This means you will need to turn off caching if you want to be able to modify VM templates in themes and see the changes right away.

```
velocity.engine.resource.manager=com.liferay.portal.velocity.LiferayResourceManager
velocity.engine.resource.manager.cache=com.liferay.portal.velocity.LiferayResourceCache
velocity.engine.resource.manager.cache.enabled=true
```

```
#velocity.engine.resource.manager.modification.check.interval=0
```

Input a list of comma delimited macros that will be loaded. These files must exist in the class path.

```
velocity.engine.velocimacro.library=VM_global_library.vm,VM_liferay.vm
```

Set the Velocity logging configuration.

```
velocity.engine.logger=org.apache.velocity.runtime.log.SimpleLog4JLogSystem  
velocity.engine.logger.category=org.apache.velocity
```

Virtual Hosts

Set the extensions that will be ignored for virtual hosts.

```
virtual.hosts.ignore.extensions=\  
/c,\  
.css,\  
.gif,\  
.image/company_logo,\  
.ico,\  
.js,\  
.jpeg,\  
.jsp,\  
.png,\  
/portal/layout,\  
/portal/login,\  
/portal/logout
```

Set the hosts that will be ignored for virtual hosts.

```
virtual.hosts.ignore.hosts=\  
127.0.0.1,\  
localhost
```

Set the paths that will be ignored for virtual hosts.

```
virtual.hosts.ignore.paths=\  
/c,\  
\  
/c/portal/change_password,\  
  
/c/portal/extend_session,\  
/c/portal/extend_session_confirm,\  
  
/c/portal/json_service,\  
/c/portal/layout,\  
/c/portal/login,\  
/c/portal/logout,\  
/c/portal/portlet_url,\  
/c/portal/render_portlet,\
```

```
/c/portal/reverse_ajax,\  
/c/portal/session_tree_js_click,\  
/c/portal/status,\  
/c/portal/update_layout,\  
/c/portal/update_terms_of_use,\  
/c/portal/upload_progress_poller,\  
\  
/c/layout_configuration/templates,\  
/c/layout_management/update_page
```

Specify the community name that will default to the company's virtual host. If the specified community has a virtual host, then that will take precedence. If it does not, then it will use the company's virtual host.

This property is useful to remove /web/guest (or any other community) from the default URL. For example, if this property is not set, then the default URL may be `http://localhost:8080/web/guest/home`. If this property is set, then the default URL may be `http://localhost:8080/home`.

```
virtual.hosts.default.community.name=Guest
```

HTTP

See `system.properties` for more HTTP settings.

Set the maximum number of connections.

```
#com.liferay.portal.util.HttpImpl.max.connections.per.host=2  
#com.liferay.portal.util.HttpImpl.max.total.connections=20
```

Set the proxy authentication type.

```
#com.liferay.portal.util.HttpImpl.proxy.auth.type=username-password  
#com.liferay.portal.util.HttpImpl.proxy.auth.type=ntlm
```

Set user name and password used for HTTP proxy authentication.

```
#com.liferay.portal.util.HttpImpl.proxy.username=  
#com.liferay.portal.util.HttpImpl.proxy.password=
```

Set additional properties for NTLM authentication.

```
#com.liferay.portal.util.HttpImpl.proxy.ntlm.domain=  
#com.liferay.portal.util.HttpImpl.proxy.ntlm.host=
```

Set the connection timeout when fetching HTTP content.

```
com.liferay.portal.util.HttpImpl.timeout=10000
```

Servlet Filters

The audit filter populates the `AuditRequestThreadLocal` with the appropriate request values to generate audit requests.

```
com.liferay.portal.servlet.filters.audit.AuditFilter=false
```

The auto login filter processes the classes in the property `auto.login.hooks` to provide auto login functionality.

```
com.liferay.portal.servlet.filters.autologin.AutoLoginFilter=true
```

The cache filter will cache content. See `ehcache.xml` to modify the cache expiration time to live.

```
com.liferay.portal.servlet.filters.cache.CacheFilter=true
```

The CAS filter is used to provide CAS based single sign on.

```
com.liferay.portal.servlet.filters.sso.cas.CASFilter=true
```

This double click filter will prevent double clicks at the server side. Prevention of double clicks is already in place on the client side. However, some sites require a more robust solution. This is turned off by default since most sites will not need it.

```
com.liferay.portal.servlet.filters.doubleclick.DoubleClickFilter=false
```

The ETag filter is used to generate ETag headers.

```
com.liferay.portal.servlet.filters.etag.ETagFilter=true
```

If the user can unzip compressed HTTP content, the GZip filter will zip up the HTTP content before sending it to the user. This will speed up page rendering for users that are on dial up.

```
com.liferay.portal.servlet.filters.gzip.GZipFilter=true
```

The header filter is used to set request headers.

```
com.liferay.portal.servlet.filters.header.HeaderFilter=true
```

The I18n filter is used to internationalize URLs. See the property `locale.prepend.friendly.url.style` for more information.

```
com.liferay.portal.servlet.filters.i18n.I18nFilter=true
```

The Language filter replaces JavaScript code that make a client side call to translate a piece of text with the actual translated value. For example, a typical piece of JavaScript code fits the pattern `Liferay.Language.get('key')` where `key` is the text to translate. This filter will replace the entire piece of code with the translated text. This is very useful because it will lower the number of client calls by translating the text before the browser receives the JavaScript file.

```
com.liferay.portal.servlet.filters.language.LanguageFilter=true
```

The minifier filter is used to minify CSS and JavaScript.

```
com.liferay.portal.servlet.filters.minifier.MinifierFilter=true
```

The monitoring filter monitors portal request performance.

```
com.liferay.portal.servlet.filters.monitoring.MonitoringFilter=true
```

The NTLM filter is used to provide NTLM based single sign on.

```
com.liferay.portal.servlet.filters.sso.ntlm.NtlmFilter=true
```

The NTLM post filter is used to fix known issues with NTLM and ajax requests. See LPS-3795.

```
com.liferay.portal.servlet.filters.sso.ntlm.NtlmPostFilter=true
```

The OpenSSO filter is used to provide OpenSSO based single sign on.

```
com.liferay.portal.servlet.filters.sso.opensso.OpenSSOFilter=true
```

The secure filter is used to protect servlets based on IP and protocol. See the properties *.servlet.hosts.allowed and *.servlet.https.required.

```
com.liferay.portal.servlet.filters.secure.SecureFilter=true
```

The servlet authorizing filter allows external servlets to be authorized by the portal. See LEP-4682.

```
com.liferay.portal.servlet.filters.servletauthorizing.ServletAuthorizingFilter=true
```

The strip filter will remove blank lines from the content. This will speed up page rendering for users that are on dial up.

```
com.liferay.portal.servlet.filters.strip.StripFilter=true
```

The layout cache filter will cache pages to speed up page rendering for guest users. See *ehcache.xml* to modify the cache expiration time to live.

```
com.liferay.portal.servlet.filters.layoutcache.LayoutCacheFilter=true
```

The session id filter ensure that only one session is created between http and https sessions. This is useful if you want users to login via https but have them view the rest of the site via http. This is disabled by default. Do not enable this unless you thoroughly understand how cookies, http, and https work.

```
com.liferay.portal.servlet.filters.sessionid.SessionIdFilter=false
```

The Sharepoint filter allows users to access documents in the Document Library directly from Microsoft Office using the Sharepoint protocol.

```
com.liferay.portal.sharepoint.SharepointFilter=true
```

The strip filter will remove blank lines from the outputted content. This will speed up page rendering for users that are on dial up.

```
com.liferay.portal.servlet.filters.strip.StripFilter=true
```

The theme preview filter generates a preview of the currently applied theme that can be used by the Dreamweaver Theming plugin. This is disabled by default. Set the themePreview parameter to 1 in the URL to access the

theme preview for any page. For example, a URL can be <http://localhost:8080/web/guest?themePreview=1>.

```
com.liferay.portal.servlet.filters.themepreview.ThemePreviewFilter=false
```

The thread local filter cleans up registered thread locals to prevent memory leaks. Register your thread local with com.liferay.portal.kernel.util.ThreadLocalRegistry.

```
com.liferay.portal.servlet.filters.threadlocal.ThreadLocalFilter=true
```

The valid HTML filter will move JavaScript that is outside of the closing body tag to its proper place inside the body tag. Most sites will prefer to leave this filter disabled because having JavaScript outside of the body tag causes the page to render faster. However, the side effect is that it will also make the site inaccessible to screen readers because the HTML is technically invalid. Setting this property to true optimizes for accessibility while setting this property to false optimizes for browser performance.

```
com.liferay.portal.servlet.filters.validhtml.ValidHtmlFilter=false
```

The Velocity filter will process */css/main.css as a Velocity template.

```
com.liferay.portal.servlet.filters.velocity.VelocityFilter=false
```

The virtual host filter maps hosts to public and private pages. For example, if the public virtual host is www.helloworld.com and the friendly URL is /helloworld, then http://www.helloworld.com is mapped to http://localhost:8080/web/helloworld.

```
com.liferay.portal.servlet.filters.virtualhost.VirtualHostFilter=true
```

Upload Servlet Request

Set the maximum file size. Default is 1024 * 1024 * 100.

```
com.liferay.portal.upload.UploadServletRequestImpl.max.size=104857600
```

Set the temp directory for uploaded files.

```
#com.liferay.portal.upload.UploadServletRequestImpl.temp.dir=C:/Temp
```

Set the threshold size to prevent extraneous serialization of uploaded data.

```
com.liferay.portal.upload.LiferayFileItem.threshold.size=262144
```

Set the threshold size to prevent out of memory exceptions caused by caching excessively large uploaded data. Default is 1024 * 1024 * 10.

```
com.liferay.portal.upload.LiferayInputStream.threshold.size=10485760
```

Web Server

Set the HTTP and HTTPS ports when running the portal in a J2EE server that is sitting behind another web server like Apache. Set the values to -1 if the portal is not running behind another web server like Apache.

```
web.server.http.port=-1  
web.server.https.port=-1
```

Set the hostname that will be used when the portlet generates URLs. Leaving this blank will mean the host is derived from the servlet container.

```
web.server.host=
```

Set the preferred protocol.

```
web.server.protocol=https
```

Set this to true to display the server name at the bottom of every page. This is useful when testing clustering configurations so that you can know which node you are accessing.

```
web.server.display.node=false
```

WebDAV

Set a list of files for the WebDAV servlet to ignore processing.

```
webdav.ignore=.DS_Store,.metadata_index_homes_only,.metadata_never_index,.Spotlight-V100,.TemporaryItems,.Trashes
```

Main Servlet

Servlets can be protected by *com.liferay.portal.servlet.filters.secure.SecureFilter*.

Input a list of comma delimited IPs that can access this servlet. Input a blank list to allow any IP to access this servlet. SERVER_IP will be replaced with the IP of the host server.

```
main.servlet.hosts.allowed=
```

Set the following to true if this servlet can only be accessed via https.

```
main.servlet.https.required=false
```

Axis Servlet

See Main Servlet on how to protect this servlet.

```
axis.servlet.hosts.allowed=127.0.0.1,SERVER_IP  
axis.servlet.https.required=false
```

Google Gadget Servlet

Set the servlet mapping for the Google Gadget servlet.

```
google.gadget.servlet.mapping=/google_gadget
```

JSON Tunnel Servlet

See Main Servlet on how to protect this servlet.

```
json.servlet.hosts.allowed=
json.servlet.https.required=false
```

Liferay Tunnel Servlet

See Main Servlet on how to protect this servlet.

```
tunnel.servlet.hosts.allowed=127.0.0.1, SERVER_IP
tunnel.servlet.https.required=false
```

Netvibes Servlet

Set the servlet mapping for the Netvibes servlet.

```
netvibes.servlet.mapping=/netvibes
```

Spring Remoting Servlet

See Main Servlet on how to protect this servlet.

```
spring.remoting.servlet.hosts.allowed=127.0.0.1, SERVER_IP
spring.remoting.servlet.https.required=false
```

WebDAV Servlet

See Main Servlet on how to protect this servlet.

```
webdav.servlet.hosts.allowed=
webdav.servlet.https.required=false
```

Widget Servlet

Set the servlet mapping for the widget servlet.

```
widget.servlet.mapping=/widget
```

Admin Portlet

You can set some administrative defaults by using these properties. The first time you bring up your portal, these values will then already be set in the Admin portlet. All values should be separated by \n characters.

Set up default group names.

```
admin.default.group.names=
```

Set up default role names.

```
admin.default.role.names=Power User\nUser
```

Set up default user group names.

```
admin.default.user.group.names=
```

Set this to true to ensure that a user is synchronized with the default associations of groups, roles, and user groups upon every log in. Set this to false if default associations should only be applied to a user when a user is created.

```
admin.sync.default.associations=false
```

The rest of these properties map to their values in the Admin portlet.

```
admin.mail.host.names=
admin.reserved.screen.names=
admin.reserved.email.addresses=
admin.email.from.name=Joe Bloggs
admin.email.from.address=test@liferay.com
admin.email.user.added.enabled=true
admin.email.user.added.subject=com/liferay/portlet/admin/dependencies/email_
user_added_subject tmpl
admin.email.user.added.body=com/liferay/portlet/admin/dependencies/email_use
r_added_body tmpl
admin.email.password.sent.enabled=true
admin.email.password.sent.subject=com/liferay/portlet/admin/dependencies/ema
il_password_sent_subject tmpl
admin.email.password.sent.body=com/liferay/portlet/admin/dependencies/email_
password_sent_body tmpl
```

Announcements Portlet

Configure email notification settings.

```
announcements.email.from.name=Joe Bloggs
announcements.email.from.address=test@liferay.com
announcements.email.to.name=
announcements.email.to.address=noreply@liferay.com
announcements.email.subject=com/liferay/portlet/announcements/dependencies/e
mail_subject tmpl
announcements.email.body=com/liferay/portlet/announcements/dependencies/ema
il_body tmpl
```

Set the list of announcement types. The display text of each of the announcement types is set in content/Language.properties.

```
announcements.entry.types=general,news,test
```

Set the interval on which the CheckEntryJob will run. The value is set in one minute increments.

```
announcements.entry.check.interval=15
```

Asset Publisher Portlet

Input a list of comma separated display styles that will be available in the configuration screen of Asset Publisher portlet.

```
asset.publisher.display.styles=table,title-list,abstracts,full-content
```

Asset

Input a list of comma delimited default properties for new categories. Each item of the list should have the following format: key:value.

```
asset.categories.properties.default=
```

Set the following to false to specify that searching and browsing using categories should only show assets that have been assigned the selected category explicitly. When set to true, the children categories are also included in the search.

```
asset.categories.search.hierarchical=true
```

Set this to true to enable incrementing the view counter for assets.

```
asset.entry.increment.view.counter.enabled=true
```

Input a class name that implements com.liferay.portlet.asset.util.AssetEntryValidator. This class will be called to validate entries. The DefaultAssetEntryValidator class is just an empty class that doesn't actually do any validation. The MinimalAssetEntryValidator requires all entries to have at least one tag.

```
asset.entry.validator=com.liferay.portlet.asset.util.DefaultAssetEntryValidator  
asset.entry.validator=com.liferay.portlet.asset.util.MinimalAssetEntryValidator
```

Input a list of comma delimited default tag properties for new tags. Each item of the list should have the following format: key:value.

```
asset.tag.properties.default=
```

Set the name of the default vocabulary which will be created by default.

```
asset.vocabulary.default=Topic
```

Set a property with the prefix asset.renderer.enabled. and a suffix with the asset renderer factory class name to enable or disable an asset renderer factory. The default setting is true. See LPS-6085 for more information.

```
asset.renderer.enabled.com.liferay.portlet.documentlibrary.asset.DLFFileEntry  
AssetRendererFactory=false
```

Blogs Portlet

The following properties affect the Blogs portlet.

```
blogs.email.comments.added.enabled=true  
blogs.email.comments.added.subject=com/liferay/portlet/blogs/dependencies/email_comments_added_subject tmpl  
blogs.email.comments.added.body=com/liferay/portlet/blogs/dependencies/email_comments_added_body tmpl  
blogs.page.abstract.length=400  
blogs.rss.abstract.length=200  
blogs.trackback.excerpt.length=50
```

Configure email notification settings.

```
blogs.email.from.name=Joe Bloggs  
blogs.email.from.address=test@liferay.com  
blogs.email.entry.added.enabled=true  
blogs.email.entry.added.subject=com/liferay/portlet/blogs/dependencies/email_entry_added_subject tmpl  
blogs.email.entry.added.body=com/liferay/portlet/blogs/dependencies/email_entry_added_body tmpl  
blogs.email.entry.updated.enabled=true  
blogs.email.entry.updated.subject=com/liferay/portlet/blogs/dependencies/email_entry_updated_subject tmpl  
blogs.email.entry.updated.body=com/liferay/portlet/blogs/dependencies/email_entry_updated_body tmpl
```

Set the interval on which the TrackbackVerifierJob will run. The value is set in one minute increments.

```
blogs.trackback.verifier.job.interval=5
```

Set the excerpt length for linkbacks.

```
blogs.linkback.excerpt.length=200
```

Set the interval on which the LinkbackMessageListener will run. The value is set in one minute increments.

```
blogs.linkback.job.interval=5
```

Set this to true to enable pingbacks.

```
blogs.pingback.enabled=true
```

Set this to true to enable trackbacks.

```
blogs.trackback.enabled=true
```

Set this to true to enable pinging Google on new and updated blog entries.

```
blogs.ping.google.enabled=true
```

Set this to true to enable comments for blogs entries.

```
blogs.entry.comments.enabled=true
```

Set this to true to enable previous and next navigation for blogs entries.

```
blogs.entry.previous.and.next.navigation.enabled=true
```

Breadcrumb Portlet

Set this to true to show the Guest community as the top level parent in the breadcrumbs. It will only show if it has at least one page.

```
breadcrumb.show.guest.group=true
```

Set this to true to show the path of parent communities or organizations in the breadcrumbs. A community or organization will only be shown if it has at least one page.

```
breadcrumb.show.parent.groups=true
```

Calendar Portlet

Set the list of event types. The display text of each of the event types is set in *content/Language.properties*.

```
calendar.event.types=anniversary,appointment,bill-payment,birthday,breakfast,call,chat,class,club-event,concert,dinner,event,graduation,happy-hour,holiday,interview,lunch,meeting,movie,net-event,other,party,performance,press-release,reunion,sports-event,training,travel,tv-show,vacation,wedding
```

Set the interval on which the CheckEventJob will run. The value is set in one minute increments.

```
calendar.event.check.interval=15
```

Configure email notification settings.

```
calendar.email.from.name=Joe Bloggs  
calendar.email.from.address=test@liferay.com  
calendar.email.event.reminder.enabled=true  
calendar.email.event.reminder.subject=com/liferay/portlet/calendar/dependencies/email_event_reminder_subject.tml  
calendar.email.event.reminder.body=com/liferay/portlet/calendar/dependencies/email_event_reminder_body.tml
```

Set this to true to enable comments for calendar events.

```
calendar.event.comments.enabled=true
```

Communities Portlet

Set this to true to allow community members to see the Communities portlet and the communities he is a member of in the Control Panel. Setting this to false will only allow administrators to see this portlet in the Control Panel.

```
communities.control.panel.members.visible=true
```

Configure email notification settings.

```
communities.email.from.name=Joe Bloggs
```

```
communities.email.from.address=test@liferay.com
communities.email.membership.reply.subject=com/liferay/portlet/communities/d
ependencies/email_membership_reply_subject tmpl
communities.email.membership.reply.body=com/liferay/portlet/communities/depe
ndencies/email_membership_reply_body tmpl
communities.email.membership.request.subject=com/liferay/portlet/communities/
dependencies/email_membership_request_subject tmpl
communities.email.membership.request.body=com/liferay/portlet/communities/de
pendencies/email_membership_request_body tmpl
```

Discussion Tag Library

Set the thread view for discussion comments. This will affect Blogs, Document Library, and other portlets that use the Discussion tag library to provide comments. Set the value to flat to paginate comments. Set the value to combination to show all comments in one page along with a tree view of the comments.

```
discussion.thread.view=combination
discussion.thread.view=flat
```

Document Library Portlet

Set the name of a class that implements com.liferay.documentlibrary.util.Hook. The document library server will use this persist documents.

```
dl.hook.impl=com.liferay.documentlibrary.util.AdvancedFileSystemHook
dl.hook.impl=com.liferay.documentlibrary.util.CMISHook
dl.hook.impl=com.liferay.documentlibrary.util.FileSystemHook
dl.hook.impl=com.liferay.documentlibrary.util.JCRHook
dl.hook.impl=com.liferay.documentlibrary.util.S3Hook
FileSystemHook
dl.hook.file.system.root.dir=${liferay.home}/data/document_library
JCRHook
dl.hook.jcr.fetch.delay=500
dl.hook.jcr.fetch.max.failures=5
```

A file extension of * will permit all file extensions.

```
dl.file.extensions=*
dl.file.extensions=.bmp,.css,.doc,.docx,.dot,.gif,.gz,.htm,.html,.jpg,.js,.js
ar,.odb,.odf,.odg,.odp,.ods,.odt,.pdf,.png,.ppt,.pptx,.rtf,.swf,.sxc,.sxi,.s
xw,.tar,.tiff,.tgz,.txt,.vsd,.xls,.xlsx,.xml,.zip,.jrxml
```

Set this to false to allow users to update file entries by uploading a file with an extension different from the one of the originally uploaded file. There is a known issue where setting this to true will break OSX compatibility. See LPS-10770 for more information.

```
dl.file.extensions.strict.check=false
```

Set the maximum file size for indexing file contents. Files larger than this property will not have their contents indexed, only their metadata will

be indexed. A value of -1 indicates that all file contents will be indexed. A value of 0 indicates that no file contents will be indexed.

```
dl.file.indexing.max.size=10485760
```

Set this to true to enable the read count for document library files.

```
dl.file.entry.read.count.enabled=true
```

Set this to true to enable file rank for document library files.

```
dl.file.rank.enabled=true
```

Set this to true if document library should be published to live by default.

```
dl.publish.to.live.default=true
```

Set this to true to hold the lock when an unlock is requested through WebDAV until a user manually releases the lock from the Document Library portlet. Set this to false to release the lock.

```
dl.webdav.hold.lock=false
```

Set this to true to create only one document library file version when saving multiple times during a WebDAV session.

```
dl.webdav.save.to.single.version=false
```

Dockbar Portlet

Set the portlet ids that will be shown directly in the *Add Application* menu.

```
dockbar.add.portlets=56,101,110,71
```

Flags Portlet

Input a list of questions used for flag reasons.

```
flags.reasons=sexual-content,violent-or-repulsive-content,hateful-or-abusive-content,harmful-dangerous-acts,spam,infringes-my-rights
```

Email Notification Settings

```
flags.email.from.name=Joe Bloggs  
flags.email.from.address=test@liferay.com
```

```
flags.email.subject=com/liferay/portlet/flags/dependencies/email_flag_subject tmpl  
flags.email.body=com/liferay/portlet/flags/dependencies/email_flag_body tmpl
```

Set this to true to enable guest users to flag content

```
flags.guest.users.enabled=false
```

IFrame Portlet

Specify a role name that a user must be associated with in order to configure the IFrame portlet to use the @password@ token. This token is used to

post the password of users who access this portlet in order to automatically login to the framed site.

