



Portal Administrator's Guide

Liferay Administrator's Guide
by Richard L. Sezov, Jr.
Copyright © 2009 by Liferay, Inc.

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, 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, Michael Young

Table of Contents

1. Introduction.....	15
CREATED FOR THE ENTERPRISE.....	15
Personalization and easy customization.....	15
Workflow adaptable.....	16
Branding friendly.....	16
Flexible organization.....	16
Built With The End-User In Mind.....	16
Award-winning UI.....	16
One-Click Look and Feel Changes.....	16
Web OS.....	16
TECHNOLOGY TO HELP DEVELOPERS.....	16
Standards Compliant.....	17
Ready Integration.....	17
Liferay Plugins Catalog.....	17
SUPPORTED TECHNOLOGIES	17
LANGUAGES	17
2. Initial Setup.....	19
EDITIONS OF LIFERAY.....	19
OBTAINING LIFERAY.....	20
INSTALLING A BUNDLE.....	20
INSTALLING LIFERAY FOR AN ENTERPRISE.....	22
Sample Data.....	22
Liferay Home.....	22
Database Setup.....	23
Default Method: Automatic.....	23
Manual Method.....	23
Turning a Bundle into an Enterprise Portal.....	24
The portal-ext.properties File.....	25
Installing Liferay on an Existing