No role is required by default. However, it is recommended that you specify a role in high security environments where users who configure this portlet may attempt password theft. See LPS-5272 for more information.

```
iframe.password.token.role=
```

Image Gallery Portlet

Set the maximum file size and valid file extensions for images. A value of 0 for the maximum file size can be used to indicate unlimited file size. However, the maximum file size allowed is set in the property `com.liferay.portal.upload.UploadServletRequestImpl.max.size`.

```
ig.image.max.size=10240000
```

A file extension of * will permit all file extensions.

```
ig.image.extensions=.bmp,.gif,.jpeg,.jpg,.png,.tif,.tiff
```

Set the maximum thumbnail height and width in pixels. Set dimension of the custom images to 0 to disable creating a scaled image of that size.

```
ig.image.thumbnail.max.dimension=150  
ig.image.custom1.max.dimension=100  
ig.image.custom2.max.dimension=0
```

Set this to true if image gallery should be published to live by default.

```
ig.publish.to.live.by.default=true
```

Login Portlet

Set this to true to allow the user to choose a password during account creation.

```
login.create.account.allow.custom.password=false
```

Invitation Portlet

```
invitation.email.max.recipients=20  
invitation.email.message.body=com/liferay/portlet/invitation/dependencies/email_message_body.tml  
invitation.email.message.subject=com/liferay/portlet/invitation/dependencies/email_message_subject.tml
```

Journal Portlet

Set this to true if article ids should always be autogenerated.

```
journal.article.force.autogenerate.id=true
```

Set this to true so that only the latest version of an article that is also not approved can be saved without incrementing version.

```
journal.article.force.increment.version=false
```

Set the list of article types. The display text of each of the article types is set in content/Language.properties.

```
journal.article.types=announcements,blogs,general,news,press-release,test
```

Set the token used when inserting simple page breaks in articles.

```
journal.article.token.page.break=@page_break@
```

Set the interval on which the CheckArticleJob will run. The value is set in one minute increments.

```
journal.article.check.interval=15
```

Set this to true to check that a user has the VIEW permission on a Journal article when its content is rendered.

```
journal.article.view.permission.check.enabled=false
```

Set this to true if feed ids should always be autogenerated.

```
journal.feed.force.autogenerate.id=false
```

Set this to true if structure ids should always be autogenerated.

```
journal.structure.force.autogenerate.id=false
```

Set this to true if template ids should always be autogenerated.

```
journal.template.force.autogenerate.id=false
```

Input a comma delimited list of variables which are restricted from the context in Velocity based Journal templates.

```
journal.template.velocity.restricted.variables=serviceLocator
```

Set the maximum file size and valid file extensions for images. A value of 0 for the maximum file size can be used to indicate unlimited file size. However, the maximum file size allowed is set in the property *com.liferay.portal.upload.UploadServletRequestImpl.max.size*.

```
journal.image.small.max.size=51200
```

A file extension of * will permit all file extensions.

```
journal.image.extensions=.gif,.jpeg,.jpg,.png
```

Input a list of comma delimited class names that extend *com.liferay.portlet.journal.util.TransformerListener*. These classes will run in sequence to allow you to modify the XML and XSL before it's transformed and allow you to modify the final output.

```
journal.transformer.listener=\
```

```
com.liferay.portlet.journal.util.TokensTransformerListener,\ 
#com.liferay.portlet.journal.util.PropertiesTransformerListener,\ 
com.liferay.portlet.journal.util.ContentTransformerListener,\ 
com.liferay.portlet.journal.util.LocaleTransformerListener,\ 
com.liferay.portlet.journal.util.RegexTransformerListener,\ 
com.liferay.portlet.journal.util.ViewCounterTransformerListener
```

Enter a list of regular expression patterns and replacements that will be applied to outputted Journal content. The list of properties must end with a subsequent integer (0, 1, etc.) and it is assumed that the list has reached an end when the pattern or replacement is not set. See *com.liferay.portlet.journal.util.RegexTransformerListener* for implementation details.

```
#journal.transformer.regex.pattern.0=beta.sample.com
#journal.transformer.regexreplacement.0=production.sample.com
#journal.transformer.regex.pattern.1=staging.sample.com
#journal.transformer.regexreplacement.1=production.sample.com
```

Set whether to synchronize content searches when server starts.

```
journal.sync.content.search.on.startup=false
```

Configure mail notification settings.

```
journal.email.from.name=Joe Bloggs
journal.email.from.address=test@liferay.com
journal.email.article.approval.denied.enabled=false
journal.email.article.approval.denied.subject=com/liferay/portlet/journal/de-
pendencies/email_article_approval_denied_subject tmpl
journal.email.article.approval.denied.body=com/liferay/portlet/journal/de-
pendencies/email_article_approval_denied_body tmpl
journal.email.article.approval.granted.enabled=false
journal.email.article.approval.granted.subject=com/liferay/portlet/journal/d-
ependencies/email_article_approval_granted_subject tmpl
journal.email.article.approval.granted.body=com/liferay/portlet/journal/de-
pendencies/email_article_approval_granted_body tmpl
journal.email.article.approval.requested.enabled=false
journal.email.article.approval.requested.subject=com/liferay/portlet/journal/
dependencies/email_article_approval_requested_subject tmpl
journal.email.article.approval.requested.body=com/liferay/portlet/journal/de-
pendencies/email_article_approval_requested_body tmpl
journal.email.article.review.enabled=false
journal.email.article.review.subject=com/liferay/portlet/journal/dependencie-
s/email_article_review_subject tmpl
journal.email.article.review.body=com/liferay/portlet/journal/dependencies/e-
mail_article_review_body tmpl
```

Specify the strategy used when Journal content is imported using the LAR system.

```
journal.lar.creation.strategy=com.liferay.portlet.journal.lar.JournalCreatio-
nStrategyImpl
```

Specify the path to the template used for providing error messages on Journal templates.

```
journal.error.template.velocity=com/liferay/portlet/journal/dependencies/error.vm  
journal.error.template.xsl=com/liferay/portlet/journal/dependencies/error.xsl
```

Journal Articles Portlet

Set the available values for the number of articles to display per page.

```
journal.articles.page.delta.values=5,10,25,50,100
```

Journal Content Search Portlet

Set whether unlisted articles are excluded from search results.

```
journal.content.search.show.listed=true
```

Message Boards Portlet

Configure mail notification settings.

```
message.boards.email.from.name=Joe Bloggs  
message.boards.email.from.address=test@liferay.com  
message.boards.email.html.format=true  
message.boards.email.message.added.enabled=true  
message.boards.email.message.added.subject.prefix=com/liferay/portlet/messageboards/dependencies/email_message_added_subject_prefix.tml  
message.boards.email.message.added.body=com/liferay/portlet/messageboards/dependencies/email_message_added_body.tml  
message.boards.email.message.added.signature=com/liferay/portlet/messageboards/dependencies/email_message_added_signature.tml  
message.boards.email.message.updated.enabled=true  
message.boards.email.message.updated.subject.prefix=com/liferay/portlet/messageboards/dependencies/email_message_updated_subject_prefix.tml  
message.boards.email.message.updated.body=com/liferay/portlet/messageboards/dependencies/email_message_updated_body.tml  
message.boards.email.message.updated.signature=com/liferay/portlet/messageboards/dependencies/email_message_updated_signature.tml
```

Set this to true to allow anonymous posting.

```
message.boards.anonymous.posting.enabled=true
```

Enter time in minutes on how often this job is run. If a user's ban is set to expire at 12:05 PM and the job runs at 2 PM, the expire will occur during the 2 PM run.

```
message.boards.expire.ban.job.interval=120
```

Enter time in days to automatically expire bans on users. Set to 0 to disable auto expire.

Examples:

```
message.boards.expire.ban.interval=10
```

```
message.boards.expire.ban.interval=0
```

Enter RSS feed abstract length. This value limits what goes in the RSS feed from the beginning of the message board post. The default is the first 200 characters.

```
message.boards.rss.abstract.length=200
```

Set this to true to enable pingbacks.

```
message.boards.pingback.enabled=true
```

Set this to true to enable previous and next navigation for message boards threads

```
message.boards.thread.previous.and.next.navigation.enabled=true
```

Set the allowed thread views and the default thread view.

```
message.boards.thread.views=combination,flat,tree
```

```
message.boards.thread.views.default=combination
```

My Places Portlet

Set the display style for the My Places navigation menu.

```
my.places.display.style=simple
```

```
my.places.display.style=classic
```

Set this to true to show user public sites with no layouts.

```
my.places.show.user.public.sites.with.no.layouts=true
```

Set this to true to show user private sites with no layouts.

```
my.places.show.user.private.sites.with.no.layouts=true
```

Set this to true to show organization public sites with no layouts.

```
my.places.show.organization.public.sites.with.no.layouts=true
```

Set this to true to show organization private sites with no layouts.

```
my.places.show.organization.private.sites.with.no.layouts=true
```

Set this to true to show community public sites with no layouts.

```
my.places.show.community.public.sites.with.no.layouts=true
```

Set this to true to show community private sites with no layouts.

```
my.places.show.community.private.sites.with.no.layouts=true
```

Set the maximum number of elements that will be shown in the My Places navigation menu. For example, if the maximum is set to 10, then, at most, 1 personal community, 10 organizations, and 10 communities will be shown.

```
my.places.max.elements=10
```

Navigation Portlet

Specify the options that will be provided to the user in the edit configuration mode of the portlet.

```
navigation.display.style.options=1,2,3,4,5,6
```

Define each mode with 4 comma delimited strings that represent the form: headerType, rootLayoutType, rootLayoutLevel, includedLayouts, and nestedChildren.

```
navigation.display.style[1]=breadcrumb,relative,0,auto,true  
navigation.display.style[2]=root-layout,absolute,2,auto,true  
navigation.display.style[3]=root-layout,absolute,1,auto,true  
navigation.display.style[4]=none,absolute,1,auto,true  
navigation.display.style[5]=none,absolute,1,all,true  
navigation.display.style[6]=none,absolute,0,auto,true
```

Nested Portlets Portlet

```
nested.portlets.layout.template.default=2_columns_i
```

Add a comma separated list of layout template ids that should not be allowed in the Nested Portlets Portlet.

```
nested.portlets.layout.template.unsupported=freeform,1_column
```

Portlet CSS Portlet

Set this to true to enable the ability to modify portlet CSS at runtime via the Look and Feel icon. Disabling it can speed up performance.

```
portlet.css.enabled=true
```

Search Portlet

Set any of these to false to disable the portlet from being searched by the Search portlet.

```
com.liferay.portlet.blogs.util.BlogsOpenSearchImpl=true  
com.liferay.portlet.bookmarks.util.BookmarksOpenSearchImpl=true  
com.liferay.portlet.calendar.util.CalendarOpenSearchImpl=true  
com.liferay.portlet.directory.util.DirectoryOpenSearchImpl=true  
com.liferay.portlet.documentlibrary.util.DLOpenSearchImpl=true  
com.liferay.portlet.imagegallery.util.IGOpenSearchImpl=true  
com.liferay.portlet.journal.util.JournalOpenSearchImpl=true  
com.liferay.portlet.messageboards.util.MBOpenSearchImpl=true  
com.liferay.portlet.wiki.util.WikiOpenSearchImpl=true
```

Shopping Portlet

Set the following to true if cart quantities must be a multiple of the item's minimum quantity.

```
shopping.cart.min.qty.multiple=true
```

Set the following to true to forward to the cart page when adding an item from the category page. The item must not have dynamic fields. All items with dynamic fields will forward to the item's details page regardless of the following setting.

```
shopping.category.forward.to.cart=false
```

Set the following to true to show special items when browsing a category.

```
shopping.category.show.special.items=false
```

Set the following to true to show availability when viewing an item.

```
shopping.item.show.availability=true
```

Set the maximum file size and valid file extensions for images. A value of 0 for the maximum file size can be used to indicate unlimited file size. However, the maximum file size allowed is set in the property `com.liferay.portal.upload.UploadServletRequestImpl.max.size`.

```
shopping.image.small.max.size=51200  
shopping.image.medium.max.size=153600  
shopping.image.large.max.size=307200
```

A file extension of * will permit all file extensions.

```
shopping.image.extensions=.gif,.jpeg,.jpg,.png
```

Configure email notification settings.

```
shopping.email.from.name=Joe Bloggs  
shopping.email.from.address=test@liferay.com  
shopping.email.order.confirmation.enabled=true  
shopping.email.order.confirmation.subject=com/liferay/portlet/shopping/dependencies/email_order_confirmation_subject tmpl  
shopping.email.order.confirmation.body=com/liferay/portlet/shopping/dependencies/email_order_confirmation_body tmpl  
shopping.email.order.shipping.enabled=true  
shopping.email.order.shipping.subject=com/liferay/portlet/shopping/dependencies/email_order_shipping_subject tmpl  
shopping.email.order.shipping.body=com/liferay/portlet/shopping/dependencies/email_order_shipping_body tmpl
```

Set this to true to enable comments for shopping orders.

```
shopping.order.comments.enabled=true
```

Software Catalog Portlet

Set the maximum file size and max file dimensions for thumbnails. A value of 0 for the maximum file size can be used to indicate unlimited file size. However, the maximum file size allowed is set in the property `com.liferay.portal.upload.UploadServletRequestImpl.max.size`.

```
sc.image.max.size=307200  
sc.image.thumbnail.max.height=200  
sc.image.thumbnail.max.width=160
```

Set this to true to enable comments for software catalog products.

```
sc.product.comments.enabled=true
```

Tags Compiler Portlet

Set this to true to enable the ability to compile tags from the URL. Disabling it can speed up performance.

```
tags.compiler.enabled=true
```

Tags Portlet

Input a class name that implements `com.liferay.portlet.tags.util.TagsAssetValidator`. This class will be called to validate assets. The `DefaultTagsAssetValidator` class is just an empty class that doesn't actually do any validation.

The `MinimalTagsAssetValidator` requires all assets to have at least one tag entry.

Examples:

```
tags.asset.validator=com.liferay.portlet.tags.util.DefaultTagsAssetValidator  
#tags.asset.validator=com.liferay.portlet.tags.util.MinimalTagsAssetValidator
```

Input a list of comma delimited default properties for new tag entries. Each item of the list should have the following format: `0:key:value`

```
tags.properties.default=
```

Set the name of the default tag set where new tags are created by default.

```
tags.vocabulary.default=Default Tag Set
```

Tasks Portlet

Specify the default number of approval stages.

```
tasks.default.stages=2
```

Specify the default role name for each stage of approval ordered from lowest level of approval to highest. These Roles must have the APPROVE_PROPOSAL permission.

```
tasks.default.role.names=Community Administrator,Community Owner
```

Translator Portlet

Set the default languages to translate a given text.

```
translator.default.languages=en_es
```

Wiki Portlet

Set the URL of a page that contains more information about the classic syntax of the wiki. It will be shown to the user when editing a page.

```
wiki.classic.syntax.help.url=http://wiki.liferay.com/index.php/Wiki_Portlet
```

Set the name of the default page for a wiki node. The name for the default page must be a valid wiki word. A wiki word follows the format of having an upper case letter followed by a series of lower case letters followed by another upper case letter and another series of lower case letters. See <http://www.usemod.com/cgi-bin/wiki.pl?WhatIsaWiki> for more information on wiki naming conventions.

```
wiki.front.page.name=FrontPage
```

Set the name of the default node that will be automatically created when the Wiki portlet is first used in a community.

```
wiki.initial.node.name>Main
```

Set the following property to specify the requirements for the names of wiki pages. By default only a few characters are forbidden. Uncomment the regular expression below to allow only CamelCase titles.

```
wiki.page.titles.regexp=([^\\\\[\\]]%&?@]+)  
#wiki.page.titles.regexp=((\\p{Lu}\\p{Ll}+)_?)+
```

Set the following property to specify the characters that will be automatically removed from the titles when importing wiki pages. This regexp should remove any characters that are forbidden in the regexp specified in `wiki.page.titles.regexp`.

```
wiki.page.titles.remove.regexp=([/\\\[\\]]%&?@]+)
```

Set this to true to enable ratings for wiki pages.

```
wiki.page.ratings.enabled=true
```

Set this to true to enable comments for wiki pages.

```
wiki.page.comments.enabled=true
```

Set the list of supported wiki formats and the default wiki format.

```
wiki.formats=creole,html  
wiki.formats.default=creole
```

Configure settings for each of the wiki formats.

```
wiki.formats.engine[creole]=com.liferay.portlet.wiki.engines.jspwiki.JSPWiki  
Engine  
wiki.formats.configuration.main[creole]=jspwiki.properties  
wiki.formats.edit.page[creole]=/html/portlet/wiki/edit/wiki.jsp  
wiki.formats.help.page[creole]=/html/portlet/wiki/help/creole.jsp  
wiki.formats.help.url[creole]=http://www.wikicreole.org/wiki/Creole1.0
```

```
wiki.formats.engine[html]=com.liferay.portlet.wiki.engines.HtmlEngine  
wiki.formats.edit.page[html]=/html/portlet/wiki/edit/html.jsp
```

```
wiki.formats.engine[plain_text]=com.liferay.portlet.wiki.engines.TextEngine  
wiki.formats.edit.page[plain_text]=/html/portlet/wiki/edit/plain_text.jsp
```

Set the list of supported wiki importers.

```
wiki.importers=MediaWiki
```

Configure settings for each of the wiki importers.

```
wiki.importers.page[MediaWiki]=/html/portlet/wiki/import/mediawiki.jsp  
wiki.importers.class[MediaWiki]=com.liferay.portlet.wiki.importers.mediawiki  
.MediaWikiImporter
```

Configure email notification settings.

```
wiki.email.from.name=Joe Bloggs  
wiki.email.from.address=test@liferay.com  
  
wiki.email.page.added.enabled=true  
wiki.email.page.added.subject.prefix=com/liferay/portlet/wiki/dependencies/e  
mail_page_added_subject_prefix.tml  
wiki.email.page.added.body=com/liferay/portlet/wiki/dependencies/email_page_  
added_body.tml  
wiki.email.page.added.signature=com/liferay/portlet/wiki/dependencies/email_  
page_added_signature.tml  
  
wiki.email.page.updated.enabled=true  
wiki.email.page.updated.subject.prefix=com/liferay/portlet/wiki/dependencies/  
email_page_updated_subject_prefix.tml  
wiki.email.page.updated.body=com/liferay/portlet/wiki/dependencies/email_pag  
e_updated_body.tml  
wiki.email.page.updated.signature=com/liferay/portlet/wiki/dependencies/ema  
il_page_updated_signature.tml  
wiki.rss.abstract.length=200
```

Plugin Management

One of the primary ways of extending the functionality of Liferay Portal is by the use of *plugins*. Plugins are an umbrella term for installable *portlet*, *theme*, *layout template*, *hook*, *Ext*, and *web module* Java EE .war files. Though Liferay comes bundled with a number of functional portlets, themes, layout templates, hooks, and web modules, plugins provide a means of extending Liferay to be able to do almost anything.

Portlets

Portlets are small web applications that run in a portion of a web page. The heart of any portal implementation is its portlets, because all of the functionality of a portal resides in its portlets. Liferay's core is a portlet container. The container's job is to manage the portal's pages and to aggregate the set of portlets that are to appear on any particular page. This means that the core doesn't contain application code. Instead, all of the features and functionality of your portal application must reside in its portlets.

Portlet applications, like servlet applications, have become a Java standard which various portal server vendors have implemented. The JSR-168 standard defines the portlet 1.0 specification, and the JSR-286 standard defines the portlet 2.0 specification. A Java standard portlet should be deployable on any portlet container which supports the standard. Portlets are placed on the page in a certain order by the end user and are served up dynamically by the portal server. This means that certain "givens" that apply to servlet-based projects, such as control over URLs or access to the `HttpServletRequest` object, don't apply in portlet projects, because the portal server generates these objects dynamically.



Tip: Liferay 4.4.2 and below support the Portlet 1.0 standard: JSR-168. Liferay 5.0 and above support the Portlet 2.0 standard: JSR-286. You cannot run Portlet 2.0 portlets in Liferay 4.4.2, but because the Portlet 2.0 standard is backwards-compatible, portlets written to the 1.0 standard will run in Liferay 5.x and above.

`HttpServletRequest` object, don't apply in portlet projects, because the portal server generates these objects dynamically.

Portal applications come generally in two flavors: 1) portlets can be written to provide small amounts of functionality and then aggregated by the portal server into a larger application, or 2) whole applications can be written to reside in only one or a few portlet windows. The choice is up to those designing the application. The developer only has to worry about what happens inside of the portlet itself; the portal server handles building out the page as it is presented to the user.

Most developers nowadays like to use certain frameworks to develop their applications, because those frameworks provide both functionality and structure to a project. For example, Struts enforces the Model-View-Controller design pattern and provides lots of functionality—such as custom tags and form validation—that make it easier for a developer to implement certain standard features. With Liferay, developers are free to use all of the leading frameworks in the Java EE space, including Struts, Spring MVC, and Java Server Faces. This allows developers familiar with those frameworks to more easily implement portlets, and also facilitates the quick porting of an application using those frameworks over to a portlet implementation.

Additionally, Liferay allows for the consuming of PHP and Ruby applications as “portlets,” so you do not need to be a Java developer in order to take advantage of Liferay’s built-in features (such as user management, communities, page building and content management). You can also use scripting languages such as Groovy if you wish. You can use the Plugins SDK to deploy your PHP or Ruby application as a portlet, and it will run seamlessly inside of Liferay. We have plenty of examples of this; to see them, check out the Plugins SDK from Liferay’s public code repository.

Does your organization make use of any Enterprise Planning (ERP) software that exposes its data via web services? You could write a portlet plugin for Liferay that can consume that data and display it as part of a dashboard page for your users. Do you subscribe to a stock service? You could pull stock quotes from that service and display them on your page, instead of using Liferay’s built-in Stocks portlet. Do you have a need to combine the functionality of two or more servlet-based applications on one page? You could make them into portlet plugins and have Liferay display them in whatever layout you want. Do you have existing Struts, Spring MVC, or JSF applications that you want to integrate with your portal? It is a straightforward task to migrate these applications into Liferay, and then they can take advantage of the layout, security, and administration infrastructure that Liferay provides.

Themes

Themes are hot deployable plugins which can completely transform the look and feel of the portal. Most organizations have their own look and feel standards which go across all of the web sites and web applications in the infrastructure. Liferay makes it possible for a site designer to create a theme



Illustration 91: Envision theme from Liferay’s theme repository

plugin which can then be installed, allowing for the complete transformation

of the portal to whatever look and feel is needed. There are lots of available theme plugins on Liferay's web site, and more are being added every day. This makes it easier for those who wish to develop themes for Liferay, as you can now choose a theme which most closely resembles what you want to do and then customize it. This is much easier than starting a theme from scratch. There is more about theme development in *Liferay in Action*.

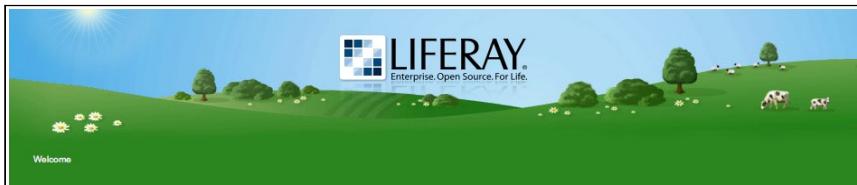


Illustration 92: Murali theme from Liferay's theme repository

Layout Templates

Layout Templates are ways of choosing how your portlets will be arranged on a page. They make up the body of your page, the large area where you drag and drop your portlets to create your pages. Liferay Portal comes with several built-in layout templates, but if you have a complex page layout (especially for your home page), you may wish to create a custom layout template of your own. This is covered in *Liferay in Action*.

Hook Plugins

Hook plugins were introduced with Liferay 5.2. As the name implies, they allow “hooking” into Liferay's core functionality. This means that they enable developers to override or replace functionality that is in the core of the system. You can hook into the eventing system, model listeners, and portal properties. You can also override Liferay's core JSPs with your own. Hooks are very powerful and have been designed to replace most of the reasons for using the extension environment with something that is easier to use and hot deployable.

Web Plugins

Web plugins are regular Java EE web modules that are designed to work with Liferay. Liferay supports integration with various Enterprise Service Bus (ESB) implementations, as well as Single Sign-On implementations, workflow engines and so on. These are implemented as web modules that are used by Liferay portlets to provide functionality.

Installing Plugins from Repositories

Liferay Portal has a section of the Control Panel which can handle plugin installation: **Plugins Installation**, which is in the **Server** category. This not only allows you to see what plugins are installed in your portal, but also it enables you to run the search indexer on those portlets that support it and install new portlets.

Go to the Dock and select *Control Panel*. In the Server category, select *Plugins Installation*.

You should now see the screen which allows you to configure and install portlets.

The default look of the Plugin Installer shows which plugins are already installed on the system, whether it is active, and what Portal roles have access to it.

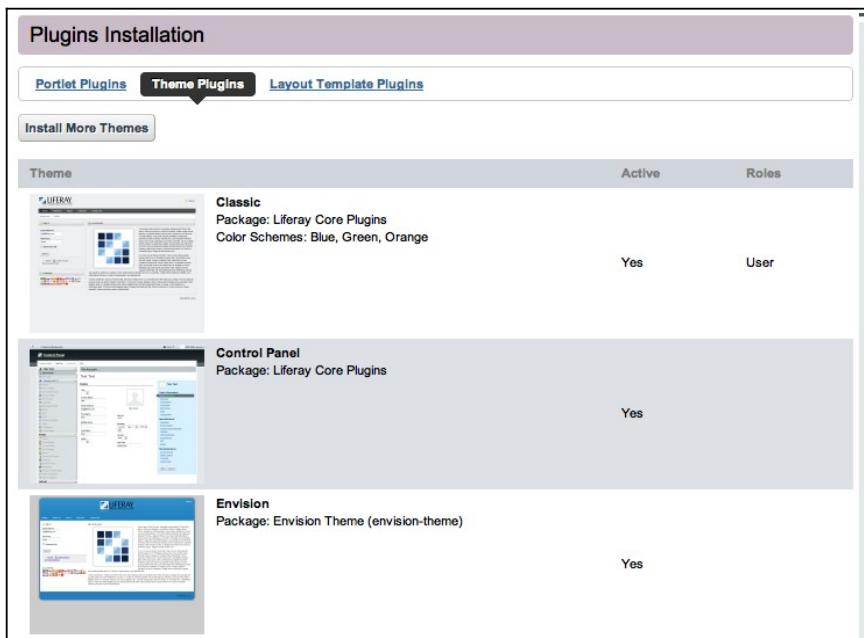


Illustration 93: Default plugin installer view

If you would like to see what plugins are available, you can do so by clicking the *Install More [Plugin Type]* button, where [Plugin Type] changes based on which tab you are viewing. Please note that the machine upon which Liferay is running must have access to the Internet in order to be able to read the Official and Community repositories. If the machine does not have access to the Internet, you will need to download the plugins from the site and install them manually. We will discuss how to do this later in this chapter.

From the initial page you can navigate to the different pages of plugins, as they are all in alphabetical order. You can also change the number of items per page and navigate to a specific page if you know where a particular plugin appears in the list. This is a standard feature of Liferay, and you will see it in most of Liferay's portlets.

The screenshot shows the Liferay Plugin Installer interface. At the top, there are tabs for 'Browse Repository', 'Upload File', 'Download File', and 'Configuration'. Below these are five more tabs: 'Portlet Plugins' (which is selected), 'Theme Plugins', 'Layout Template Plugins', 'Hook Plugins', and 'Web Plugins'. A search bar with placeholder text 'Keywords' is followed by dropdown menus for 'Tag' (set to 'All') and 'Repository' (set to 'All'). Underneath is a section for 'Install Status' with a dropdown menu set to 'Not Installed or Out of Date'. A 'Search' button is located below the search bar. The main area displays a table of portlet plugins:

Portlet Plugin	Trusted	Tags	Installed Version	Available Version	Modified Date
Web Form 6.0.4.1 ID: liferay/web-form/6.0.4.1/war This is the Web Form Portlet.	Yes	web form	-	6.0.4.1	7/23/10 11:26 PM
Tweet My World 6.0.3.1 ID: liferay/tweetmyworld/6.0.3.1/war This portlet helps in setting Twitter Status whenever a new blog post or wiki page or Product is added.	Yes	twitter, blog, wiki, software product, bitty	-	6.0.3.1	7/23/10 8:38 AM
Google Chartlet 6.0.3.1 ID: liferay/liferay/6.0.3.1/war GoogleChartlet is a google chart portlet for Liferay. This portlet has some utility charts and some some static sample charts using chart4j api.	Yes	google charts, charts4j, plugin, chartlet	-	6.0.3.1	7/23/10 5:54 AM

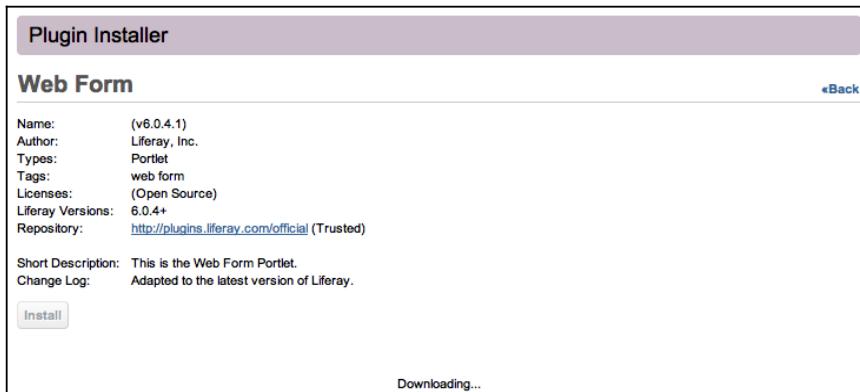
Showing 3 results.

Illustration 94: Installing plugins

After the *Install More [Plugin Type]* button is clicked, a new view appears. This view has multiple tabs, and by default, displays the *Portlet Plugins* tab. Note that the list displayed is a list of all of the plugins that are available across all of the repositories to which the server is subscribed. Above this is a search mechanism which allows you to search for plugins by their name, by whether they are installed, by tag, or by which repository they are in.

To install a plugin, choose the plugin by clicking on its name. For example, if you want to use online web forms on your web site, you might want to install the Web Form portlet. This portlet provides a handy interface which allows you to create forms for users to fill out. The results get emailed to an address which you specify.

Find the Web Form Portlet in the list by searching for it or browsing to it. Once you have found it, click on its name. Another page will be displayed which describes the portlet plugin in more detail. Below the description is an *Install* button. Click this button to install your plugin.



The plugin chosen will be automatically downloaded and installed on your instance of Liferay. If you have the Liferay console open, you can view the deployment as it happens. When it is finished, you should be able to go back to the Add Application window and add your new plugin to a page in your portal.

The same procedure is used for installing new Liferay Themes, Layout Templates, hooks, and web modules. Instead of the *Portlet Plugins* tab, you would use the appropriate tab for the type of plugin you wish to install to view the list of plugins of that type. For themes, convenient thumbnails (plus a larger version when you click on the details of a particular theme) are shown in the list.

After clicking on the *Install* button for a theme, the theme becomes available on the *Look and Feel* tab of any page.

Installing Plugins Manually

Installing plugins manually is almost as easy as installing plugins via the Plugin Installer. There are several scenarios in which you would need to install plugins manually rather than from Liferay's repositories:

- Your server is firewalled without access to the Internet. This makes it impossible for your instance of Liferay to connect to the plugin repositories.
- You are installing portlets which you have either purchased from a vendor, downloaded separately, or developed yourself.
- For security reasons, you do not want to allow portal administrators to install plugins from the Internet before they are evaluated.

You can still use the Control Panel to install plugins that are not available from the on line repositories. This is by far the easiest way to install plugins.

If your server is firewalled, you will not see any plugins displayed in the *Portlet Plugins* tab or in the *Theme Plugins* tab. Instead, you will need to click the *Upload File* tab. This gives you a simple interface for uploading a .war file containing a plugin to your Liferay Portal.

Click the *Browse* button and navigate your file system to find the portlet or theme .war you have downloaded. The other field on the page is optional: you can specify your own context for deployment. If you leave this field blank, the default context defined in the plugin (or the .war file name itself) will be used.

That's all the information the Plugin Installer needs in order to deploy your portlet, theme, layout template, hook, or web module. Click the *Install* button, and your plugin will be uploaded to the server and deployed. If it is a portlet, you should see it in the *Add Content* window. If it is a theme, it will be available on the *Look and Feel* tab in the page definition.

If you do not wish to use the Update Manager or Plugin Installer to deploy plugins, you can also deploy them at the operating system level. The first time Liferay starts, it creates a *hot deploy* folder which is by default created inside the Liferay Home folder. This folder generally resides one directory up from where your application server is installed, though it sometimes is elsewhere depending on which application server you are running. To find out where the Liferay Home folder is for your application server, please see the section on your server in Chapter 1. The first time Liferay is launched, it will create a folder structure in Liferay Home to house various configuration and administrative data. One of the folders it creates is called *deploy*. If you copy a portlet or theme plugin into this folder, Liferay will deploy it and make it available for use just as though you'd installed it via the Plugin Installer in the Control Panel. In fact, this is what the Plugin Installer is doing behind the scenes.

You can change the defaults for this directory structure so that it is stored anywhere you like by modifying the appropriate properties in your *portal-ext.properties* file. Please see the above section on the *portal-ext.properties* file for more information.

To have Liferay hot deploy a portlet or theme plugin, copy the plugin into your hot deploy folder, which by default is in [*Liferay Home*]/*deploy*. If you are watching the Liferay console, you should see messages like the following:

```

16:11:47,616 INFO [PortletAutoDeployListener:71] Copying portlets for
/Users/stephenkostas/java/liferay/bundles/liferay-portal-
6.0.4/deploy/weather-portlet-6.0.4.1.war
    Expanding: /Users/stephenkostas/java/liferay/bundles/liferay-portal-
6.0.4/deploy/weather-portlet-6.0.4.1.war into
/Users/stephenkostas/java/liferay/bundles/liferay-portal-6.0.4/tomcat-
6.0.26/temp/20100729161147694
        Copying 1 file to /Users/stephenkostas/java/liferay/bundles/liferay-
portal-6.0.4/tomcat-6.0.26/temp/20100729161147694/WEB-INF
        Copying 1 file to /Users/stephenkostas/java/liferay/bundles/liferay-
portal-6.0.4/tomcat-6.0.26/temp/20100729161147694/WEB-INF/classes
        Copying 1 file to /Users/stephenkostas/java/liferay/bundles/liferay-
portal-6.0.4/tomcat-6.0.26/temp/20100729161147694/WEB-INF/classes
        Copying 1 file to /Users/stephenkostas/java/liferay/bundles/liferay-
portal-6.0.4/tomcat-6.0.26/temp/20100729161147694/META-INF
        Copying 37 files to /Users/stephenkostas/java/liferay/bundles/liferay-
portal-6.0.4/tomcat-6.0.26/webapps/weather-portlet
        Copying 1 file to /Users/stephenkostas/java/liferay/bundles/liferay-
portal-6.0.4/tomcat-6.0.26/webapps/weather-portlet
    Deleting directory /Users/stephenkostas/java/liferay/bundles/liferay-
portal-6.0.4/tomcat-6.0.26/temp/20100729161147694
16:11:48,072 INFO [PortletAutoDeployListener:81] Portlets for
/Users/stephenkostas/java/liferay/bundles/liferay-portal-
6.0.4/deploy/weather-portlet-6.0.4.1.war copied successfully. Deployment
will start in a few seconds.
Jul 29, 2010 4:11:50 PM org.apache.catalina.startup.HostConfig
deployDirectory
INFO: Deploying web application directory weather-portlet
16:11:50,585 INFO [PortletHotDeployListener:222] Registering portlets for
weather-portlet
16:11:50,784 INFO [PortletHotDeployListener:371] 1 portlet for weather-
portlet is available for use

```

As long as you see the *available for use* message, your plugin was installed correctly, and will be available for use in the portal.

Plugin Troubleshooting

Sometimes for various reasons plugins fail to install. There are different reasons for this based on several factors, including

- Liferay configuration
- The container upon which Liferay is running
- Changing the configuration options in multiple places
- How Liferay is being launched

You will often be able to tell if you have a plugin deployment problem by looking at the Liferay server console. If you see the plugin get recognized by the hot deploy listener, you will see a *plugin copied successfully* message. If this message is not followed up by an *available for use* message, you have an issue with your plugin deployment configuration, and it is likely one of the factors above.

We will look at each of these factors.

Liferay Configuration Issues



Tip: This applies to Liferay versions prior to version 4.3.5. Liferay versions above 4.3.5 are able to auto detect the type of server it is running on, which makes things a lot easier. If you are running a newer version of Liferay, you can skip this section. If you are upgrading from one of these versions, continue reading.

Liferay by default comes as a bundle or as a .war file. Though every effort has been made to make the .war file as generic as possible, sometimes the default settings are inappropriate for the container upon which Liferay is running. Most of these problems were resolved in Liferay 4.3.5 with the addition of code that allows Liferay to determine which application server it is running on and adjust the way it deploys plugins as a result.

If you have upgraded from one of these older versions, you may still have settings in your portal.ext.properties file that are no longer needed. One of these settings is the manual override of the default value of auto.deploy.dest.dir.

In versions of Liferay prior to 4.3.5, there is a property called auto.deploy.dest.dir that defines the folder where plugins are deployed after the hot deploy utilities have finished preparing them. This folder maps to a folder that the container defines as an auto-deploy or a hot deploy folder. By default in older versions of Liferay, this property is set to ../webapps. This default value works for Tomcat containers (if Tomcat has been launched from its bin folder), but will not work for other containers that define their hot deploy folders in a different place. In newer versions of Liferay, this value is automatically set to the default for the application server upon which Liferay is running.

For example, Glassfish defines the hot deploy folder as a folder called autodeploy inside of the domain folder in which your server is running. By default, this is in <Glassfish Home>/domains/domain1/autodeploy. JBoss defines the hot deploy folder as a root folder inside of the particular server configuration you are using. By default, this is in <JBoss Home>/server/default/deploy. WebLogic defines this folder inside of the domain directory. By default, this is in <Bea Home>/user_projects/domains/<domain name>/autodeploy.

The best thing to do when upgrading to newer versions of Liferay Portal is to remove this property altogether. It is not needed, as the autodetection of the container handles the hot deploy location. If, for whatever reason, you

need to customize the location of the hot deploy folder, follow the instructions below.

You will first need to determine where the hot deploy folder is for the container you are running. Consult your product documentation for this. Once you have this value, there are two places in which you can set it: the `portal-ext.properties` file and in the Plugin Installer portlet.

To change this setting in the `portal-ext.properties` file, browse to where Liferay was deployed in your application server. Inside of this folder should be a `WEB-INF/classes` folder. Here you will find the `portal-ext.properties` file. Open this file in a text editor and look for the property `auto.deploy.dest.dir`. If it does not appear in the file, you can add it. The safest way to set this property—as we will see later—is to define the property using an absolute path from the root of your file system to your application server's hot deploy folder. For example, if you are using Glassfish, and you have the server installed in `/java/glassfish`, your `auto.deploy.dest.dir` property would look like the following:

```
auto.deploy.dest.dir=/java/glassfish/domains/domain1/autodeploy
```

Remember, if you are on a Windows system, use forward slashes instead of back slashes, like so:

```
auto.deploy.dest.dir=C:/java/glassfish/domains/domain1/autodeploy
```

Save the file and then restart your container. Now plugins should install correctly.

If you would rather change this setting via the Plugin Installer in the Control Panel (because you do not wish to restart your container), you can do that by clicking on the *Configuration* tab. On this page are a number of settings you can change, including the default folders for hot deploy, where Liferay should look for plugin repositories, and so on.



Illustration 96: Changing the hot deploy destination directory

The setting to change is the field marked *Destination Directory*. Change this to the full path to your container's auto deploy folder from the root of your file system. When you are finished, click the *Save* button at the bottom of the form. The setting will now take effect without your having to restart your container.

Note that the setting in the Control Panel overrides the setting in the properties file.

If you are having hot deploy trouble in Liferay versions 4.3.5 and greater, it is possible that the administrator of your application server has changed the default folder for auto deploy in your application server. In this case, you

would want to set `auto.deploy.dest.dir` to the customized folder location as you would with older versions of Liferay. In Liferay 4.3.5 and greater, this setting still exists, but is blank. Add the property to your `portal-ext.properties` file and set its value to the fully qualified path to the auto deploy folder configured in your application server.

Deploy Issues for Specific Containers

There are some containers, such as WebSphere®, which do not have a hot deploy feature. Unfortunately, these containers do not work with Liferay's hot deploy system. But this does not mean that you cannot install plugins on these containers. You can deploy plugins manually using the application server's deployment tools. Liferay is able to pick up the portlet plugins once they get deployed to the container manually, especially if you add it to the same Enterprise Application project that was created for Liferay.

When Liferay hot deploys portlet and theme .war files, it sometimes makes modifications to those files right before deployment. In order to successfully deploy plugins using an application server vendor's tools, you will want to run your plugins through this process before you attempt to deploy them.

In the **Plugin Installer** section of the Control Panel, click the *Configuration* tab. The second-most field on the form is labeled **Destination Directory**. Place the path to which you would like plugin .war files copied after they are processed by Liferay's plugin installer process. You will use this as a staging directory for your plugins before you install them manually with your server's deployment tools. When you are finished, click *Save*.

Now you can deploy plugins using the Plugin Installer portlet or by dropping .war files into your auto deploy directory. Liferay will pick up the files, modify them, and then copy the result into the destination directory you have configured. You may then deploy them from here to your application server.

Example: WebSphere® Application Server

1. If you don't have one already, create a `portal-ext.properties` file in the Liferay Home folder of your Liferay installation. Add the following directive to it:

```
auto.deploy.dest.dir=${liferay.home}/websphere-deploy
```

2. Create a folder called `websphere-deploy` inside your `$LIFERAY_HOME` folder. This is the folder where the Lucene index, Jackrabbit config, and deploy folders are.

3. Make sure the `web.xml` file inside the plugin you want to install has the following context parameter in it:

```
<context-param>
    <param-
name>com.ibm.websphere.portletcontainer.PortletDeploymentEnabled</param-
name>
    <param-value>false</param-value>
</context-param>
```

Liferay versions 5.2.2 and higher will automatically inject this into the `web.xml` file on WebSphere containers.

4. The WebSphere deploy occurs in two steps. You will first use Liferay's tools to "pre-deploy" the file, and then use WebSphere's tools to do the actual deployment. This is because Liferay makes deployment-time modifications to the plugins right before they are actually deployed to the application server. For other application servers, this can usually be done in one step, because Liferay can make the modifications and then copy the resulting .war file into an autodeploy folder to have it actually deployed. Because WebSphere does not have an autodeploy feature, we need to separate these two steps.
5. Deploy your .war file using Liferay's Plugin Installer or by copying it into `$LIFERAY_HOME/deploy`. Liferay will make its modifications and because we changed the `auto.deploy.dest.dir` in the first step, it will copy the resulting .war file into `$LIFERAY_HOME/websphere-deploy`. You will see a *copied successfully* message in the log.
6. Use WebSphere's tools to deploy the .war file. Make the context root for the .war file equal to the file name (i.e., `/my-first-portlet`). Once the .war file is deployed, save it to the master configuration.
7. Go back to the *Applications -> Enterprise Applications* screen in the WebSphere Admin Console. You will see that your portlet is deployed, but not yet started. Start it.
8. Liferay will immediately recognize that the portlet has been deployed and register it. The portlet will be automatically started and registered upon subsequent restarts of WebSphere.

Experienced WebSphere system administrators can further automate this by writing a script which watches the `websphere-deploy` directory and uses `wsadmin` commands to then deploy plugins automatically.

Changing the Configuration Options in Multiple Places

Sometimes, especially during development when several people have administrative access to the server at the same time, the auto deploy folder loc-

ation can get customized in both the `portal-ext.properties` file and in the Control Panel. If this happens, the value in the Control Panel takes precedence over the value in the properties file. If you go into the Control Panel and change the value to the correct setting, plugin deployment will start working again.

Creating Your Own Plugin Repository

As your enterprise builds its own library of portlets for internal use, you can create your own plugin repository to make it easy to install and upgrade portlets. This will allow different departments who may be running different instances of Liferay to share portlets and install them as needed. If you are a software development house, you may wish to create a plugin repository for your own products. Liferay makes it easy for you to create your own plugin repository and make it available to others.

You can create your plugin repository in two ways:

1. Use the Software Catalog in the Control Panel to create the repository by using its graphical interface and an HTTP server.
2. Create an XML file using the Liferay Plugin Repository DTD (http://www.liferay.com/dtd/liferay-plugin-repository_6_0_0.dtd) and an HTTP server.

Both methods have their benefits. The first method allows users to upload their plugins to an HTTP server to which they have access. They can then register their plugins with the repository by adding a link to it via the Control Panel's graphical user interface. Liferay will then generate the XML necessary to connect the repository to a Control Panel running on another instance of Liferay. This XML file can then be placed on an HTTP server, and the URL to it can be added to the Plugin Installer, making the portlets in this repository available to the server running Liferay.

The second method does not require an instance of Liferay to be running. You can upload plugins to an HTTP server of your choice, and then create an XML file called `liferay-plugin-repository.xml` manually. If you make this file available on an HTTP server (it can be the same one upon which the plugins are stored, or a different one altogether), you can connect the repository to a Plugin Installer in the Control Panel running on an instance of Liferay.

We will first look at creating a plugin repository using the Software Catalog in the Control Panel.

The Software Catalog

You will want to use the **Software Catalog** if you will have multiple users submitting portlets into the repository, and if you don't want to worry about creating the `liferay-plugin-repository.xml` file yourself.

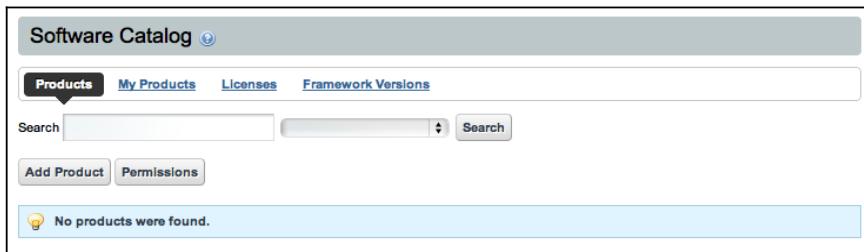


Illustration 97: The Software Catalog with nothing installed

Each community in your portal can have an instance of the Software Catalog. The Control Panel will keep track of which community or organization you are in and present you with the software catalog for that community or organization. This means that different communities/organizations can have different software repositories, so you can host several software repositories on the same instance of Liferay if you wish—they just have to be in different communities or organizations.

Choose the community that will host the plugin repository and go to the Control Panel. You will see at the top of the screen a message that says “Content for [Organization/Community],” where [Organization/Community] is the organization or community you were on when you selected the Control Panel from the Dock. If you want to administer the software catalog for a different community or organization, you can select it from the selection box.

Advanced Liferay Configuration

The screenshot shows the Liferay Software Catalog portlet. At the top, there are tabs for 'Products' and 'My Products', with 'Products' selected. Below the tabs is a search bar with fields for 'Search' and 'Search Plugins'. A dropdown arrow is located to the right of the search bar. The main area is a table with columns: 'Name', 'Version', 'Type', and 'Modified'. The table contains the following data:

Name	Version	Type	Modified
1-2-1 Columns This is a 1-2-1 columns layout template.	5.2.0.1	Layout Template	2/17/09 5:00 PM
1-3-1 Columns This is a 1-3-1 columns layout template.	5.2.0.1	Layout Template	2/17/09 5:00 PM
2-1-2 Columns This is a 2-1-2 columns layout template.	5.2.0.1	Layout Template	2/17/09 5:01 PM
3-2-3 Columns This is a 3-2-3 columns layout template.	5.2.0.1	Layout Template	2/17/09 5:01 PM
7Cogs Hook This hook contains custom configuration used for Liferay's demo website.	6.0.4.1	hook	7/23/10 7:59 PM
7Cogs Mobile Theme This is the 7Cogs Mobile Theme.	6.0.4.1	Theme	7/23/10 7:46 PM
7Cogs Theme This is the 7Cogs theme.	6.0.4.1	Theme	7/23/10 7:38 PM
CAS This is the Central Authentication Service plugin.	5.2.0.1	web	2/17/09 5:02 PM
Chat This portlet implements chat features in the portal.	6.0.4.1	Portlet	7/23/10 7:01 PM
Default Site Templates This hook contains default page and site templates.	6.0.4.1	hook	7/23/10 7:13 PM

Illustration 98: Populated Software Catalog from liferay.com

The Software Catalog has several tabs. The first tab is labeled *Products*. The default view of the portlet, when populated with software, displays what plugins are available for install or download. This can be seen in the version on Liferay's home page.

We will use an example community in order to better illustrate how to use the Software Catalog portlet. Assume you, as the portal administrator, have created a community called *Old Computers*. This community will be a web site for users to collaborate on setting up and using old computers with obsolete hardware and operating systems. Users who participate in the site will eventually get upgraded to a more privileged status and get their own blog page. To implement this, you have created a My Summary portlet which displays the user's name, picture, and description from his or her user profile. Because this portlet is generic enough that it could be useful to anyone using Liferay, you have decided to make it available in your own software catalog.

The first step in adding a plugin to your software repository is to add a *license* for your product. A license communicates to users the terms upon which you are allowing them to download and use your software. Click the *Licenses* tab and then click the *Add License* button that appears. You will then see a form which allows you to enter the title of your license, a URL pointing to the actual license document, and check boxes denoting whether the license is open source, active, or recommended.

When you have finished filling out the form, click the *Save* button. Your license will be saved. Once you have at least one license in the system, you can begin adding software products to your software catalog. Click the *Products* tab, and then click the *Add Product* button.

Your next step will be to create the product record in the software catalog. This will register the product in the software catalog and allow you to start adding versions of your software for users to download and / or install directly from their instances of Liferay. You will first need to put the .war file containing your software on a web server that is accessible without authentication to the users who will be installing your software. In the example above, the *Old Computers* site is on the Internet, so you would place the file on a web server that is accessible to anyone on the Internet. If you are creating a software catalog for an internal Intranet, you would place the file on a web server that is available to anyone inside of your organization's firewall.

To create the product record in the Software Catalog portlet, click the *Products* tab, and then click the *Add Product* button. Fill out the form with information about your product.

Software Catalog 

New Product 

Name	Test Portlet
Type	Portlet Plugin 
Licenses	Recommended Licenses LGPL Other Licenses 
Author	Test User
Page URL	http://www.liferay.com
Tags	Test Portlet  (Comma delimited list)
Short Description	Portlet for documentation demonstration.
Long Description	
Permissions	Viewable by Anyone (Guest Role)  More Options 

Save **Cancel**

Illustration 99: Adding a product to the Software Catalog

Name: The name of your software product.

Type: Select whether this is a portlet, theme, layout template, hook, or web plugin.

Licenses: Select the license(s) under which you are releasing this software.

Author: Enter the name of the author of the software.

Page URL: If the software has a home page, enter its url here.

Tags: Enter any tags you would like added to this software.

Short Description: Enter a short description. This will be displayed in the summary table of your software catalog.

Long Description: Enter a longer description. This will be displayed on the details page for this software product.

Permissions: Click the *Configure* link to set permissions for this software product.

Group ID: Enter a group ID. A group ID is a name space which usually identifies the company or organization that made the software. For our example, we will use *old-computers*.

Artifact ID: Enter an Artifact ID. The artifact ID is a unique name within the name space for your product. For our example, we will use *my-summary-portlet*.

Screenshot: Click the *Add Screenshot* button to add a screen shot of your product for users to view.

When you have finished filling out the form, click the *Save* button. You will be brought back to the product summary page, and you will see that your product has been added to the repository.

Notice that in the version column, *N/A* is being displayed. This is because there are not yet any released *versions* of your product. To make your product downloadable, you need to create a version of your product and point it to the file you uploaded to your HTTP server earlier.

Before you do that, however, you need to add a *Framework Version* to your software catalog. A Framework version denotes what version of Liferay your plugin is designed for and works on. You cannot add a version of your product without linking it to a version of the framework for which it is designed.

Why is this so important? Because as Liferay gains more and more features, you may wish to take advantage of those features in future versions of your product, while still keeping older versions of your product available for those who are using older versions of Liferay. This is perfectly illustrated in the example My Summary portlet we are using. Liferay had a My Summary portlet of its own, which does exactly what we have described here. This portlet was added to the suite of portlets which Liferay provides in the Social Networking plugin. This plugin makes use of the many social networking features which have been added to Liferay. So rather than just displaying a summary of your information, the Social Networking portlet adds features such as status updates, a “wall” for each user in his or her profile that other users can “write” on, the ability to become “friends” with other users—thereby granting them access to their profiles—and more.

None of this would work in older versions of Liferay, because the core engine that enables developers to create features like this is not there. So in this case, you would want to keep the older My Summary portlet available for users who have not yet upgraded, and make the newer social portlets available to those using latest version of Liferay. This is what *Framework Versions* does for you. If you connect to Liferay's software repositories with an old version of Liferay Portal, you will see the My Summary portlet. If you connect to

Liferay's software repositories with new version of Liferay, you will see the social portlets.

So click the *Framework Versions* tab and then click the *Add Framework Version* button.

Give the framework a name, a URL, and leave the *Active* check box checked. For our example, we have entered 6.0.3 for the name, because our portlet should work on that version and higher, and <http://www.liferay.com> for the URL. Click *Save*.

The screenshot shows the 'Software Catalog' interface with a 'Test Portlet' selected. The 'Main Fields' section contains a 'Version Name' field set to '1.0' with a note 'Initial version.' Below it is a 'Change Log' field and a 'Supported Framework Versions' field containing '6.0.4'. The 'Repository Fields' section includes 'Download Page URL', 'Direct Download URL (Recommended)', 'Test Direct Download URL' (set to 'Yes'), and 'Include Artifact in Repository' (with a note: 'You must specify a group ID and artifact ID before you can add a product version'). At the bottom are 'Save' and 'Cancel' buttons.

Illustration 100: Adding a product version to the Software Catalog

Now go back to the *Products* tab and click on your product. You will notice that a message is displayed stating that the product does not have any released versions. Click the *Add Product Version* button.

Version Name: Enter the version of your product.

Change Log: Enter some comments regarding what changed between this version and any previous versions.

Supported Framework Versions: Select the framework version for which your software product is intended. Enter a + at the end of the version number if you want to specify a version plus any future versions.

Download Page URL: If your product has a descriptive web page, enter its URL here.

Direct Download URL (Recommended): Enter a direct download link to your software product here. The Plugin Installer portlet will follow this link in order to download your software product.

Include Artifact in Repository: To enable others to use the Plugin Installer portlet to connect to your repository and download your plugin, select yes here.

When you are finished filling out the form, click the *Save* button. Your product version will be saved, and your product will now be available in the software repository.

Generating The Software Catalog

The Software Catalog works by generating an XML document which the Plugin Installer reads. Using the data from this XML document, the Plugin Installer knows where it can download the plugins from, what version of Liferay the plugins are designed for, and all other data about the plugins that have been entered into the Software Catalog portlet.

In order to get your Software Catalog to generate this XML data, you will need to access a particular URL. If you have created a friendly URL for your community (for example, the default community, which is called *guest*, has a friendly URL of /guest already configured for it), you can use the friendly URL. If not, you will first need to know the Group ID of the community in which your Software Catalog portlet resides. You can do this by accessing the Manage Pages interface and looking at the URLs for any of the pages. The URL will look something like this: <http://localhost:8080/web/10148/1>.

Obviously, it is much easier if you are using Friendly URLs, and we highly recommend that you do.

Next, go to your browser and go to the following URL:

```
http://<server name>:<port number>/software_catalog?<Friendly URL name or Group ID>
```

For example, if you are on the same machine as your Liferay instance, and that instance is running on port 8080, and your group ID from the database is 10148, you would use the following URL:

```
http://localhost:8080/software_catalog?10148
```

If you have also created a friendly URL called *old-computers* for this organization or community, you would use the following URL:

```
http://localhost:8080/software_catalog?old-computers
```

If you have configured everything properly, an XML document should be returned:

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin-repository>
  <settings/>
  <plugin-package>
```

```
<name>My Summary</name>
<module-id>old-computers/my-summary-portlet/1.0/war</module-id>
<modified-date>Thu, 23 Apr 2009 20:40:16 +0000</modified-date>
<types>
  <type>portlet</type>
</types>
<tags>
  <tag>social</tag>
  <tag>profile</tag>
</tags>
<short-description>My Summary</short-description>
<long-description>My Summary</long-description>
<change-log>Initial Version</change-log>
<download-url>http://www.liferay.com/portlets/my-summary-portlet-6.0.4.war</download-url>
<author>Rich Sezov</author>
<screenshots/>
<licenses>
  <license osi-approved="true">MIT License</license>
</licenses>
<liferay-versions/>
</plugin-package>
</plugin-repository>
```

You can now give the URL to your software repository out on your web-site, and other administrators of Liferay can enter it into the Plugins Installation module of their Liferay Control Panels to connect to your repository.

If you want to serve your repository off of a static web server, you can save this document to a file called `liferay-plugin-package.xml` and put this file on your HTTP server. You can then give out the URL to the directory which holds this file on your web site, and anyone with an instance of Liferay will be able to point their Plugin Installer portlets to it.

Benefits of the Software Catalog

As you can see, the Software Catalog makes it easy for you to create a repository of your software. Users of Liferay can configure their Plugin Installers to attach to your repository, and the proper versions of your software will be automatically made available to them by a single click. This is by far the easiest way for you to keep track of your software, and for your users to obtain your software.

Another benefit of the Software Catalog is that by using it, you make available to your users a standard interface for manually downloading your software. For those who prefer to manually download plugins, your Software Catalog gives them an interface to go in, find your software either by browsing or by searching, preview screen shots, and download your software—and

you don't have to build any of those pages yourself. Simply configure your software in the portlet, and all of that is done for you.

How can you do this? The Software Catalog is also available as a portlet. You can add it to any page on your web site through the *Add Application* menu. You will find the portlet in the *Tools* category.

Manually Creating A Software Catalog

If you do not wish to use the Control Panel to create your software catalog, you can create it manually by manually typing out the XML file that the Software Catalog section of the Control Panel would normally generate. Note that if you do this, you will not be able to use the Software Catalog portlet as a graphical user interface to your software that end users can use to download your software manually: you will have to build this yourself. Keep in mind that many instances of Liferay Portal sit behind a firewall without access to the Internet. Because of this, if you are making your software available to Internet users, some of them will have to download it manually anyway, because their installations are firewalled. In this case, the Software Catalog portlet is the easiest way to provide a user interface for downloading your software.

If you still wish to use a text editor to create your software catalog, you can. To manually create a software catalog, obtain the DTD for the XML file from Liferay's source code. You will find this DTD in the *definitions* folder in the Liferay source. It is a file called `liferay-plugin-package_6_0_0.dtd`. Use this DTD with a validating XML editor (a good, free choice is jEdit with all the XML plugins) to create your software catalog manually.

Connecting to a Software Catalog

If there is a software catalog of plugins that you would like to point your instance of Liferay to, all you need is the URL to the catalog. Once you have the URL, go to the Plugin Installer in your Control Panel and click the *Configuration* tab. You will see that there are two fields in which you can enter URLs to plugin repositories: *Trusted Plugin Repositories* and *Untrusted Plugin Repositories*. Currently, the only difference between the two is to provide a visual cue for administrators as to which repositories are trusted and untrusted.

Enter the URL to the repository to which you wish to connect in one of the fields and click *Save*. The portlet will connect to the repository, and items from this repository will be shown in the list.

Liferay Services Oriented Architecture

Liferay includes a utility called the *Service Builder* which is used to generate all of the low level code for accessing resources from the portal database.

This utility is further explained in *Liferay in Action*, but it is mentioned here because of its feature which generates interfaces not only for Java code, but also for web services and JavaScript. This means that the method calls for storing and retrieving portal objects are all the same, and are generated in the same step.

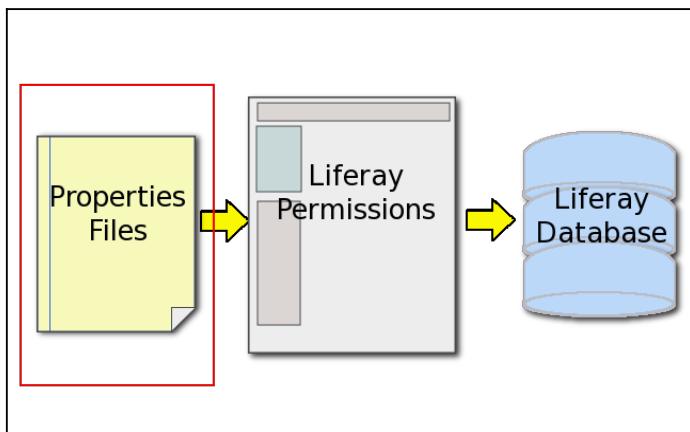


Illustration 101: Liferay SOA's first layer of security.

Because the actual method calls for retrieving data are the same regardless of how one gets access to those methods (i.e., locally or through web services), Liferay provides a consistent interface for accessing portal data that few other products can match. The actual interfaces for the various services are covered in *Liferay in Action*, but before they can be used there are steps that need to be taken to enable users to access those services remotely.

In the default `portal.properties` file, there is a section called **Main Servlet**. This section defines the security settings for all of the remote services provided by Liferay. Copy this section and paste it into your custom `portal-ext.properties` file, and you can configure security settings for the Axis Servlet, the Liferay Tunnel Servlet, the Spring Remoting Servlet, the JSON Tunnel Servlet, and the WebDAV servlet.

By default, a user connecting from the same machine Liferay is running on can access remote services so long as that user has the permission to use those services in Liferay's permissions system. Of course, you are not really "remote" unless you are accessing services from a different machine. Liferay has two layers of security when it comes to accessing its services remotely. Without explicit rights to both layers, a remote exception will be thrown and access to those services will not be granted.

The first layer of security that a user needs to get through in order to call a method from the service layer is servlet security. The *Main Servlet* section of

the `portal-ext.properties` file is used to enable or disable access to Liferay's remote services. In that section of the properties file, there are properties for each of Liferay's remote services.

You can set each service individually with the security settings that you require. For example, you may have a batch job which runs on another machine in your network. This job looks in a particular shared folder on your network and uploads documents to your community's document library portlet on a regular basis, using Liferay's web services. To enable this batch job to get through the first layer of security, you would modify the `portal-ext.properties` file and put the IP address of the machine on which the batch job is running in the list for that particular service. For example, if the batch job uses the Axis web services to upload the documents, you would enter the IP address of the machine on which the batch job is running to the `axis.servlet.hosts.allowed` property. A typical entry might look like this:

```
axis.servlet.hosts.allowed=192.168.100.100, 127.0.0.1, SERVER_IP
```

If the machine on which the batch job is running has the IP address `192.168.100.100`, this configuration will allow that machine to connect to Liferay's web services and pass in user credentials to be used to upload the documents.

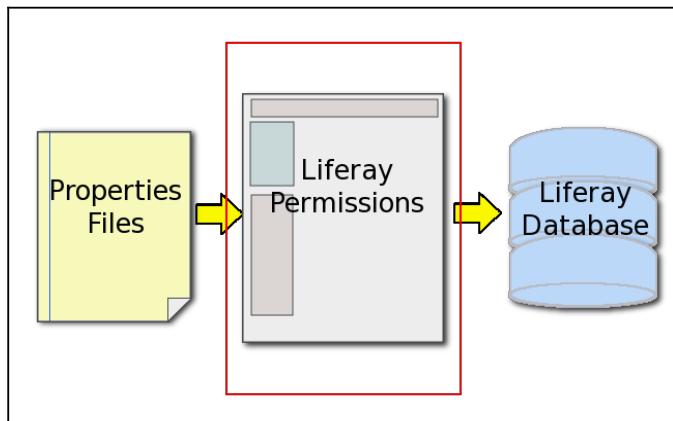


Illustration 102: Liferay SOA's second layer of security.

The second layer of security is Liferay's security model that it uses for every object in the portal. The user ID that accesses the services remotely must have the proper permission to operate on the objects it will be accessing. Otherwise, a remote exception will be thrown. The Portal Administrator will need to make use of Liferay's usual means of granting access to these resources to the user ID that will be operating on them remotely.

For example, say that a Document Library folder called *Documents* has been set up in a community. A role has been created called *Document Upload-*

ers which has the rights to add documents to this folder. Your batch job will be accessing Liferay's web services in order to upload documents into this folder. In order for this to work, you will have to call the web service using a user ID that is a member of this group (or that has individual rights to add documents to this folder). Otherwise, you will be prevented from using the Web Service.

To call the web service using credentials, you would use the following URL syntax:

```
http://" + userIdAsString + ":" + password + "<server.com>:<port>/tunnel-web/secure/axis/" + serviceName
```

The user ID is the user's ID from the Liferay database. This may be obtained by logging in as the user and clicking *My Account* from the Dock. In the top left corner of the portlet that appears is the user ID.

For example, to get Organization data using a user that has the ID of 2 with a password of *test*, you would use the following URL:

```
http://2:test@localhost:8080/tunnel-web/secure/axis/Portal_OrganizationService
```

It is important to note here how *Password Policies* (covered in Chapter 3) can be used in combination with this feature. If you are enforcing password policies on your users (requiring them to change their passwords on a periodic basis, etc.), any administrative ID which accesses Liferay's web services in a batch job will have its password expire too.

To prevent this from happening, you can add a new password policy which does not enforce the password expiration and add your administrative user ID to it. Then your batch job can run as many times as you need it to, and the administrative ID's password will never expire.

In summary, accessing Liferay remotely requires the successful passing of two security checks:

1. The IP address must be pre-configured in the server's portal-ext.-properties file.
2. The user ID being used must have permission to access the resources it is attempting to access.

Accessing Liferay's WSDL

After configuring the security settings properly, your first step in obtaining access to remote web services is to access the WSDL. If you are on a browser on the same machine Liferay is running on, you can do this by accessing the following URL:

```
http://localhost:<port number>/tunnel-web/axis
```

If, for example, you are running on Tomcat on port 8080, you would specify this URL:

```
http://localhost:8080/tunnel-web/axis
```

If you are accessing a web service that was created as part of a portlet plugin, the URL is similar, but uses the context of your application rather than the tunnel-web servlet. You can get a list of your Service Builder-generated WSDL documents by using the URL pattern below:

```
http://localhost:8080/your-portlet/axis
```

If you are on a different machine from the Liferay server, you will need to pass in your user credentials on the URL to access the WSDL:

```
http://<user ID>:<password>@<server name>:<port number>/tunnel-web/axis
```

In any case, once you successfully browse to this URL, you will see the list of web services.

WSDL for each service is available by clicking on the *WSDL* link next to the name of the service. There are many services; one for each of the services available from the Liferay API.

Once you click on one of the WSDL links, the Web Service Definition Language document will be displayed. This document can be used to generate client code in any language that supports it. You can either save the document to your local machine and then generate the client code that way, or use your tool to trigger Liferay to generate the document dynamically by using one of the URLs above.

For further information about developing applications that take advantage of Liferay's remote services, please see *Liferay in Action*.

Summary

This very long chapter covered a cornucopia of topics. First, we went through all of the options which can be customized in your `portal-ext.properties` file, exhaustively. This serves as a reference section for the file, so you can quickly find documentation for any property you might encounter.

Next, we took a good, long look at plugin management. Every aspect of plugin management was covered, including plugin types, installing plugins from a repository, installing plugins manually, troubleshooting, creating your own plugin repository, and more.

After this, we covered how to configure Liferay's web services in a secure way so that you can access the back end services you need. We also saw how

to access web services that your own developers have created with Liferay's Service Builder utility.

Combined with the preceding chapters, the information you've gained so far gives you the ability to competently install and configure a website which runs on Liferay. Next, we'll take a look at some of the ways you can configure Liferay for the enterprise.

7. ENTERPRISE CONFIGURATION

Liferay Portal is a robust, enterprise-ready portal solution. As such, it is fully ready to support mission-critical, enterprise applications in an environment configured for multiple redundancy and 24/7 up times. The product, however, like other products of its kind, does not come configured this way out of the box, and so there are some steps that need to be taken in order to configure it this way.

This chapter will cover these topics in detail. Because Liferay runs on so many different Java EE application servers, it will be impossible to cover all of the differences between these application servers. For this reason, we will cover the configuration of Liferay only. As an example, we will cover how to configure Liferay to work in a clustered environment, but we will not cover how to create the cluster in your application server. Please consult the documentation for your particular application server to see how you can configure your application server of choice to work as a cluster.

We will, however, cover the configuration of Liferay for a number of advanced scenarios, such as

- Clustering and Distributed Caching
- Liferay Workflow
- Deploying Customized versions of Liferay
- Performance Testing and Tuning

During this discussion, we will mention a number of other open source products upon which Liferay relies for much of this functionality. These products all have their own documentation which should be consulted for a

fuller view of what these products can do. For example, Liferay uses Ehcache for its caching mechanism. We will cover how to configure Ehcache to enable various caching functionality in Liferay, but will refer you to that product's documentation for further information about that product.

Sometimes Liferay supports multiple products which perform the same function. There are, for example, multiple implementations of Enterprise Service Buses for use with workflow, and Liferay supports several of them. We will leave it up to you to select which product best fits the needs of your project without recommending one product over another.

With all of that said, let's get started configuring Liferay for the enterprise.

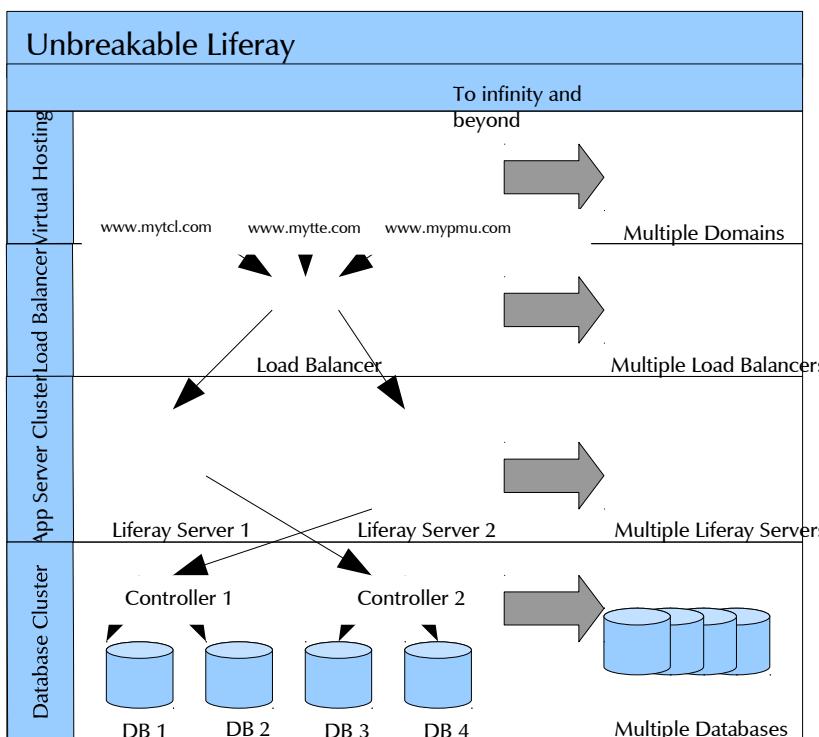


Illustration 103: "Unbreakable" Liferay architecture

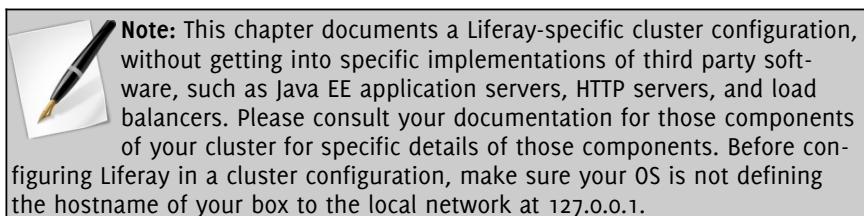
Liferay Clustering

Once you have Liferay installed in more than one node on your application server, there are several optimizations that need to be made. At a min-

imum, Liferay should be configured in the following way for a clustered environment:

- All nodes should be pointing to the same Liferay database
- Jackrabbit, the JSR-170 content repository, should be:
 - On a shared file system available to all the nodes (not really recommended, though), or
 - In a database that is shared by all the nodes
- Alternatively, the Document Library should be configured to use the File System Hook, and the files can be stored on a SAN for better performance.
- Similarly, Lucene, the full text search indexer, should be:
 - On a shared file system available to all the nodes (not really recommended, though), or
 - In a database that is shared by all the nodes, or
 - On separate file systems for all of the nodes, or
 - Disabled, and a separate pluggable enterprise search server configured (recommended).
- If you have not configured your application server to use farms for deployment, the hot deploy folder should be a separate folder for all the nodes, and plugins will have to be deployed to all of the nodes individually. This can be done via a script.

Many of these configuration changes can be made by adding or modifying properties in your `portal-ext.properties` file. Remember that this file overrides the defaults that are in the `portal.properties` file. The original version of this file can be found in the Liferay source code or can be extracted from the `portal-impl.jar` file in your Liferay installation. It is a best practice to copy the relevant section that you want to modify from `portal.properties` into your `portal-ext.properties` file, and then modify the values there.



All Nodes Should Be Pointing to the Same Liferay Database

This is pretty self-explanatory. Each node should be configured with a data source that points to one Liferay database (or a database cluster) that all of the nodes will share. This ensures that all of the nodes operate from the same basic data set. This means, of course, that Liferay cannot (and should not) use the embedded HSQL database that is shipped with the bundles. It is also best if the database server is a separate physical box from the server which is running Liferay.

Document Library Configuration

There are several options available for configuring how Liferay's document library stores files. Each option is a hook which can be configured through the `portal-ext.properties` file by setting the `dl.hook.impl=` property.

Default File System Hook

This is the hook that Liferay will use to manage your document library by default. It uses the file system to store documents which has proven to be the highest performing configuration for large document libraries. You can use the file system for your clustered configuration, but the advanced file system hook is generally recommended for more complex clustering environments.

You can configure the path where your documents are stored by setting the `dl.hook.file.system.root.dir=` property in your `portal-ext.properties`.

Jackrabbit Sharing

Liferay uses Jackrabbit—which is a project from Apache—as its JSR-170 compliant document repository. By default, Jackrabbit is configured to store the documents on the local file system upon which Liferay is installed, in the `$LIFERAY_HOME/liferay/jackrabbit` folder. Inside this folder is Jackrabbit's configuration file, called `repository.xml`.

To simply move the default repository location to a shared folder, you do not need to edit Jackrabbit's configuration file. Instead, find the section in `portal.properties` labeled **JCR** and copy/paste that section into your `portal-ext.properties` file. One of the properties, by default, is the following:

```
jcr.jackrabbit.repository.root=${liferay.home}/data/jackrabbit
```

Change this property to point to a shared folder that all of the nodes can see. A new Jackrabbit configuration file will be generated in that location.

Note that because of file locking issues, this is not the best way to share Jackrabbit resources. If you have two people logged in at the same time uploading content, you could encounter data corruption using this method, and because of this, we do not recommend it for a production system. Instead, to enable better data protection, you should redirect Jackrabbit into your database of choice. You can use the Liferay database or another database for this purpose. This will require editing Jackrabbit's configuration file.

The default Jackrabbit configuration file has sections commented out for moving the Jackrabbit configuration into the database. This has been done to make it as easy as possible to enable this configuration. To move the Jackrabbit configuration into the database, simply comment out the sections relating to the file system and comment in the sections relating to the database. These by default are configured for a MySQL database. If you are using another database, you will likely need to modify the configuration, as there are changes to the configuration file that are necessary for specific databases. For example, the default configuration uses Jackrabbit's `DbFileSystem` class to mimic a file system in the database. While this works well in MySQL, it does not work for all databases. For example, if you are using an Oracle database, you will need to modify this to use `OracleFileSystem`. Please see the Jackrabbit documentation at <http://jackrabbit.apache.org> for further information.

You will also likely need to modify the JDBC database URLs so that they point your database. Don't forget to create the database first, and grant the user ID you are specifying in the configuration file access to create, modify, and drop tables.

Once you have configured Jackrabbit to store its repository in a database, the next time you bring up Liferay, the necessary database tables will be created automatically. Jackrabbit, however, does not create indexes on these tables, and so over time this can be a performance penalty. To fix this, you will need to manually go into your database and index the primary key columns for all of the Jackrabbit tables.

All of your Liferay nodes should be configured to use the same Jackrabbit repository in the database. Once that is working, you can create a Jackrabbit cluster (please see the section below).

Other Storage Options

There are other options available to configure Liferay's Document Library. The default option has the best performance with large document libraries, because it simply uses the file system. If you require a JSR-170 compliant document store, you can use Jackrabbit, which can be configured to use the file system or database, depending on your needs. If, however, you have very specific configuration needs, or if you already have a content repository

that you want to continue using, you might want to use one of the following options Liferay has available.

Advanced File System Hook

To use the Advanced File System Hook, set:

```
dl.hook.impl=com.liferay.documentlibrary.util.AdvancedFileSystemHook
```

This is the preferred hook for clustered environments especially if you're using a SAN to store files.

From a performance standpoint, this method is superior to using Jackrabbit. The Advanced File System Hook distributes the files into multiple directories and thus circumvents file system limitations. Liferay does not implement any file level locking, so only use this if you're using a SAN that supports file locking (most modern ones do but check your SAN documentation to be sure).

The path for storing your documents is set also set using the `dl.hook.-file.system.root.dir` property in the `portal-ext.properties`.

Amazon Simple Storage

To use Amazon's Simple Storage Service to store your documents for a Liferay Portal, set

```
dl.hook.impl=com.liferay.documentlibrary.util.S3Hook
```

in `portal-ext.properties`. You will need to consult the Amazon Simple Storage documentation for additional details on setting it up.

Documentum

You can use this hook to store your documents with Documentum. Before configuring Documentum, you will need to install the documentum-hook plugin. Before installing, however, please note that the documentum-hook plugin is only supported on Liferay Enterprise Edition, and is currently in an experimental stage, and may not be ready for production use. To use this hook, set

```
dl.hook.impl=liferay.documentum.hook.DocumentumHook
```

If you are using Documentum, there are additional settings that must be configured in the `${liferay_home}/documentum-hook/docroot/WEB-INF/src/dfc.properties` and `documentum-hook/docroot/WEB-INF/src/portlet.properties` files.

Search Configuration

You can configure search in one of two ways: use pluggable enterprise search (recommended for a cluster configuration) or configure Lucene in such a way that either the index is stored on each node's file system or is shared in a database.

Pluggable Enterprise Search

As an alternative to using Lucene, Liferay supports pluggable search engines. The first implementation of this uses the open source search engine *Solr*, but in the future there will be many such plugins for your search engine of choice. This allows you to use a completely separate product for search, which can be installed on another application server in your environment. Your search engine then operates completely independently of your Liferay Portal nodes in a clustered environment, acting as a search service for all of the nodes simultaneously.

This solves the problem described below with sharing Lucene indexes. You can now have one search index for all of the nodes of your cluster without having to worry about putting it in a database (if you wish, you can still do this if you configure Solr or another search engine that way) or maintaining separate search indexes on all of your nodes. Each Liferay node will send requests to the search engine to update the search index when needed, and these updates are then queued and handled automatically by the search engine, independently.

Configuring the Solr Search Server

Since Solr is a standalone search engine, you will need to download it and install it first according to the instructions on the Solr web site (<http://lucene.apache.org/solr>). Of course, it is best to use a server that is separate from your Liferay installation, as your Solr server will be responsible for all indexing and searching for your entire cluster. Solr is distributed as a .war file with several .jar files which need to be available on your application server's class path. Once you have Solr up and running, integrating it with Liferay is easy, but it will require a restart of your application server.

The first thing you will need to define is the location of your search index. Assuming you are running a Linux server and you have mounted a file system for the index at `/solr`, create an environment variable that points to this folder. This environment variable needs to be called `$SOLR_HOME`. So for our example, we would define:

```
$SOLR_HOME=/solr
```

This environment variable can be defined anywhere you need: in your operating system's start up sequence, in the environment for the user who is

logged in, or in the start up script for your application server. If you are going to use Tomcat to host Solr, you would modify `setenv.sh` or `setenv.bat` and add the environment variable there.

Once you have created the environment variable, you then can use it in your application server's start up configuration as a parameter to your JVM. This is configured differently per application server, but again, if you are using Tomcat, you would edit `catalina.sh` or `catalina.bat` and append the following to the `$JAVA_OPTS` variable:

```
-Dsolr.solr.home=$SOLR_HOME
```

This takes care of telling Solr where to store its search index. Go ahead and install Solr to this box according to the instructions on the Solr web site (<http://lucene.apache.org/solr>). Once it's installed, shut it down, as there is some more configuration to do.

Installing the Solr Liferay Plugin

Next, you have a choice. If you have installed Solr on the same system upon which Liferay is running, you can simply go to the Control Panel and install the `solr-web` plugin. This, however, defeats much of the purpose of using Solr, because the goal is to offload search indexing to another box in order to free up processing for your installation of Liferay. For this reason, you should not run Liferay and your search engine on the same box. Unfortunately, the configuration in the plugin defaults to having Solr and Liferay running on the same box, so to run them separately, you will have to make a change to a configuration file in the plugin before you install it so you can tell Liferay where to send indexing requests. In this case, go to the Liferay web site (<http://www.liferay.com>) and download the plugin manually.

Open or extract the plugin. Inside the plugin, you will find a file called `solr-spring.xml` in the `WEB-INF/classes/META-INF` folder. Open this file in a text editor and you will see the entry which defines where the Solr server can be found by Liferay:

```
<bean  
class="com.liferay.portal.spring.context.PortletBeanFactoryPostProcessor" />  
<bean id="com.liferay.portal.search.solr.server.BasicAuthSolrServer"  
class="com.liferay.portal.search.solr.server.BasicAuthSolrServer">  
<constructor-arg  
type="java.lang.String" value="http://localhost:8080/solr" />
```

Modify this value so that they point to the server upon which you are running Solr. Then save the file and put it back into the plugin archive in the same place it was before.

Next, extract the file `schema.xml` from the plugin. It should be in the `docroot/WEB-INF/conf` folder. This file tells Solr how to index the data com-

ing from Liferay, and can be customized for your installation. Copy this file to `$SOLR_HOME/conf` (you may have to create the `conf` directory) on your Solr box. Now you can go ahead and start Solr.

You can now hot deploy the `solr-web` plugin to all of your nodes. See the next section for instructions on hot deploying to a cluster.

Once the plugin is hot deployed, your Liferay search is automatically upgraded to use Solr. It is likely, however, that initial searches will come up with nothing: this is because you will need to reindex everything using Solr.

Go to the Control Panel. In the *Server* section, click *Server Administration*. Click the *Execute* button next to *Reindex all search indexes* at the bottom of the page. It may take a while, but Liferay will begin sending indexing requests to Solr for execution. When the process is complete, Solr will have a complete search index of your site, and will be running independently of all of your Liferay nodes.

Installing the plugin to your nodes has the effect of overriding any calls to Lucene for searching. All of Liferay's search boxes will now use Solr as the search index. This is ideal for a clustered environment, as it allows all of your nodes to share one search server and one search index, and this search server operates independently of all of your nodes.

Lucene Configuration

Lucene, the search indexer which Liferay uses, can be in a shared configuration for a clustered environment, or an index can be created on each node of the cluster. The easiest configuration to implement is to have an index on each node of the cluster. Liferay provides a method called ClusterLink which can send indexing requests to all nodes in the cluster to keep them in sync. This configuration does not require any additional hardware, and it performs very well. It may increase network traffic when an individual server reboots, since in that case a full reindex will be needed. But since this should only rarely happen, it's a good tradeoff if you don't have the extra hardware to implement a Solr search server.

You can enable ClusterLink by setting one property in your `portal-ext.properties` file:

```
cluster.link.enabled=true
```

Of course, this needs to be set on all the nodes.

If you wish to have a shared index, you will need to either share the index on the file system or in the database. This requires changing your Lucene configuration.

The Lucene configuration can be changed by modifying values in your `portal-ext.properties` file. Open your `portal.properties` file and search for the text `Lucene`. Copy that section and then paste it into your `portal-ext.properties` file.

If you wish to store the Lucene search index on a file system that is shared by all of the Liferay nodes, you can modify the location of the search index by changing the `lucene.dir` property. By default, this property points to the `lucene` folder inside the Liferay home folder:

```
lucene.dir=${liferay.home}/data/lucene/
```

Change this to the folder of your choice. To make the change take effect, you will need to restart Liferay. You can point all of the nodes to this folder, and they will use the same index.

Like Jackrabbit, however, this is not the best way to share the search index, as it could result in file corruption if different nodes try reindexing at the same time. We do not recommend this for a production system. A better way is to share the index via a database, where the database can enforce data integrity on the index. This is very easy to do; it is a simple change to your `portal-ext.properties` file.

There is a single property called `lucene.store.type`. By default this is set to go to the file system. You can change this so that the index is stored in the database by making it the following:

```
lucene.store.type=jdbc
```

The next time Liferay is started, new tables will be created in the Liferay database, and the index will be stored there. If all the Liferay nodes point to the same database tables, they will be able to share the index. Performance on this is not always as good as it could be. Your DBAs may be able to tweak the database indexes a bit to improve performance. For better performance, you should consider using a separate search server (see the section on Solr above).

Note: MySQL users need to modify their JDBC connection string for this to work. Add the following parameter to your connection string:

```
emulateLocators=true
```

Alternatively, you can leave the configuration alone, and each node will then have its own index. This ensures that there are no collisions when multiple nodes update the index, because they all will have separate indexes. This, however, creates duplicate indexes and may not be the best use of resources. Again, for a better configuration, you should consider using a separate search server (see the section on Solr above).

Hot Deploy

Plugins which are hot deployed will need to be deployed separately to all of the Liferay nodes. Each node should, therefore, have its own hot deploy folder. This folder needs to be writable by the user under which Liferay is running, because plugins are moved from this folder to a temporary folder when they are deployed. This is to prevent the system from entering an endless loop, because the presence of a plugin in the folder is what triggers the hot deploy process.

When you want to deploy a plugin, copy that plugin to the hot deploy folders of all of the Liferay nodes. Depending on the number of nodes, it may be best to create a script to do this. Once the plugin has been deployed to all of the nodes, you can then make use of it (by adding the portlet to a page or choosing the theme as the look and feel for a page or page hierarchy).

Some containers contain a facility which allows the end user to deploy an application to one node, after which it will get copied to all of the other nodes. If you have configured your application server to support this, you won't need to hot deploy a plugin to all of the nodes—your application server will handle it transparently. Make sure, however, that you use Liferay's hot deploy mechanism to deploy plugins, as in many cases Liferay slightly modifies plugin .war files when hot deploying them.

All of the above will get basic Liferay clustering working; however, the configuration can be further optimized. We will see how to do this next.

Distributed Caching

Liferay uses **Ehcache**, which has robust distributed caching support. This means that the cache can be distributed across multiple Liferay nodes running concurrently. Enabling this cache can increase performance dramatically. For example, say that two users are browsing the message boards. The first user clicks on a thread in order to read it. Liferay must look up that thread from the database and format it for display in the browser. With a distributed Ehcache running, this thread can be pulled from the database and stored in a cache for quick retrieval. Say then that the second user wants to read the same forum thread and clicks on it. This time, because the thread is in the local cache, no trip to the database is necessary, and so retrieving the data is much faster.

This could be done by simply having a cache running separately on each node, but the power of *distributed* caching allows for more functionality. The first user can post a message to the thread he or she was reading, and the cache will be updated across all of the nodes, making the new post available immediately from the local cache. Without that, the second user would need

to wait until the cache was invalidated on the node he or she connected to before he or she could see the updated forum post.

Configuring distributed caching requires the modification of the portal-ext.properties file as well as one or more other files depending on what you want to cache. The first thing you will want to do is determine where on your server you will want to store your cache configuration files. This will have to be somewhere on Liferay's class path, so you will need to find where your application server has stored the deployed version of Liferay, and create a folder in Liferay's WEB-INF/classes folder to store the files. Because the original, default files are stored inside of a .jar file, you will need to extract them to this area and then tell Liferay (by use of the portal-ext.properties file) where they are.

For example, say you are running Liferay on Tomcat. Tomcat stores the deployed version of Liferay in <Tomcat Home>/webapps/ROOT. Inside of this folder is the folder structure WEB-INF/classes. You can create a new folder in here called *myehcache* to store the custom versions of the cache configuration files. Copy the files from the /ehcache folder—which is inside the portal-impl.jar file—into the myehcache folder you just created. You then need to modify the properties in portal-ext.properties that point to these files. Copy / paste the **Hibernate** section of portal.properties into your portal-ext.properties file and then modify the net.sf.ehcache.configurationResourceName property to point to the clustered version of the configuration file that is now in your custom folder:

```
net.sf.ehcache.configurationResourceName=/myehcache/hibernate-clustered.xml
```

Now that Liferay is pointing to your custom file, you can modify the settings in this file to change the cache configuration for Hibernate.

Next, copy / paste the *Ehcache* section from the portal.properties file into your portal-ext.properties file. Modify the properties so that they point to the files that are in your custom folder. For example:

```
ehcache.multi.vm.config.location=/myehcache/liferay-multi-vm.xml
```

If you are going to enable distributed clustering, uncomment the following line and point it to your custom version of the file:

```
ehcache.multi.vm.config.location=/myehcache/liferay-multi-vm-clustered.xml
```

You can now take a look at the settings in these files and tune them to fit your environment and application.

Alternatively, if your Liferay project is using the Ext plugin to make customizations to Liferay, you can place your cache configuration in the extension environment. The settings there will override the default settings that ship with Liferay. If you wish to do this, you can create new versions of the files in ext-impl/src/ehcache. The files should be have with -ext.xml

tacked onto the end of the file name. For example, the custom version of `hibernate.xml` should be called `hibernate-ext.xml`, and the custom version of `liferay-multi-vm-clustered.xml` should be called `liferay-multi-vm-clustered-ext.xml`. You can then modify the files and tune them to fit your environment / application, and they will be deployed along with the rest of your extension environment.

Hibernate Cache Settings

By default, Hibernate (Liferay's database persistence layer) is configured to use Ehcache as its cache provider. This is the recommended setting. The default configuration, however, points to a file that does not have clustering enabled. To enable clustering, copy the *Hibernate* section from `portal.properties` into your `portal-ext.properties` file. To enable a clustered cache, comment out the default file (`hibernate.xml`) and uncomment the clustered version of the file, making sure that you change the path so that it points to your custom version of the file:

```
net.sf.ehcache.configurationResourceName=/myehcache/hibernate-clustered.xml
```

Next, open this file in a text editor. You will notice that the configuration is already set up to perform distributed caching through a multi-cast connection. It is likely, however, that the configuration is not set up optimally for your particular application. You will notice that by default, the only object cached in the Hibernate cache is the User object (`com.liferay.portal.model.impl.UserImpl`). This means that when a user logs in, his or her User object will go in the cache so that any portal operation that requires access to it (such as permission checking) can retrieve that object very quickly from the cache.

You may wish to add other objects to the cache. For example, a large part of your application may be document management using the Document Library portlet. In this case, you may want to cache Document Library objects, such as `DLFileEntryImpl` in order to improve performance as users access documents. To do that, add another block to the configuration file with the class you want to cache:

```
<cache  
    name="com.liferay.portlet.documentlibrary.model.impl.DLFileEntryImpl"  
    maxElementsInMemory="10000"  
    eternal="false"  
    timeToIdleSeconds="600"  
    overflowToDisk="true"  
>  
    <cacheEventListenerFactory  
        class="net.sf.ehcache.distribution.RMICacheReplicatorFactory"  
        properties="replicatePuts=false,replicateUpdatesViaCopy=false"  
        propertySeparator=","
```

```
>
<bootstrapCacheLoaderFactory
class="net.sf.ehcache.distribution.RMIBootstrapCacheLoaderFactory" />
</cache>
```

Your site may use the message boards portlet, and those message boards may get a lot of traffic. To cache the threads on the message boards, configure a block with the `MBMessageImpl` class:

```
<cache
  name="com.liferay.portlet.messageboards.model.impl.MBMessageImpl"
  maxElementsInMemory="10000"
  eternal="false"
  timeToIdleSeconds="600"
  overflowToDisk="true"
>
  <cacheEventListenerFactory
    class="net.sf.ehcache.distribution.RMICacheReplicatorFactory"
    properties="replicatePuts=false,replicateUpdatesViaCopy=false"
    propertySeparator=","
  />
  <bootstrapCacheLoaderFactory
class="net.sf.ehcache.distribution.RMIBootstrapCacheLoaderFactory" />
</cache>
```

Note that if your developers have overridden any of these classes, you will have to specify the overridden versions rather than the stock ones that come with Liferay Portal.

As you can see, it is easy to add specific data to be cached. Be careful, however, as too much caching can actually reduce performance if the JVM runs out of memory and starts garbage collecting too frequently. You will likely need to experiment with the memory settings on your JVM as well as the cache settings above. You can find the specifics about these settings in the documentation for Ehcache.

Clustering Jackrabbit

If you are using the Document Library, can configure it to use the JSR-170 document repository, which is the Apache product *Jackrabbit*. You have already configured basic data sharing among nodes by moving its configuration into a database. The next thing you need to do is configure clustering for Jackrabbit, so that each node knows about data being entered into the repository by other nodes.

You can find the Jackrabbit configuration file in `[Liferay Home]/liferay/jackrabbit`. The file is called `repository.xml`. You have likely already edited this file when you modified the configuration to move the data into the database.

At the bottom of this file is a cluster configuration that is commented out. If you are using a MySQL database, you can uncomment this section and use it as-is. You will need to change the cluster ID for each node so that they don't conflict with one another.

If you are using another database, the only changes necessary are the connection, credentials, and schema settings. Modify them according to your database of choice and then save the file. This is all it takes to set up clustering for Jackrabbit.

Workflow with Kaleo

Liferay Portal includes a workflow engine called Kaleo. In Greek, this word means “called ones,” which is appropriate for a workflow engine that will be calling users to participate in a process that has been designed for them.

Kaleo workflow allows a user to define any number of simple to complex business processes/workflows, deploy them, and manage them through a portal interface. Those processes have knowledge of users, groups, and roles without writing a single line of code—it only requires the creation of a single XML document.

Installation

Liferay's Kaleo workflow engine ships with CE versions of Liferay. If you have EE or if you have uninstalled it, the plugin can be installed through the built-in plugin repository. The name is `kaleo-web`, and you'll find it in the list of web plugins. Installing the plugin will add a number of new options to the Control Panel.

- My Workflow Tasks
- Workflow Configuration
- My Submissions
- Workflow

There is one default workflow that is bundled with the `kaleo-web` plugin: Single Approver Workflow. This workflow requires one approval before an asset can be published. One of the conveniences of using Liferay's workflow engine is that any roles that are specified in the workflow definition are created automatically when the definition is deployed. This provides a level of integration with the portal that third party engines cannot match. The Single Approver Workflow contains three roles each with different scopes. The scope of each role can be easily deduced by their names - Community Content Reviewer, Organization Content Reviewer, and Portal Content Reviewer.

Kaleo Workflow in a Nutshell

Liferay's Kaleo workflows are defined in an XML file and executed by users on the portal. Administrators can create as many different workflow definitions as they need to manage the work done on their portal. You can define new user roles in the workflow to manage the approval process or use roles that already exist in your portal.

Process Definitions

Each workflow definition is defined by a single XML file. The XML file has several parts which define the workflow. To get an idea of how this works, we'll be examining the default `single-approver-definition.xml` file which is included in the the Liferay Kaleo plugin.

The key parts of the workflow definition are the asset, states, transitions, and tasks. The asset is whatever piece of content is being reviewed and approved in the workflow. States represent stages of the workflow; for example: created, rejected, or approved. Transitions occur between states, and indicate what the next state should be. Tasks are steps in the workflow that require user action.

Generally speaking a state will contain a task and the user input from the task will determine which transition will occur. The transition will then move the workflow to the next task. This cycle will continue until the end “approved” state is reached.

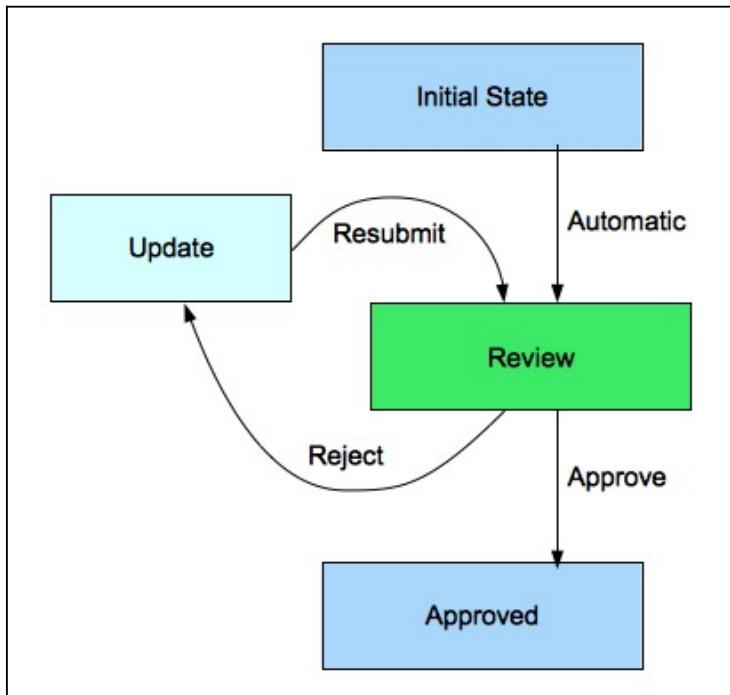


Illustration 104: The default single approver workflow. Arrows represent transitions, and boxes represent states and tasks.

First we define the schema. For Liferay workflows using Kaleo, `liferay-workflow-definition-6_0_0.xsd` should be your schema. You can find this schema in the definitions folder of the Liferay source or a good XML editor should be able to cache it from Liferay's website.

```

<workflow-definition
    xmlns="urn:liferay.com:liferay-workflow_6.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:liferay.com:liferay-workflow_6.0.0
    http://www.liferay.com/dtd/liferay-workflow-definition_6_0_0.xsd"
>
  
```

Next we define a name and description for the workflow. This will appear in the Control Panel when you are choosing and configuring workflows.

```

<name>Single Approver</name>
<description>A single approver can approve a workflow content.</description>
<version>1</version>
  
```

After that, we need to define our initial state. In this case, it is simply that the asset has been created. States can contain actions and transitions. Actions can contain scripts. You can specify the language of the script with

the `<script-language>` tag. Scripts can be written in Groovy, JavaScript, Ruby, or Python. Transitions will move you to a new state or task.

```
<state>
    <name>created</name>
    <initial>true</initial>
```

From the initial state, we transition to a new state where the asset will need to be reviewed.

```
<transitions>
    <transition>
        <name>review</name>
        <target>review</target>
        <default>true</default>
    </transition>
</transitions>
</state>
```

After that, we create a task. The task has several parts, and is the most complex part of the definition. In the task, we first need to choose a role to notify that there's new content which is in need of review. If the role doesn't exist, defining it here causes it to be created automatically.

The first task listed in the `singleApproverDefinition.xml` workflow definition is actually not the first task in the workflow. It is the *update* task. This is the task that will be assigned by the workflow if the asset is rejected by an approver. It is listed first because it is the default task: when this task is triggered, the workflow process has been reset back to the beginning. In this task, the asset is assigned back to the content creator, who will receive an email notification and be required to resubmit the asset. Once the task is resubmitted, it goes back to the review stage.

You can also see that the task is assigned to `<user />`. This tag will always assign the task back to the user who created the asset.

```
<task>
    <name>update</name>
    <actions>
        <notification>
            <name>Creator Modification Notification</name>
            <execution-type>onAssignment</execution-type>
            <template>Your submission was rejected by a reviewer,
            please modify and resubmit.</template>
            <template-language>text</template-language>
            <notification-type>email</notification-type>
        </notification>
    </actions>
    <assignments>
        <user />
    </assignments>
</task>
```

```

</assignments>
<transitions>
    <transition>
        <name>resubmit</name>
        <target>review</target>
        <default>true</default>
    </transition>
</transitions>
</task>

```

The review task is the first task in the workflow . This is where users on the portal need to review the content to move to the next step.

Once the transition has been made to this task, a notification is sent out to those who are assigned to the task. You can edit the name or content of the notification in the XML file.

```

<task>
    <name>review</name>
    <actions>
        <notification>
            <name>Review Notification</name>
            <execution-type>onAssignment</execution-type>
            <template>You have a new submission waiting for your review
            in the workflow.</template>
            <template-language>text</template-language>
            <notification-type>email</notification-type>
        </notification>
    </actions>

```

You must also assign the task to a specific role or roles. This role does not have to be the role which you notified if, for example, you wanted to notify all of the content creators any time a new item was submitted. Regardless of who else you are notifying, you will definitely want to notify anyone who will be responsible for approving content.

Notifications need an execution-type which can be onAssignment, onEntry, or onExit.

- **onEntry** generates and sends the notification when the user logs in to the portal.
- **onExit** generates and sends the notification when the user logs out.
- **onAssignment** generates and sends the notification as soon as the user is assigned the task in the workflow. onAssignment notification will not work if you wish to notify a user that is not part of the workflow.

Notifications also need a notification-type which can be email, im, or private-message. Your notification type and execution type should complement each other. You wouldn't generally want to use an onExit execution type with a private message, because the user won't receive that message until he or she logs back in. Generally speaking, email notifications work best with onExit or onAssignment, while IM or private message work better with onEntry.

Email and private message notifications can also be created as plain text, or you can create formatted content using Freemarker or Velocity templating languages. When creating the notification, you need to specify the template-language as text, freemarker, or velocity.

In this workflow, anyone who would be capable of approving the content is notified onAssignment. This includes administrators, and community and organization owners. The role-type tag helps the system sort out who should be receiving the notification based on the scope, and can be set as *community*, *organization*, or *portal*.

```
<assignments>
    <roles>
        <role>
            <role-type>community</role-type>
            <name>Community Administrator</name>
        </role>
        <role>
            <role-type>community</role-type>
            <name>Community Content Reviewer</name>
        </role>
        <role>
            <role-type>community</role-type>
            <name>Community Owner</name>
        </role>
        <role>
            <role-type>organization</role-type>
            <name>Organization Administrator</name>
        </role>
        <role>
            <role-type>organization</role-type>
            <name>Organization Content Reviewer</name>
        </role>
        <role>
            <role-type>organization</role-type>
            <name>Organization Owner</name>
        </role>
        <role>
            <role-type>regular</role-type>
            <name>Portal Content Reviewer</name>
        </role>
    </roles>
</assignments>
```

```

        </role>
        <role>
            <role-type>regular</role-type>
            <name>Administrator</name>
        </role>
    </roles>
</assignments>
```

Once the content is approved, you will want to transition to a new state. In this case, we have only need of a single approver, so we will transition to the final approved state. In more complex workflows, this might transition to a second tier approver.

```

<transitions>
    <transition>
        <name>approve</name>
        <target>approved</target>
        <default>true</default>
    </transition>
    <transition>
        <name>reject</name>
        <target>update</target>
        <default>false</default>
    </transition>
</transitions>
</task>
```

Finally, we define our end state, which runs a script to set the state of the content to approved, in the portal. The script is part of the Kaleo workflow engine. Any workflow customizations can be completely contained within XML workflow definitions.

You could also write a customized script if there were actions outside of the standard one that need to be performed on your asset. The default script, written in Javascript, sets the status of the asset to *approved*. You can add additional information into the script through Javascript, or you can change the to another supported language (Ruby, Groovy, or Python) and rewrite the action with additional details to meet your needs.

```

<state>
    <name>approved</name>
    <actions>
        <action>
            <name>approve</name>
            <execution-type>onEntry</execution-type>
            <script>
                <![CDATA[
```

```
Packages.com.liferay.portal.kernel.workflow.WorkflowStatusManagerUtil.update
```

```
status(Packages.com.liferay.portal.kernel.workflow.WorkflowConstants.toStatus("approved"), workflowContext);
    ]]>
</script>
<script-language>javascript</script-language>
</action>
</actions>
</state>
```

To create longer workflows, you would simply create additional states, tasks and transitions similar to the ones in the single approver definition, and create additional reviewer roles. For instance, if you wanted to have a second level of review before an item is approved, you could create a new task in between the *review* task and the *approved* state. The task itself would have similar content to *review*, but you would assign it to a different role – either one that you have already created, or a new one generated by Kaleo. You would set the *review* task to transition to your new task, and set the new task to transition to the *approved* state, once it is completed. You can also use *forks* and *joins* to create more complex workflows.

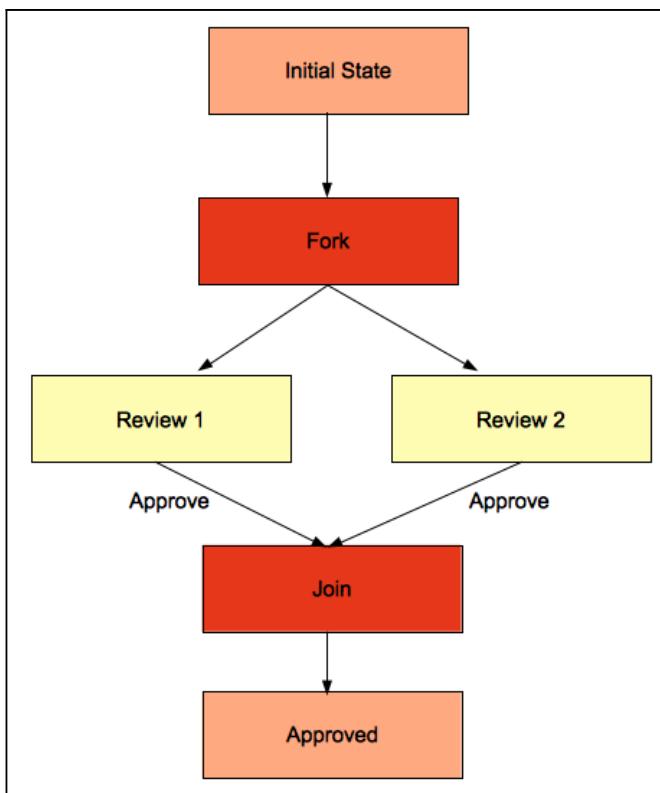


Illustration 105: A possible parallel approval design

You can transition to a fork from a task or state, and from a fork, you can transition to multiple tasks or states which will occur in parallel. In the above example, when we have multiple transitions from one task, they are mutually exclusive -- you either trigger one or the other. The transitions are also serial, meaning that one must occur, and then the next one can occur. With a parallel workflow, you can have different approvals going through different users at the same time. For example, you could use this to separate two different departments' approval chains on a single asset.

A fork should be formatted like this:

```
<fork>
  <name></name>
  <actions>
    ...
  </actions>
  <transitions>
    ...
  </transitions>
</fork>
```

To bring a fork back together, you would transition both nodes of the fork back to a single join. A join is formatted similarly to a fork, except that any transitions are serial, not parallel.

Due Dates

When you're creating a task for a workflow, you can configure due date options. The due date for an item isn't set as a specific day, but as a period of time after the task is assigned. For example, you could set the due date for a specific task to be two days after the task is assigned. This is all configured in the XML definition file, and there is currently no GUI option to configure this setting.

The due date options are formatted in the definitions file like this:

```
<task>
  <name></name>
  <description/></description>
  <due-date-duration>{any whole number}</due-date-duration>
  <due-date-scale>{second, minute, hour, day, week, month, year}<due-date-
scale>
  ...
</task>
```

The due date is set inside the task with the two elements: a duration and a scale. The duration can be any whole number, and is completely meaningless without the scale. The scale tells you what units the duration is measured in, valid values for this are *second*, *minute*, *hour*, *day*, *week*, *month*, and *year*.

Here's an example of how this can work practically: you could set the duration to 10, and then set the scale to be *hour*. This would mean that the task would be due 10 hours after it was assigned. If you edited the definition file, and changed *hour* to *day*, that would mean that the task would need to be completed within 10 days after it was assigned.

Workflow in the Control Panel

Most of your workflow configuration is done via the Control Panel. Everything you need to do in the portal can be done through simple GUI controls.

Workflow

Workflow is found under the Portal heading in the Control Panel. There are three options under Workflow. *Definitions*, *Default Configuration*, and *Submissions*.

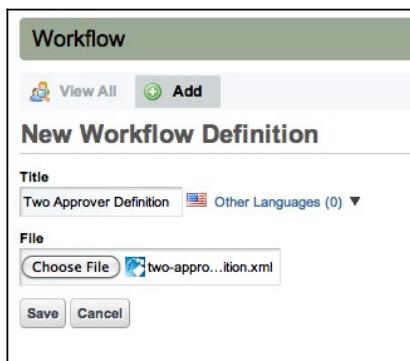


Illustration 106: Adding a workflow definition

Before you do anything else, you'll need to add workflow definitions through the Definitions to make them available. By default only the Single Approver workflow appears here. Clicking Add allows you to enter a title for a new workflow definition and upload the XML file. Once you add a file here, it is added to the previous page.

Under *Default Configuration* you can set the default workflow behavior for each content related application on the portal. You can choose to use no workflow, which is the default, or select any installed workflow definition. Setting the default configuration will cause any newly created Communities or Organizations to default to that configuration. An Administrator can then edit the definitions for each one individually through the *Workflow Configuration* page.

Clicking on *Submissions* will let you view any currently pending assets, or any assets which were previously approved.

Workflow Configuration

After you have uploaded workflow definitions and set the default workflow behavior you can go up to *Workflow Configuration* and tweak the definitions that you are using for each Community and Organization individually.

The screenshot shows a 'Workflow Configuration' page with a table. The columns are 'Resource' and 'Workflow'. The resources listed are 'Blogs Entry', 'Comments', 'Message Boards Message', 'Web Content', 'Document Library Document', and 'Wiki Page'. The 'Workflow' column contains dropdown menus for each resource, with options like 'Default: No workflow' or 'Single Approver (Version 1)'. A note at the bottom says 'Showing 6 results.' and there is a 'Save' button.

Resource	Workflow
Blogs Entry	Default: No workflow
Comments	Default: No workflow
Message Boards Message	Default: No workflow
Web Content	Single Approver (Version 1)
Document Library Document	Single Approver (Version 1)
Wiki Page	Default: No workflow

Showing 6 results.
Save

Illustration 107: The Workflow Configuration page

Using the drop down feature on the Control Panel section, you can select any community or organization in the portal. All of the options under that heading, including Workflow Configuration, now apply to that particular group.

My Workflow Tasks

My Workflow Tasks is a personalized version of the Workflow Tasks, and it is found in your personal section of the Control Panel. Here are specific tasks which have been assigned to you or assigned to a role of which you are a member. You can also view your completed tasks.

The screenshot shows a 'My Workflow Tasks' page. At the top, there are tabs for 'Pending' (which is selected) and 'Completed'. Below the tabs is a search bar with a placeholder 'Search' and an 'Advanced' link. There are two main sections: 'Assigned to Me' and 'Assigned to My Roles'. Under 'Assigned to Me', a message says 'There are no pending tasks assigned to you.' Under 'Assigned to My Roles', another message says 'There are no pending tasks assigned to your roles.'

Pending Completed
Search Advanced
Assigned to Me
There are no pending tasks assigned to you.
Assigned to My Roles
There are no pending tasks assigned to your roles.

Illustration 108: My Workflow Tasks page

It is here that workflow users review and approve content. By clicking on the actions next to a piece of content, a user can view that content and then choose to approve or reject it and add comments.

My Submissions

My Submissions is found under your user's personal information in the Control Panel. From this screen you can view any assets that you have submitted to review. Those that are currently under review are listed under the *Pending* tab, and those that have gone through the review process are listed under the *Completed* tab.

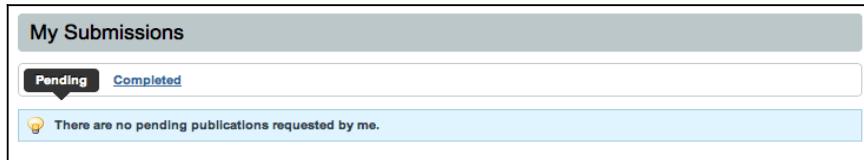


Illustration 109: The My Submissions page

Besides viewing your work, you can also withdraw a submission from the review process by clicking on *Withdraw Submission* from the *Pending* tab.

Integrating with Users, Communities, Organizations and Roles

The Kaleo workflow engine is deeply integrated with Liferay Portal. It can generate roles scoped for Organizations, Communities, and for the whole Portal based on workflow definitions. You can also customize workflow options based on individual communities and organizations,

Users and Roles

Users are the most important part of the workflow, since they're the ones who do all the work. To make a user a part of the workflow process, you assign them a role which you defined in your workflow . When you are creating your workflow definition, you can create new roles by defining them in the XML file, or by using roles which you have already created in your portal. Roles created automatically are always portal scoped, so if you want to use Community or Organization scoped roles, create the roles before deploying your workflow to the portal.

Communities and Organizations

A Portal Administrator can create a default workflow definition scheme for each application which will apply for the entire portal, and a Community or Organization Administrator can customize the settings for their community or organization.

Using Kaleo Workflow Processes in Liferay Portal

Before workflow can be used, you must define which types of assets on the portal are workflow-enabled. If you have created additional definitions, you must also define the workflow definition to use for each asset that is workflow-enabled.

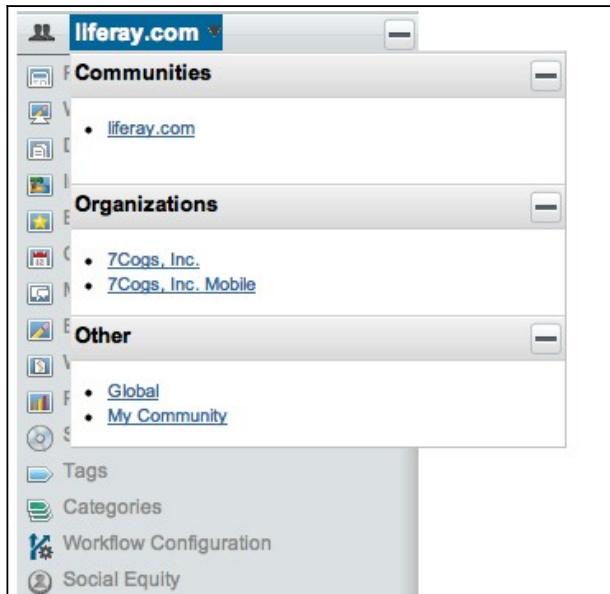


Illustration 110: You can select which community or organization you are currently working in by using the drop down menu over the Content section.

To demonstrate how this works when you configure it, we will create a press release. Press releases should be posted in the *Newsroom* section of the website, so before changing setting specific workflow configuration options or creating content, create the *Newsroom* community and switch to it in the Control Panel.

After going to Workflow Configuration, set Web Content to use the Single Approver workflow.

To demonstrate the workflow process, create two users – a Content Creator and a Content Reviewer. The Content Creator logs in and creates a new Press Release for Spartan Software and clicks *Submit for Publication*. This triggers the workflow process and notifies the Content Reviewer. When the Content Reviewer logs in, he can assign the workflow task to himself and approve the content.

The screenshot shows the 'My Workflow Tasks' portlet. At the top, there are tabs for 'Pending' (which is selected) and 'Completed'. Below the tabs is a search bar with a 'Search' button. Under the search bar is a link 'Advanced »'. The main area is titled 'Assigned to Me' and contains a message: 'There are no pending tasks assigned to you.' Below this, another section is titled 'Assigned to My Roles' and lists one task: 'Review' (Asset Title: 'Press Release', Asset Type: 'Web Content', Last Activity Date: '8/11/10 1:45 AM'). To the right of this list is a context menu with options: 'Assign to Me' (highlighted), 'Assign to...', 'Update Due Date', and 'Actions'.

Illustration 111: Before a Content Reviewer can approve content, he must assign it to himself, or have an administrator assign it to them.

Once the content is approved, it can be posted on the Press Releases page in a web content display portlet.

As you can see, Liferay Portal and the Kaleo Workflow engine combine to create a very robust environment for web content management. Simple workflows can be managed using the default configuration and GUI tools, while more complex workflows can be created to meet the workflow management needs of almost any portal.

Performance Tuning

Once you have your portal up and running, you may find a need to tune it for performance, especially if your site winds up generating more traffic than you'd anticipated. There are some definite steps you can take with regard to improving Liferay's performance.

Memory

Memory is one of the first things to look at when you want to optimize performance. If you have any disk swapping, that will have a serious impact on performance. Make sure that your server has an optimal amount of memory and that your JVM is tuned to use it.

There are three basic JVM command switches that control the amount of memory in the Java heap.

```
-Xms  
-Xmx  
-XX:MaxPermSize
```

These three settings control the amount of memory available to the JVM initially, the maximum amount of memory into which the JVM can grow, and the separate area of the heap called Permanent Generation space.

The first two settings should be set to the same value. This prevents the JVM from having to reallocate memory if the application needs more. Setting them to the same value causes the JVM to be created up front with the maximum amount of memory you want to give it.

```
-Xms1024m -Xmx1024m -XX:MaxPermSize=128m
```

This is perfectly reasonable for a moderately sized machine or a developer machine. These settings give the JVM 1024MB for its regular heap size and have a PermGen space of 128MB. If, however, you have Liferay on a server with 4GB of RAM and you are having performance problems, the first thing you might want to look at is increasing the memory available to the JVM. You will be able to tell if memory is a problem by running a profiler (such as Jprobe, YourKit, or the NetBeans profiler) on the server. If you see Garbage Collection (GC) running frequently, you will definitely want to increase the amount of memory available to the JVM.

Note that there is a law of diminishing returns on memory, especially with 64 bit systems. These systems allow you to create very large JVMs, but the larger the JVM, the more time it takes for garbage collection to take place. For this reason, you probably won't want to create JVMs of more than 2 GB in size. To take advantage of higher amounts of memory on a single system, run multiple JVMs of Liferay instead.

Issues with PermGen space can also affect performance. PermGen space contains long-lived classes, anonymous classes and interned Strings. Hibernate, in particular—which Liferay uses extensively—has been known to make use of PermGen space. If you increase the amount of memory available to the JVM, you may want to increase the amount of PermGen space accordingly.

Garbage Collection

As the system runs, various Java objects are created. Some of these objects are long-lived, and some are not. The ones that are not become *de-referenced*, which means that the JVM no longer has a link to them because they have ceased to be useful. These may be variables that were used for methods which have already returned their values, objects retrieved from the database for a user that is no longer logged on, or a host of other things. These objects sit in memory and fill up the heap space until the JVM decides it's time to clean them up.

Normally, when garbage collection (GC) runs, it stops all processing in the JVM while it goes through the heap looking for dead objects. Once it finds them, it frees up the memory they were taking up, and then processing can

continue. If this happens in a server environment, it can slow down the processing of requests, as all processing comes to a halt while GC is happening.

There are some JVM switches that you can enable which can reduce the amount of time processing is halted while garbage collecting happens. These can improve the performance of your Liferay installation if applied properly. As always, you will need to use a profiler to monitor garbage collection during a load test to tune the numbers properly for your server hardware, operating system, and application server.

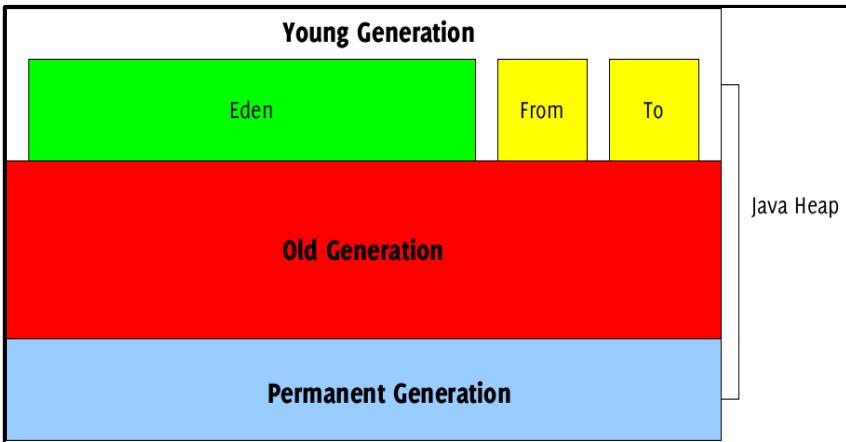


Illustration 112: Java memory

The Java heap is divided into sections for the young generation, the old generation, and the permanent generation. The young generation is further divided into three sections: Eden, which is where new objects are created, and two “survivor spaces,” which we can call the *From* and *To* spaces.

Garbage collection occurs in stages. Generally, it is more frequently done in the young generation, less frequently done in the old generation, and even less frequently done in the permanent generation, where long-lived objects reside. When garbage collection runs in the young generation, Eden is swept for objects which are no longer referenced. Those that are still around are moved to the “To” survivor space, and the “From” space is then swept. Any other objects in that space which still have references to them are moved to the “To” space, and the “From” space is then cleared out altogether. After this, the “From” and the “To” spaces swap roles, and processing is freed up again until the next time the JVM determines that garbage collection needs to run.

After a predetermined number of “generations” of garbage collection, surviving objects may be moved to the old generation. Similarly, after a predetermined number of “generations” of garbage collection in the old generation, surviving objects may be moved to the permanent generation.

By default, the JDK uses a serial garbage collector to achieve this. This works very well for a short-lived desktop Java application, but is not necessarily the best performer for a server-based application like Liferay. For this reason, you may wish to switch to the Concurrent Mark-Sweep (CMS) collector.

Rather than halting application processing altogether, this garbage collector makes one short pause in application execution to mark objects directly reachable from the application code. Then it allows the application to run while it marks all objects which are reachable from the set it marked. Finally, it adds another phase called the *remark* phase which finalizes marking by revisiting any objects modified while the application was running. It then sweeps through and garbage collects. This has the effect of greatly reducing the amount of time that execution needs to be halted in order to clean out dead objects.

Just about every aspect of the way memory management works in Java can be tuned. In your profiling, you may want to experiment with some of the following settings to see if any of them can increase your performance.

NewSize, MaxNewSize: The initial size and the maximum size of the New or Young Generation.

+UseParNewGC: Causes garbage collection to happen in parallel, using multiple CPUs. This decreases garbage collection overhead and increases application throughput.

+UseConcMarkSweepGC: Use the Concurrent Mark-Sweep Garbage Collector. This uses shorter garbage collection pauses, and is good for applications that have a relatively large set of long-lived data, and that run on machines with two or more processors, such as web servers.

+CMSParallelRemarkEnabled: For the CMS GC, enables the garbage collector to use multiple threads during the CMS remark phase. This decreases the pauses during this phase.

SurvivorRatio: Controls the size of the two survivor spaces. It's a ratio between the survivor space size and Eden. The default is 25. There's not much bang for the buck here, but it may need to be adjusted.

ParallelGCThreads: The number of threads to use for parallel garbage collection. Should be equal to the number of CPU cores in your server.

A sample configuration using the above parameters might look something like this:

```
JAVA_OPTS="$JAVA_OPTS -XX:NewSize=700m -XX:MaxNewSize=700m -Xms2048m  
-Xmx2048m -XX:MaxPermSize=128m -XX:+UseParNewGC -XX:+UseConcMarkSweepGC -XX:  
+CMSParallelRemarkEnabled -XX:SurvivorRatio=20 -XX:ParallelGCThreads=8"
```

Properties File Changes

There are also some changes you can make to your *portal-ext.properties* file once you are in a production environment.

Set the following to false to disable checking the last modified date on server side CSS and JavaScript.

```
last.modified.check=false
```

Set this property to true to load the theme's merged CSS files for faster loading for production. By default it is set to false for easier debugging for development. You can also disable fast loading by setting the URL parameter *css_fast_load* to 0.

```
theme.css.fast.load=true
```

Set this property to true to load the combined JavaScript files from the property *javascript.files* into one compacted file for faster loading for production. By default it is set to false for easier debugging for development. You can also disable fast loading by setting the URL parameter *js_fast_load* to 0.

```
javascript.fast.load=true
```

Servlet Filters

Liferay comes by default with 17 servlet filters enabled and running. It is likely that for your installation, you don't need them all.

To disable a servlet filter, simply comment it out of your *web.xml* file.

If there is a feature supported by a servlet filter that you know you are not using, you can comment it out as well to achieve some performance gains. For example, if you are not using CAS for single sign-on, comment out the CAS Filter. If you are not using NTLM for single sign-ons, comment out the Ntlm Filter. The fewer servlet filters you are running, the less processing power is needed for each request.

Portlets

Liferay comes pre-bundled with many portlets which contain a lot of functionality, but not every web site that is running on Liferay needs to use them all. In *portlet.xml* and *liferay-portlet.xml*, comment out the ones you are not using. While having a loan calculator, analog clock, or game of hangman available for your users to add to pages is nice, those portlets may be taking up resources that are needed by custom portlets you have written for your site. If you are having performance problems, commenting out some of the unused portlets may give you the performance boost you need.

Read-Writer Database Configuration

Liferay allows you to use two different data sources for reading and writing. This enables you to split your database infrastructure into two sets: one that is optimized for reading and one that is optimized for writing. Since all major databases support replication in one form or another, you can then use your database vendor's replication mechanism to keep the databases in sync in a much faster manner than if you had a single data source which handled everything.

Enabling a read-writer database is simple. In your *portal-ext.properties* file, configure two different data sources for Liferay to use, one for reading, and one for writing:

```
jdbc.read.driverClassName=com.mysql.jdbc.Driver  
jdbc.read.url=jdbc:mysql://dbread.com/lportal?useUnicode=true& \  
    characterEncoding=UTF-8&useFastDateParsing=false  
jdbc.read.username=  
jdbc.read.password=  
  
jdbc.write.driverClassName=com.mysql.jdbc.Driver  
jdbc.write.url=jdbc:mysql://dbwrite.com/lportal?useUnicode=true& \  
    characterEncoding=UTF-8&useFastDateParsing=false  
jdbc.write.username=  
jdbc.write.password=
```

Of course, specify the user name and password to your database in the above configuration.

After this, enable the read-writer database configuration by uncommenting the Spring configuration file which enables it in your *spring.configs* property (line to uncomment is in bold):

```
spring.configs=\  
    META-INF/base-spring.xml,\ \  
    \  
    META-INF/hibernate-spring.xml,\ \  
    META-INF/infrastructure-spring.xml,\ \  
    META-INF/management-spring.xml,\ \  
    \  
    META-INF/util-spring.xml,\ \  
    \  
    META-INF/editor-spring.xml,\ \  
    META-INF/jcr-spring.xml,\ \  
    META-INF/messaging-spring.xml,\ \  
    META-INF/scheduler-spring.xml,\ \  
    META-INF/search-spring.xml,\ \  
    \  
    META-INF/counter-spring.xml,\
```

```
META-INF/document-library-spring.xml,\  
META-INF/lock-spring.xml,\  
META-INF/mail-spring.xml,\  
META-INF/portal-spring.xml,\  
META-INF/portlet-container-spring.xml,\  
META-INF/wsrf-spring.xml,\  
\  
META-INF/mirage-spring.xml,\  
\  
META-INF/dynamic-data-source-spring.xml,\  
#META-INF/shard-data-source-spring.xml,\  
\  
META-INF/ext-spring.xml
```

The next time you restart Liferay, it will now use the two data sources you have defined. Be sure to make sure that you have correctly set up your two databases for replication before starting Liferay.

Database Sharding

Liferay starting with version 5.2.3 supports database sharding for different portal instances. Sharding is a term used to describe an extremely high scalability configuration for systems with massive amounts of users. In diagrams, a database is normally pictured as a cylinder. Instead, picture it as a glass bottle full of data. Now take that bottle and smash it onto a concrete sidewalk. There will be shards of glass everywhere. If that bottle were a database, each shard now is a database, with a subset of the data in each shard.

This allows you to split up your database by various types of data that might be in it. For example, some implementations of sharding a database split up the users: those with last names beginning with A to D go in one database; E to I go in another; etc. When users log in, they are directed to the instance of the application that is connected to the database that corresponds to their last names. In this manner, processing is split up evenly, and the amount of data the application needs to sort through is reduced.

By default, Liferay allows you to support sharding through different portal instances, using the *round robin shard selector*. This is a class which serves as the default algorithm for sharding in Liferay. Using this algorithm, Liferay will select from several different portal instances and evenly distribute the data across them.

Of course, if you wish to have your developers implement your own sharding algorithm, you can do that. You can select which algorithm is active via the *portal-ext.properties* file:

```
shard.selector=com.liferay.portal.dao.shard.RoundRobinShardSelector  
#shard.selector=com.liferay.portal.dao.shard.ManualShardSelector
```

```
#shard.selector=[your implementation here]
```

Enabling sharding is easy. You will need to make sure you are using Liferay's data source implementation instead of your application server's. Set your various database shards in your *portal-ext.properties* file this way:

```
jdbc.default.driverClassName=com.mysql.jdbc.Driver
jdbc.default.url=jdbc:mysql://localhost/lportal?
useUnicode=true&characterEncoding=UTF-8&useFastDateParsing=false
jdbc.default.username=
jdbc.default.password=
jdbc.one.driverClassName=com.mysql.jdbc.Driver
jdbc.one.url=jdbc:mysql://localhost/lportal1?
useUnicode=true&characterEncoding=UTF-8&useFastDateParsing=false
jdbc.one.username=
jdbc.one.password=
jdbc.two.driverClassName=com.mysql.jdbc.Driver
jdbc.two.url=jdbc:mysql://localhost/lportal2?
useUnicode=true&characterEncoding=UTF-8&useFastDateParsing=false
jdbc.two.username=
jdbc.two.password=
shard.available.names=default,one,two
```

Once you do this, you can set up your DNS so that several domain names point to your Liferay installation (e.g., abc1.com, abc2.com, abc3.com). Next, go to the Control Panel and click *Portal Instances* in the Server category. Create two to three instances bound to the DNS names you have configured.

If you are using the RoundRobinShardSelector class, Liferay will automatically enter data into each instance one by one, automatically. If you are using the ManualShardSelector class, you will have to specify a shard for each instance using the UI.

The last thing you will need to do is modify the *spring.configs* section of your *portal-ext.properties* file to enable the sharding configuration, which by default is commented out. To do this, your *spring.configs* should look like this (modified section is in bold):

```
spring.configs=\
META-INF/base-spring.xml, \
\
META-INF/hibernate-spring.xml, \
META-INF/infrastructure-spring.xml, \
META-INF/management-spring.xml, \
\
META-INF/util-spring.xml, \
\
META-INF/editor-spring.xml, \
META-INF/jcr-spring.xml, \
META-INF/messaging-spring.xml, \
META-INF/scheduler-spring.xml,
```

```
META-INF/search-spring.xml,\  
 \  
META-INF/counter-spring.xml,\  
META-INF/document-library-spring.xml,\  
META-INF/lock-spring.xml,\  
META-INF/mail-spring.xml,\  
META-INF/portal-spring.xml,\  
META-INF/portlet-container-spring.xml,\  
META-INF/wsdp-spring.xml,\  
 \  
META-INF/mirage-spring.xml,\  
 \  
#META-INF/dynamic-data-source-spring.xml,\  
META-INF/shard-data-source-spring.xml,\  
 \  
 \
```

That's all there is to it. Your system is now set up for sharding.

Summary

We've seen how good a fit Liferay Portal is for the enterprise. It can be scaled linearly to grow to whatever size you need to serve your users. Clustering is a snap, and Liferay harmonizes very well with whatever environment you may have.

Kaleo workflow is a simple, yet robust workflow solution for your enterprise. With deep integration with Liferay's portlet applications and permissions system, it is an ideal choice for implementing your business processes.

Liferay Portal is also built for performance. You can tune it to support over 3300 concurrent users on a single server with mean log in times under half a second and maximum throughput of more than 79 log ins per second. We've seen some tips for tuning Liferay Portal, and we have to keep in mind the adage about tuning: load test and profile, tune, repeat.

You can also take advantage of read-writer database configurations, as well as database sharding. In all, Liferay Portal gives you all the options you need to build a high-performance, robust environment that supports your enterprise.

8. MAINTAINING A LIFERAY PORTAL

Maintaining a running implementation of Liferay Portal is not much different from maintaining the application server environment upon which it is running. There are, however, several factors which administrators should be aware of when they are responsible for a running instance of Liferay. This chapter will cover these issues, outlining for system administrators some specifics about keeping a running Liferay instance stable and secure.

This chapter will cover the following topics:

- Liferay Monitoring using Google Analytics
- Backing Up a Liferay Installation
- Changing Logging Levels
- Upgrading Liferay

The discussion on back up will cover what parts of Liferay should be backed up. We will not cover specific backup software or procedures; generally, most organizations have standards for doing backups of their systems, and Liferay as a Java EE application fits well into these standards.

Liferay Monitoring Using Google Analytics

Liferay includes built-in support for Google Analytics, allowing administrators to make use of Google's tool set for analyzing site traffic data. When you sign up for Google Analytics, a snippet of code is provided which needs to

be added to your web pages in order to allow Google's system to register the page hit. It can be a tedious process to add this code to every page on a site, especially if it is a large site and there is a lot of user-generated content.

This problem can be solved in Liferay by putting Google's code into a custom theme written especially for the site on which the portal is running. Doing this, however, requires that a theme developer make specific changes to the theme, and it prevents users from using the many freely available themes that are available for Liferay "out of the box."

Because of this, support for Google Analytics has been built into Liferay, and can be turned on through a simple user interface. This allows Liferay Administrators to make use of Google Analytics on a community by community basis and turn it on and off when needed.

To enable Google Analytics support, go to the Manage Pages screen for the community for which you want to enable support. You can do this through the Control Panel by going to either the *Organizations* or *Communities* link in the *Portal* section, and then clicking *Actions* → *Manage Pages* for the community or organization you want to analyze. Click the *Settings* tab.

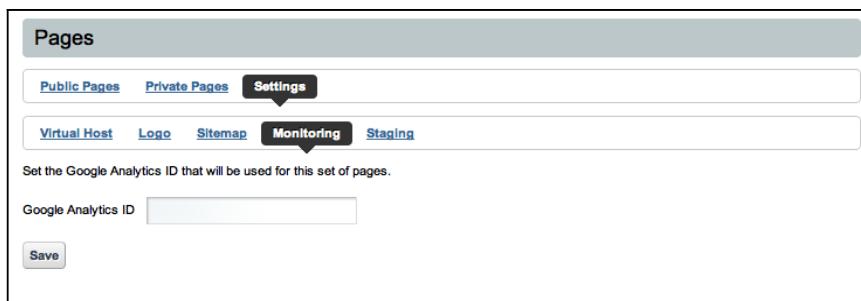


Illustration 113: Setting Up Google Analytics

Click the **Monitoring** Tab. Put your Google Analytics ID (which should have been provided to you when you signed up for the service) in the field and click *Save*. All of the pages in the community you selected will now have the Google Analytics code in them and will be tracked.

Backing Up A Liferay Installation

Once you have an installation of Liferay Portal running, you will want to have proper backup procedures in place in case of a catastrophic failure of some kind. Liferay is not very different from any other application that may be running in your application server, but there are some specific components that need to be backed up in addition to your regular backup procedures for your application server.

Source Code

If you have extended Liferay or have written any plugins, they should be stored in a source code repository such as Subversion, CVS, or Git. This repository should be backed up on a regular basis to preserve your ongoing work.

If you are extending Liferay with the Ext Plugin, you will want to make sure that you also store the version of the Liferay source on which your extension environment is based. This allows your developers convenient access to all of the tools they need to build your extension and deploy it to a server.

Liferay's File System

Liferay's configuration file, `portal-ext.properties`, gets stored in the *Liferay Home* folder, which is generally one folder up from where your application server is installed (see Chapter 2 for specific info for your application server). At a minimum, this file should be backed up, but it is generally best to back up your whole application server.

If you have followed the procedure in the previous chapter to modify your Ehcache configuration, you will have cache configuration files in the deploy location of Liferay. You will need to back up this location.

Liferay also stores configuration files, search indexes, cache information, and the default Jackrabbit document repository in a folder called `data` in Liferay Home. You should generally back up all of the contents of your Liferay Home folder.

If you have modified the location where the Document Library stores files, you should also back up this location.

Database

Liferay's database is the central repository for all of the Portal's information and is the most important component which needs to be backed up. You can do this by either backing up the database live (if your database allows this) or by exporting the database and then backing up the exported file. For example, MySQL ships with a `mysqldump` utility which allows you to export the entire database and data into a large SQL file. This file can then be backed up. In case of a database failure, it can be used to recreate the state of the database at the time the dump was created.

If you are using Liferay's Document Library with the Jackrabbit JSR-170 repository to store documents in a database, the Jackrabbit database should be backed up also.

Liferay's Logging System

Liferay uses Log4j extensively to implement logging for nearly every class in the portal. If you need to debug something specific while a system is running, you can use the Control Panel to set logging levels by class dynamically.

To view the log levels, go to the Control Panel, click *Server Administration* in the Server section, and then click the *Log Levels* tab.

You will then see a paginated list of logging categories. These categories correspond to Liferay classes that have log messages in them. By default, all categories are set to display messages only if there is an error that occurs in the class. This is why you see ERROR displayed in all of the drop down list boxes on the right side of the portlet.

Each category is filtered by its place in the class hierarchy. For example, if you wanted to see logging for a specific class that is registered in Liferay, you would browse to that specific class and change its log level to something that is more descriptive, such as DEBUG. Once you click the *Save* button at the bottom of the list, you will start seeing DEBUG messages from that class in your application server's log file.

If you are not sure which class you want to see log messages for, you can find a place higher up in the hierarchy and select the package name instead of an individual class name. If you do this, messages for every class lower in the hierarchy will be displayed in your application server's log file.

Category	Level
com.ecyrd.jspwiki	ERROR
com.germinus.easyconf	ERROR
com.liferay	ERROR
com.liferay.documentlibrary	ERROR
com.liferay.documentlibrary.util	ERROR
com.liferay.documentlibrary.util.CMISHook	INFO
com.liferay.documentlibrary.util.DLIndexer	ERROR
com.liferay.documentlibrary.util.HookFactory	ERROR
com.liferay.jdbc	ERROR
com.liferay.mail.service.impl.MailServiceImpl	ERROR

Illustration 114: Changing Logging Levels

Be careful when you do this. If you set the log level to DEBUG somewhere near the top of the hierarchy (such as `com.liferay`, for example), you may wind up with a lot of messages in your log file. This could make it difficult to find the one you were looking for, and causes the server to do more work writing messages to its log file.

If you are want to set the log level for one of your own classes in a deployed plugin, you can register that class (so long as it uses Log4J to do its logging) with Liferay so that you can control the log levels more easily.

You will first need to implement Log4J logging in your class, with a statement such as the following (taken from Liferay's `JCRHook` class):

```
private static Log _log = LogFactory.getLog(JCRHook.class);
```

You would then use this `_log` variable to create log messages in your code for the various logging levels:

```
_log.error("Reindexing " + node.getName(), e1);
```

To enable your logging messages to appear in your server's log file via the Control Panel, click the *Add Category* tab on the same *Log Levels* page.

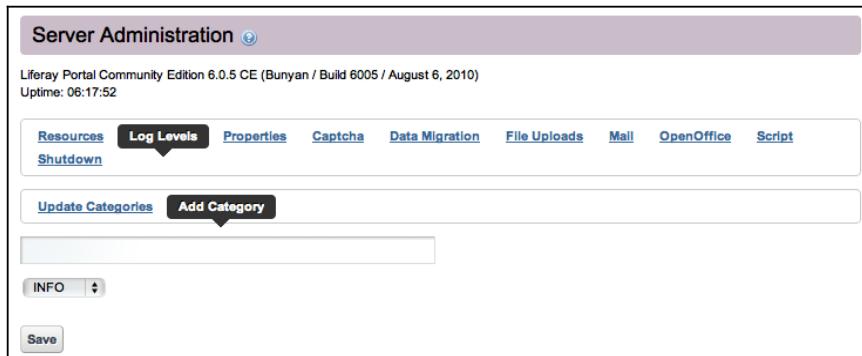


Illustration 115: Adding a Logging Category

You will see that you can add a logging category. Simply put in the fully qualified name of your class or of the package that contains the classes whose log messages you want to view, choose a log level, and then click the *Save* button. You will now start to see log messages from your own class or classes in the server's log file.

Upgrading Liferay

Liferay upgrades are fairly straightforward. A consistent set of steps is all you need to follow to upgrade a standard Liferay installation. Things do get more complicated if your organization has used the extension environment to customize Liferay, as it will need to be converted to an Ext Plugin for Liferay 6, and it is possible that API changes in the new version will break your existing code. This, however, is usually pretty easy for your developers to fix. Portlet plugins are generally backwards compatible, as they are written to the Java standard. This includes Portlet 1.0 (JSR-168) portlets, as the Portlet 2.0 (JSR-286) standard has also been designed to be backwards-compatible. Theme plugins may require some modifications in order to take advantage of new features. Much effort has been made to make upgrades as painless as possible; however, this is not a guarantee that everything will work without modification. Extension environment code is the most complicating factor in an upgrade, so it is important to test as much as possible.

As a general rule, you can upgrade from one major release to the next major release. For example, you can upgrade directly from Liferay 5.1.x to 5.2.x, but not from 5.1.x to 6.0.x. If you need to upgrade over several major releases, you will need to run the upgrade procedure for each major release until you reach the release you want. This doesn't mean you need to run the procedure for every point release (i.e., 4.3.5 to 4.3.6 to 4.4.0 to 4.4.1, etc.); you only need to run the procedure for the major releases. A good practice is to use the latest version of each major release to upgrade your system.

Liferay Upgrade Procedure

Liferay can auto-detect whether the database requires an upgrade the first time the new version is started. When Liferay does this, it will automatically upgrade the database to the format required by the new version. In order to do this, Liferay *must* be accessing the database with an ID that can create, drop, and modify tables. Make sure that you have granted these permissions to the ID before you attempt to upgrade Liferay. It is also a good idea to backup your database before attempting an upgrade in case something goes wrong during the process.



Tip: Liferay versions prior to 4.3.0 require that you manually run SQL scripts on your database to perform an upgrade. If you need to upgrade from Liferay 4.1.x to 4.2.x in preparation for an upgrade to a current version of Liferay, you can find these SQL scripts in the source code archive for the version of Liferay you are running. They will be in the *SQL* folder of the archive.

Upgrade Steps

It takes only five steps to upgrade a standard Liferay installation to Liferay 6:

1. Copy your customized `portal-ext.properties` file to a safe place, and then undeploy the old version of Liferay and shut down your application server.
2. Copy the new versions of the dependency `.jars` to a location on your server's class path, overwriting the ones you already have for the old version of Liferay.
3. Deploy the new Liferay `.war` file to your application server. Follow the deployment instructions in Chapter 2 .
4. Modify your `portal-ext.properties` file and set `permissions.user.check.algorithm=5`.
5. Start (or restart) your application server. Watch the console as Liferay starts: it should upgrade the database automatically. Verify that your portal is operating normally, and then run the upgrade procedure to upgrade to permissions algorithm 6 (see below) and restart.

That's all there is to it. Everything else is handled by Liferay's upgrade procedure. Note that as stated above, if you have to upgrade over several Liferay versions, you will need to repeat these steps for each major release. You can now deploy your own plugins to the system.

Once your upgrade is complete, you may wish to review the portal.-properties changes for this version of Liferay to see whether the new defaults (see below) are appropriate for your implementation.

What follows are instructions for upgrading for specific versions.

Upgrading From Liferay 5.1 to Liferay 5.2

Always use the latest version of 5.2 available as it will have fixed all the potential upgrade issues that may have been found.

Prerequisite

It's recommended to upgrade first at least to 5.1.2SE if you are running any previous version.

Changes in configuration properties

How to keep the old values

The default values of some properties has been changed. In order to keep the previous values you have to run Liferay passing the following system property:

```
java ... -Dexternal-properties=portal-legacy-5.1.properties
```

Each application server has different methods to add this system property. In Tomcat modify setenv.sh/setenv.bat and append that option to the environment variable JAVA_OPTS. The scripts setenv.sh or setenv.bat are not delivered with Tomcat but if they exist, Tomcat will use them in the startup process, so it's a nice way to separate your own settings from tomcat's default shell scripts.

Here are the complete contents of that file (portal-legacy-5.1.properties) for reference:

```
resource.repositories.root=${user.home}/liferay

theme.portlet.sharing.default=true

organizations.country.required[regular]=true
organizations.assignment.auto=true
organizations.assignment.strict=false
organizations.membership.strict=true

lucene.dir=${resource.repositories.root}/lucene/

jcr.jackrabbit.repository.root=${resource.repositories.root}/jackrabbit
```

```
dl.hook.impl=com.liferay.documentlibrary.util.JCRHook
dl.hook.file.system.root.dir=${resource.repositories.root}/document_library
```

Important changes in the configuration of Database access and mail integration

One very important aspect of the upgrade is that now the configuration of the database parameters and those for mail integration are handled through the portal-ext.properties file to unify the configuration through all application servers.

It's still possible to use application server specific data sources and pools if desired by using certain configuration properties. This is documented in Chapter 2.

Theme Upgrade

Instructions for maintaining customized themes built in 5.1 without redeploying with the new SDK :

- Change the header of /WEB-INF/liferay-plugin-package.xml to:

```
<!DOCTYPE plugin-package PUBLIC "-//Liferay//DTD Plugin Package 5.2.0//EN"
"http://www.liferay.com/dtd/liferay-plugin-package_5_2_0.dtd">
```

- Change the header of /WEB-INF/liferay-look-and-feel.xml to:

```
<!DOCTYPE look-and-feel PUBLIC "-//Liferay//DTD Look and Feel 5.2.0//EN"
"http://www.liferay.com/dtd/liferay-look-and-feel_5_2_0.dtd">]
```

- Upgrade compatibility version in liferay-look-and-feel.xml:

```
<compatibility>
    <version>5.2.2+</version>
</compatibility>
```

- In portal.vm, delete the following lines :

```
$theme.include($bottom_ext_include)
$theme.include($session_timeout_include)
$theme.include($sound_alerts_include)
```

If you don't remove these, you will see a blank page and an exception.

- In order to display the control panel in the dock, add the following lines in dock.vm:

```
#if ($show_control_panel)
<li class="control-panel">
<a href="$control_panel_url">$control_panel_text</a>
</li>
#end
```

- In navigation.css:

```
.lfr-dock li.control-panel a {  
    background-image: url(..../images/dock/control_panel.png);  
}
```

- Then copy /images/dock/control_panel.png from the classic theme (ROOT/html/themes/classic) into your theme.
- In WEB-INF/web.xml, change the deprecated declaration com.liferay.portal.servlet.filters.compression.CompressionFilter into com.liferay.portal.servlet.filters.gzip.GZipFilter.

API Changes

Usage of ServiceContext in Liferay's Services Layer

The most significant API change in 5.2 is that most APIs of the service layer have been adapted to use the Service Context Pattern. The Service Context is an object that contains context information about a given API call. All of the fields in this object are optional, although the services that store any type of content will require you to specify at least the scopeGroupId. Here is a simple example of how to create a ServiceContext instance and pass it to a service API:

```
ServiceContext serviceContext = new ServiceContext();  
serviceContext.setScopeGroupId(myGroupId);  
  
BlogsEntryServiceUtil.addEntry(..., serviceContext);
```

If you are invoking the service from a servlet, a Struts action, or any other front end class which has access to the portletRequest, you can use a utility method that will create the ServiceContext object and fill it with all the necessary values automatically. In that case the above example should be re-written as follows:

```
ServiceContext serviceContext =  
ServiceContextFactory.getInstance(BlogsEntry.class.getName(),  
portletRequest);  
  
BlogsEntryServiceUtil.addEntry(..., serviceContext);
```

Upgrading From Liferay 5.2 to Liferay 6.0

Always use the latest version of 6.0 available as it will have fixed all the potential upgrade issues that may have been found.

Prerequisite

It's recommended to upgrade first at least to 5.2.3 CE if you are running any previous version.

Upgrading Your Permissions Algorithm

Liferay 6.0 introduces permissions algorithm 6. Algorithm 6 is an enhancement to our permissions system which drastically improves performance by reducing the number of queries necessary to determine permissions for any individual asset within Liferay portal.

Liferay 6 by default is configured to use algorithm 6. This is appropriate for new installations, but requires additional configuration for upgrades, because the table structure for this algorithm is different from the structures for the previous algorithms. For this reason, before you start Liferay 6 for the first time and run the upgrade process, you need to tell Liferay 6 to use Algorithm 5, and then run a separate conversion utility later, after you have verified that the upgrade was successful. To temporarily switch your Liferay 6 installation to algorithm 5, add the following entry to your `portal-ext.properties`:

```
permissions.user.check.algorithm=5
```

This will tell Liferay that you are still using algorithm 5. Next, start Liferay and allow it to upgrade your database. Once the upgrade has completed, verify that everything is working properly. You can now leave the algorithm setting as is, or if you want to take advantage of the performance of the new algorithm, you can upgrade to algorithm 6 by going through a few simple steps in the Control Panel.

To upgrade to Algorithm 6, log in as an Administrator and navigate to the Control Panel. Go to *Server Administration* and select *Data Migration* from the menu along the top of the screen. You should see a section entitled *Legacy Permissions Migration* at the bottom of the page.



Illustration 116: Dialog to update your permissions algorithm to the latest version.

Algorithms 5 and 6 do not support adding permissions at the user level. If you have permissions set to individual users, the converter can simulate this for you by auto-generating roles with those permissions, and assigning those roles to the users who have individualized permissions. If you have a lot of these, you'll likely want to go through and clean them up after the conversion process. To generate these roles, check the *Generate Custom Roles* box.

If you do not generate the roles, all custom permissions set for individual users will be discarded.

Click *Execute* to convert all existing users and roles to algorithm 6.

Upgrading EXT to EXT Plugins

With Liferay 6.0, the Ext environment no longer exists in its previous form. Instead, Ext is now a plugin. If you are using the Ext Environment to change core code, you will find that Ext Plugins handle core customizations better than the Ext Environment did, so developers can spend less time with deployments and maintaining changes than they did previously. There is now only one SDK needed for Liferay development, which also simplifies things for the developer.

Before using any code previously developed in Ext, you will need to migrate your Ext environments to Ext plugins. If you are on a version of Liferay prior to 5.2, you will also need to upgrade your existing Ext environment to Liferay 5.2 before attempting to convert it to a plugin. The upgrade process will not work on earlier versions of Liferay's Ext environment.

In the Plugins SDK, under the ext directory you can find the build.xml file to convert your existing Ext Environment to a plugin. The script creates the shell of an Ext plugin with the ext.zip file included with the Plugins SDK and merges your existing Ext Environment into the new plugin.

To run the script, use the following command:

```
ant upgrade-ext -Dext.dir=[path to existing Ext] -Dext.name=[new plugin name] -Dext.display.name="[friendly name for new plugin]"
```

This will create a directory with the name you specified, with the merged changes from your Ext Environment and the default .zip file. Both build-service (Service Builder) and build-db (DB Builder) have been ported to Ext Plugins to allow developers to regenerate their services and SQL code in Ext Plugins. However, Service Builder in Ext plugins will be deprecated in future versions, and custom services should be migrated to portlet plugins. Try to migrate your custom services to portlet plugins as soon as possible as this is the recommended practice.

Summary

Liferay Portal is an easy environment to maintain. Backup procedures are simple and straightforward. Administrators have all the options they need to view and diagnose a running Liferay Portal server through its tunable logs.

Upgrading Liferay is also a snap, because Liferay does most of the work automatically. With easy permissions migration tools and automated data-

base upgrade scripts, you'll have your new version of Liferay Portal up and running in no time.

9. APPENDIX: DOCUMENTATION LICENSE

The text of this book is copyrighted by Liferay, Inc., and is released under the *Creative Commons Attribution-Sharealike 3.0 Unported* license.

Creative Commons License

CREATIVE COMMONS CORPORATION IS NOT A LAW FIRM AND DOES NOT PROVIDE LEGAL SERVICES. DISTRIBUTION OF THIS LICENSE DOES NOT CREATE AN ATTORNEY-CLIENT RELATIONSHIP. CREATIVE COMMONS PROVIDES THIS INFORMATION ON AN "AS-IS" BASIS. CREATIVE COMMONS MAKES NO WARRANTIES REGARDING THE INFORMATION PROVIDED, AND DISCLAIMS LIABILITY FOR DAMAGES RESULTING FROM ITS USE.

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR

GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

- a. **“Adaptation”** means a work based upon the Work, or upon the Work and other pre-existing works, such as a translation, adaptation, derivative work, arrangement of music or other alterations of a literary or artistic work, or phonogram or performance and includes cinematographic adaptations or any other form in which the Work may be recast, transformed, or adapted including in any form recognizably derived from the original, except that a work that constitutes a Collection will not be considered an Adaptation for the purpose of this License. For the avoidance of doubt, where the Work is a musical work, performance or phonogram, the synchronization of the Work in timed-relation with a moving image (“synching”) will be considered an Adaptation for the purpose of this License.
- b. **“Collection”** means a collection of literary or artistic works, such as encyclopedias and anthologies, or performances, phonograms or broadcasts, or other works or subject matter other than works listed in Section 1(f) below, which, by reason of the selection and arrangement of their contents, constitute intellectual creations, in which the Work is included in its entirety in unmodified form along with one or more other contributions, each constituting separate and independent works in themselves, which together are assembled into a collective whole. A work that constitutes a Collection will not be considered an Adaptation (as defined below) for the purposes of this License.
- c. **“Creative Commons Compatible License”** means a license that is listed at <http://creativecommons.org/compatiblelicenses> that has been approved by Creative Commons as being essentially equivalent to this License, including, at a minimum, because that license: (i) contains terms that have the same purpose, meaning and effect as the License Elements of this License; and, (ii) explicitly permits the relicensing of adaptations of works made available under that license under this License or a Creative Commons jurisdiction license with the same License Elements as this License.
- d. **“Distribute”** means to make available to the public the original and copies of the Work or Adaptation, as appropriate, through sale or other transfer of ownership.
- e. **“License Elements”** means the following high-level license attributes as selected by Licensor and indicated in the title of this License: Attribution, ShareAlike.
- f. **“Licensor”** means the individual, individuals, entity or entities that offer(s) the Work under the terms of this License.

- g. "**Original Author**" means, in the case of a literary or artistic work, the individual, individuals, entity or entities who created the Work or if no individual or entity can be identified, the publisher; and in addition (i) in the case of a performance the actors, singers, musicians, dancers, and other persons who act, sing, deliver, declaim, play in, interpret or otherwise perform literary or artistic works or expressions of folklore; (ii) in the case of a phonogram the producer being the person or legal entity who first fixes the sounds of a performance or other sounds; and, (iii) in the case of broadcasts, the organization that transmits the broadcast.
- h. "**Work**" means the literary and/or artistic work offered under the terms of this License including without limitation any production in the literary, scientific and artistic domain, whatever may be the mode or form of its expression including digital form, such as a book, pamphlet and other writing; a lecture, address, sermon or other work of the same nature; a dramatic or dramatico-musical work; a choreographic work or entertainment in dumb show; a musical composition with or without words; a cinematographic work to which are assimilated works expressed by a process analogous to cinematography; a work of drawing, painting, architecture, sculpture, engraving or lithography; a photographic work to which are assimilated works expressed by a process analogous to photography; a work of applied art; an illustration, map, plan, sketch or three-dimensional work relative to geography, topography, architecture or science; a performance; a broadcast; a phonogram; a compilation of data to the extent it is protected as a copyrightable work; or a work performed by a variety or circus performer to the extent it is not otherwise considered a literary or artistic work.
- i. "**You**" means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.
- j. "**Publicly Perform**" means to perform public recitations of the Work and to communicate to the public those public recitations, by any means or process, including by wire or wireless means or public digital performances; to make available to the public Works in such a way that members of the public may access these Works from a place and at a place individually chosen by them; to perform the Work to the public by any means or process and the communication to the public of the performances of the Work, including by public digital performance; to broadcast and rebroadcast the Work by any means including signs, sounds or images.

k. **“Reproduce”** means to make copies of the Work by any means including without limitation by sound or visual recordings and the right of fixation and reproducing fixations of the Work, including storage of a protected performance or phonogram in digital form or other electronic medium.

2. Fair Dealing Rights. Nothing in this License is intended to reduce, limit, or restrict any uses free from copyright or rights arising from limitations or exceptions that are provided for in connection with the copyright protection under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

- a. to Reproduce the Work, to incorporate the Work into one or more Collections, and to Reproduce the Work as incorporated in the Collections;
- b. to create and Reproduce Adaptations provided that any such Adaptation, including any translation in any medium, takes reasonable steps to clearly label, demarcate or otherwise identify that changes were made to the original Work. For example, a translation could be marked “The original work was translated from English to Spanish,” or a modification could indicate “The original work has been modified.”;
- c. to Distribute and Publicly Perform the Work including as incorporated in Collections; and,
- d. to Distribute and Publicly Perform Adaptations.
- e. For the avoidance of doubt:
 - i. **Non-waivable Compulsory License Schemes.** In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme cannot be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License;
 - ii. **Waivable Compulsory License Schemes.** In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme can be waived, the Licensor waives the exclusive right to collect such royalties for any exercise by You of the rights granted under this License; and,
 - iii. **Voluntary License Schemes.** The Licensor waives the right to collect royalties, whether individually or, in the event that the Licensor is a member of a collecting society that ad-

ministers voluntary licensing schemes, via that society, from any exercise by You of the rights granted under this License.

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. Subject to Section 8(f), all rights not expressly granted by Licensor are hereby reserved.

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

- a. You may Distribute or Publicly Perform the Work only under the terms of this License. You must include a copy of, or the Uniform Resource Identifier (URI) for, this License with every copy of the Work You Distribute or Publicly Perform. You may not offer or impose any terms on the Work that restrict the terms of this License or the ability of the recipient of the Work to exercise the rights granted to that recipient under the terms of the License. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties with every copy of the Work You Distribute or Publicly Perform. When You Distribute or Publicly Perform the Work, You may not impose any effective technological measures on the Work that restrict the ability of a recipient of the Work from You to exercise the rights granted to that recipient under the terms of the License. This Section 4(a) applies to the Work as incorporated in a Collection, but this does not require the Collection apart from the Work itself to be made subject to the terms of this License. If You create a Collection, upon notice from any Licensor You must, to the extent practicable, remove from the Collection any credit as required by Section 4(c), as requested. If You create an Adaptation, upon notice from any Licensor You must, to the extent practicable, remove from the Adaptation any credit as required by Section 4(c), as requested.
- b. You may Distribute or Publicly Perform an Adaptation only under the terms of: (i) this License; (ii) a later version of this License with the same License Elements as this License; (iii) a Creative Commons jurisdiction license (either this or a later license version) that contains the same License Elements as this License (e.g., Attribution-ShareAlike 3.0 US)); (iv) a Creative Commons Compatible License. If you license the Adaptation under one of the licenses mentioned in (iv), you must comply with the terms of that license. If you license the Adaptation under the terms of any of the licenses mentioned in (i), (ii) or (iii) (the “Applicable License”), you must comply with the terms of the Applicable License generally and the following

- provisions: (I) You must include a copy of, or the URI for, the Applicable License with every copy of each Adaptation You Distribute or Publicly Perform; (II) You may not offer or impose any terms on the Adaptation that restrict the terms of the Applicable License or the ability of the recipient of the Adaptation to exercise the rights granted to that recipient under the terms of the Applicable License; (III) You must keep intact all notices that refer to the Applicable License and to the disclaimer of warranties with every copy of the Work as included in the Adaptation You Distribute or Publicly Perform; (IV) when You Distribute or Publicly Perform the Adaptation, You may not impose any effective technological measures on the Adaptation that restrict the ability of a recipient of the Adaptation from You to exercise the rights granted to that recipient under the terms of the Applicable License. This Section 4(b) applies to the Adaptation as incorporated in a Collection, but this does not require the Collection apart from the Adaptation itself to be made subject to the terms of the Applicable License.
- c. If You Distribute, or Publicly Perform the Work or any Adaptations or Collections, You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or if the Original Author and/or Licensor designate another party or parties (e.g., a sponsor institute, publishing entity, journal) for attribution (“Attribution Parties”) in Licensor’s copyright notice, terms of service or by other reasonable means, the name of such party or parties; (ii) the title of the Work if supplied; (iii) to the extent reasonably practicable, the URI, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and (iv), consistent with Section 3(b), in the case of an Adaptation, a credit identifying the use of the Work in the Adaptation (e.g., “French translation of the Work by Original Author,” or “Screenplay based on original Work by Original Author”). The credit required by this Section 4(c) may be implemented in any reasonable manner; provided, however, that in the case of a Adaptation or Collection, at a minimum such credit will appear, if a credit for all contributing authors of the Adaptation or Collection appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance of doubt, You may only use the credit required by this Section for the purpose of attribution in the manner set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorse-

ment by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.

- d. Except as otherwise agreed in writing by the Licensor or as may be otherwise permitted by applicable law, if You Reproduce, Distribute or Publicly Perform the Work either by itself or as part of any Adaptations or Collections, You must not distort, mutilate, modify or take other derogatory action in relation to the Work which would be prejudicial to the Original Author's honor or reputation. Licensor agrees that in those jurisdictions (e.g. Japan), in which any exercise of the right granted in Section 3(b) of this License (the right to make Adaptations) would be deemed to be a distortion, mutilation, modification or other derogatory action prejudicial to the Original Author's honor and reputation, the Licensor will waive or not assert, as appropriate, this Section, to the fullest extent permitted by the applicable national law, to enable You to reasonably exercise Your right under Section 3(b) of this License (right to make Adaptations) but not otherwise.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

- a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Adaptations or Collections from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full com-

pliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

- b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

- a. Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.
- b. Each time You Distribute or Publicly Perform an Adaptation, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.
- c. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
- d. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.
- e. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.
- f. The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised

on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.

Creative Commons Notice

Creative Commons is not a party to this License, and makes no warranty whatsoever in connection with the Work. Creative Commons will not be liable to You or any party on any legal theory for any damages whatsoever, including without limitation any general, special, incidental or consequential damages arising in connection to this license. Notwithstanding the foregoing two (2) sentences, if Creative Commons has expressly identified itself as the Licensor hereunder, it shall have all rights and obligations of Licensor.

Except for the limited purpose of indicating to the public that the Work is licensed under the CCPL, Creative Commons does not authorize the use by either party of the trademark "Creative Commons" or any related trademark or logo of Creative Commons without the prior written consent of Creative Commons. Any permitted use will be in compliance with Creative Commons' then-current trademark usage guidelines, as may be published on its website or otherwise made available upon request from time to time. For the avoidance of doubt, this trademark restriction does not form part of the License.

Creative Commons may be contacted at
<http://creativecommons.org/>.

10. COLOPHON

The text and layout were accomplished using OpenOffice.org 3.2 running on Kubuntu Linux. The file is one large document, rather than a multi-linked master document. It's just easier to manage that way. The PDF was exported directly from OpenOffice.org. Some content written for this book was also exported to the Liferay wiki at <http://wiki.liferay.com>. It is the intent of the author to continue doing this for every edition of this book that is published. We also expect that some more community documentation will make it into the official documentation.

All of the fonts used in the creation of this book are either open source fonts or freely available fonts. The body text font is Gentium Book Basic, created by SIL International. It can be found here: http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&item_id=Gentium.

The font for the titles is called Delicious, from the free exljbris font foundry. It can be found here: <http://www.josbuivenga.demon.nl/delicious.html>. This change was made from the Second Edition in an attempt to give the book a more "modern" look. The text used in tables is the same.

The font used for source code is Liberation Mono, created by Red Hat, Inc. The Liberation fonts can be found here: <https://www.redhat.com/promo/fonts>.

Screen shots were taken using Ksnapshot (an excellent screen shot program that is part of the KDE desktop) as well as with the screenshot facility built into Apple's OS X and GNOME's "take a screenshot" utility. Other screen shots were taken by the contributors using unknown software. Drawings inside the book were created in OpenOffice.org draw, Inkscape, Dia, or some

Colophon

combination of the three. Some drawings and screen shots were touched up using the GIMP. Can you tell we like open source?

The cover of the book was done in a more traditional way, by Liferay's graphics team using Adobe Illustrator.

Rich Sezov, September 2010

INDEX

A

ajax.....17, 291, 293
Asset Publisher...5, 8, 71, 118, 134, 136, 159-161, 163, 165, 166, 172, 202, 298
authentication pipeline.....7, 249
auto deploy.....6, 210-212, 322, 323, 325
auto login.....7, 252, 253, 292

B

Backup.....122, 377, 378, 383, 388
Blog 3, 5, 8, 12, 14, 16, 22, 45-52, 61, 70, 82, 107, 121, 132, 133, 159-161, 165, 167, 173-181, 187, 191, 204-206, 211, 216, 217, 222, 234, 240, 251, 272, 289, 297, 299-302, 304-306, 308, 309, 312, 321, 327, 386
Blogs 5, 8, 12, 16, 82, 121, 132, 133, 159, 161, 165, 167, 173-180, 191, 204-206, 234, 240, 251, 272, 289, 299, 301, 304, 308, 386
bundle....3, 11, 21-29, 61, 184, 227, 313, 320, 321, 344, 355, 372

C

caching....9, 221, 256, 279-281, 289, 294, 341, 342, 351-354
Calendar. 5, 8, 12, 16, 68, 70, 84-86, 161, 167, 173, 180-183, 206, 241, 250, 272, 300, 308
captcha.....4, 7, 112, 257, 258
cas.4, 7, 8, 17, 18, 23, 44, 72, 83, 96, 97, 102-105, 127, 132, 140, 149, 186, 193, 194, 196, 208, 219, 224, 225, 234, 247-249, 252, 280-283, 288, 292, 311, 323, 330, 334, 338, 348, 349, 351, 353, 357, 361, 372, 378, 379, 383, 386, 392, 393, 396
categories 5, 6, 16, 17, 89, 155, 156, 160, 162, 168, 189, 190, 193, 200-204, 206, 256, 298, 380
chat...5, 12, 149, 167, 183, 184, 268, 300
Clustering.9, 11, 275, 280, 295, 341, 342, 344, 351-355, 376
cms.....117-119, 124, 135, 139, 140, 142, 149, 157, 166, 371, 401
Communities....3, 4, 8, 9, 12, 15-17, 63, 68-73, 76, 78-81, 89, 91, 94, 105-107,

	110, 115, 120, 123, 125, 126, 131, 161, 166, 167, 231, 274, 300, 301, 307, 314, 326, 364, 366, 378
	community...13, 17, 21, 67-72, 79-82, 88-92, 94, 110, 117, 119, 120, 122-128, 131, 133, 134, 137, 141, 147, 148, 156, 167-169, 173, 174, 176, 179, 180, 182, 186-188, 192, 198, 199, 202, 205-207, 214, 229, 235, 236, 251, 267, 268, 291, 300, 307, 311, 316, 326, 327, 332, 336, 355, 360, 364-367, 378, 401
D	Control Panel...3, 5, 7, 9, 12, 23, 63, 66-68, 72-75, 77, 79-81, 83, 88, 89, 91, 92, 95, 97, 98, 101, 103, 105, 108-110, 114, 115, 118, 119, 121, 125-127, 133, 137, 142-144, 150, 156-158, 162, 184, 192, 197, 200, 202, 205, 207, 274, 300, 316, 319, 322, 323, 325, 326, 333, 334, 348, 349, 355, 357, 364-367, 375, 378, 380, 381, 385, 387
E	CSS....9, 18, 110, 122, 148, 149, 218, 281, 282, 290, 292, 294, 301, 308, 372, 386
F	Data Migration.....4, 112, 387
G	Document Library8, 9, 16, 112, 113, 122, 144, 147, 287, 293, 301, 302, 336, 343-345, 353, 354, 379
H	freemarker.....149, 360
	glassfish3, 27, 32-35, 211, 213, 216, 321, 322
	hibernate. 6, 9, 19, 37, 215-217, 219-222, 224, 279, 280, 283, 352, 353, 369, 373, 375
	hot deploy...6, 9, 18, 209, 211, 213, 214, 218, 314, 315, 319-323, 343, 349, 351
	HSQl.....23, 28, 29, 216, 223, 276, 344
	html.....32, 131, 135, 139, 144, 146-149, 164, 191, 199, 200, 222, 224, 225, 227, 238, 240, 241, 264, 268, 272, 286, 289, 294, 301, 306, 312, 386, 401
I	image gallery.8, 113, 139, 144, 147, 151, 303
	invitation.....8, 241, 272, 303
J	jaas.....7, 17, 36-38, 44, 45, 244
	jackrabbit. .9, 16, 23, 112, 284, 324, 343-346, 350, 354, 355, 379, 384
	jar file.41, 43, 44, 46, 102, 208, 343, 347, 352
	javascript....6, 17, 18, 114, 131, 225-228, 264, 281, 282, 292, 294, 335, 358, 361, 362, 372
	JBoss.....3, 28, 37-40, 211, 212, 321
	JCR .8, 209, 219, 284, 301, 344, 373, 375, 381, 384, 385
	jetty.....3, 28, 35-37, 211
	jonas.....28, 211
	journal8, 9, 104, 215, 234, 236, 241, 251, 272, 279, 289, 303-306, 308, 396
	JSP 214, 225, 240, 241, 259, 265-268, 272, 289, 290, 312, 315
	JSR-168.....64, 313, 382
	JSR-170....23, 112, 284, 343-345, 354, 379
	JSR-286.....64, 313, 382
K	Kaleo...9, 11, 19, 157, 159, 355-357, 361, 362, 366-368, 376
L	language.....7, 15, 64, 87, 107, 114, 137, 141, 149, 151, 153, 165, 188, 215, 226, 238, 239, 264, 292, 297, 300, 304, 311,

314, 338, 357-362	navigation...9, 17, 66, 92, 107, 119, 123, 131, 133, 142, 144, 155, 165, 196, 201, 274, 299, 307, 308, 386
Language.properties...264, 297, 300, 304	NtLm...4, 7, 104, 233, 244, 247, 248, 252, 291, 293, 372
LAR file.....82, 110, 122, 218, 261-263	O
Layouts....7, 81, 108, 118, 132, 149, 166, 240, 260-269, 278, 307, 308	OpenId.....4, 7, 104, 105, 248, 252
LDAP.....4, 7, 17, 63, 73, 93, 95-103, 106, 115, 219, 232, 233, 245-250, 254	OpenSSO.....4, 7, 105, 106, 248, 252, 293
Liferay.....2	organizations...3, 7, 9, 12, 15-17, 22, 26, 63, 68-72, 76-80, 91, 92, 110, 115, 120, 125, 131, 161, 166, 231, 237, 238, 255, 274, 287, 300, 307, 314, 326, 337, 364, 366, 377, 378, 384
Liferay Portal....2, 9-13, 15-19, 21-23, 25, 27, 28, 30, 35, 37, 41, 42, 53, 56, 57, 60-64, 79, 97, 101, 104, 105, 107, 112-117, 122, 123, 125, 132, 134, 136, 137, 139, 142-144, 148, 153, 155, 165-167, 174, 177, 187, 208, 248, 313, 315, 316, 319, 322, 330, 334, 341, 346, 347, 354, 355, 366-368, 376-378, 387-389	P
liferay-portlet.xml.....241, 242, 274, 288, 372	passwords7, 93, 101, 234, 245, 249, 253-255, 337
locations.....237, 271	Performance Tuning.....9, 368
logging...10, 37, 173, 174, 178, 179, 185, 215, 227, 242, 245, 260, 290, 337, 377, 380-382	permissions....4-7, 10, 17, 26, 27, 69-73, 75, 76, 78, 82, 87-93, 109, 110, 118, 131, 133-135, 139, 140, 148, 153, 168, 170, 176, 182, 183, 188, 192-195, 197, 200, 202, 255-257, 268, 275, 329, 335, 376, 383, 387, 388
logs 5, 8, 12, 16, 24, 65, 82, 95, 100, 121, 132, 133, 159, 161, 165, 167, 173-180, 191, 204-206, 234, 240, 243, 251, 259, 263, 272, 289, 290, 299, 301, 304, 308, 353, 359, 360, 367, 386, 388	plugin..4, 6, 9, 10, 21, 23, 26, 27, 31, 64, 68, 91, 108, 111, 115, 116, 121, 132, 184, 207, 213, 214, 275, 293, 313-334, 338, 343, 346-349, 351, 352, 355, 356, 379, 381-383, 385, 388
Look and Feel....4, 7, 110, 121, 144, 167, 240, 308, 314, 315, 318, 319, 351, 385	portal-ext.properties..3, 6, 26, 29-32, 34, 39, 48, 51, 55, 59, 61, 82, 95, 96, 100, 101, 112, 113, 142, 183, 200, 207, 208, 319, 322, 323, 325, 335-338, 343, 344, 346, 349, 350, 352, 353, 372-375, 379, 383, 385, 387
lucene...7, 9, 23, 275-278, 324, 343, 347-350, 384	portal.properties 101, 208, 335, 343, 344, 350, 352, 353, 384
M	portlet.5-10, 13, 16-18, 21, 41, 53, 63-68, 71, 72, 77, 79-89, 91, 92, 106, 108-110, 113, 115, 119, 121, 126-128, 131-134, 136, 139-143, 151, 152, 154, 155, 158-161, 163, 165, 166, 168-188, 190-192, 194-198, 200, 201, 205-207, 209, 211-215, 217-220, 226, 229, 232, 233, 240-
mailing list.....5, 189, 192	
message boards5, 9, 12, 16, 89, 90, 106, 113, 127, 128, 165, 167, 168, 170, 186-195, 205, 206, 306, 307, 351, 354	
My Places.....9, 307	
mysql...29, 31, 33, 34, 36-42, 44, 48, 50, 51, 53, 55, 58-60, 216, 223, 224, 276, 345, 350, 355, 373, 375, 379	
N	

	334
Solr.....	111, 347-350
spring.....	6, 8, 19, 215-217, 219-222, 224, 296, 314, 335, 348, 373-376
SSO with MAC.....	7, 253
staging....	5, 118, 124-129, 131, 166, 305, 323
structure.	5, 16, 23, 25, 27, 46-48, 54, 73, 79, 105, 117, 118, 126, 135-137, 143-156, 166, 202, 219, 221, 224, 237, 251, 256, 304, 314, 319, 352, 373, 375, 387
T	
tag.	5, 6, 8, 9, 16, 26, 42, 61, 71, 80, 108, 118, 122, 124-129, 131, 135, 136, 144, 149, 154-157, 160, 162, 166, 177, 180, 187, 195, 196, 200-203, 206, 252, 268, 270, 273, 292, 294, 298, 301, 305, 310, 311, 314, 317, 323, 329, 330, 333, 338, 346, 356, 358, 360, 369, 370, 376, 382, 387
template....	4, 5, 9, 18, 29, 66, 71, 80-87, 107, 109, 110, 115, 118, 134-137, 143, 144, 146-149, 151-154, 166, 181, 210, 211, 213, 214, 240, 251, 261-263, 269, 279, 289, 291, 294, 304-306, 308, 313, 315, 318, 319, 329, 358-360
theme..	6, 9, 10, 17, 18, 21, 25, 107, 108, 110, 115, 121-123, 131, 132, 164, 199, 207, 211, 213, 214, 218, 225, 240, 261-264, 274, 289, 293, 294, 313-315, 318, 319, 323, 329, 351, 372, 378, 382, 384-386
time zone.....	7, 107, 182, 238, 239
tomcat.	3, 21-24, 27, 28, 43-45, 102, 103, 211-213, 243, 259, 320, 321, 338, 348, 352, 384
U	
upgrade...	6, 10, 12, 19, 26, 27, 210, 275, 277, 278, 321, 325, 327, 330, 349, 382-389
user groups...	3, 4, 12, 63, 69-71, 73, 82-84, 86-88, 93, 94, 106, 107, 115, 297

V

velocity...8, 149, 151, 289, 290, 294, 304, 306, 360

virtual host4, 8, 17, 55, 93, 94, 115, 123, 290, 291, 294

W

war file. 23, 30-32, 35, 37, 39, 48, 52, 53, 57, 59-61, 313, 319, 321, 323, 324, 328, 347, 351, 383

web content.....4, 5, 11, 12, 18, 68, 109, 113, 117, 128, 134-137, 139-144, 148, 150, 151, 154-163, 165, 166, 236, 367, 368

Web Form.....117, 317

WEB-INF....37, 39, 43, 102, 208, 320, 322, 346, 348, 352, 385, 386

web.xml.....212, 219, 233, 242, 247, 248, 268, 324, 372, 386

WebDAV.....8, 16, 19, 295, 296, 302, 335

WebLogic. 3, 22, 45-52, 61, 211, 216, 217, 222, 321

websphere..3, 22, 53, 55-57, 59, 60, 212, 217, 323, 324

wiki.....6, 9, 12, 13, 16, 53, 68, 106, 107, 114, 121, 122, 128, 132, 133, 152, 160-

162, 167, 195-201, 204-206, 234, 241, 252, 253, 272, 284, 285, 308, 311, 312, 401

workflow. 5, 9, 11, 16, 19, 118, 128, 134-136, 143, 157-159, 166, 219, 220, 226, 284, 315, 341, 342, 355-368, 376

WSDL.....9, 337, 338

X

xml.....32, 36-39, 41, 43-45, 52, 61, 123, 124, 149, 178, 212, 214, 215, 218-222, 224, 225, 233, 241, 242, 247, 248, 252, 268, 270, 274, 280, 281, 284, 288, 292, 293, 301, 304, 324-326, 332-334, 344, 348, 352-359, 361, 363, 364, 366, 372-376, 385, 386, 388

xsl.....149, 304, 306

Z

zip...23, 24, 27, 28, 38, 39, 41-43, 46, 49, 292, 301, 386, 388

. 38, 41, 45, 210-213, 222, 228, 229, 234-236, 240, 242, 243, 246, 247, 250, 252, 257, 259, 262, 266, 268-270, 272, 276-280, 284, 289, 294, 296, 297, 301, 303, 305, 306, 308, 309, 312, 322, 345, 351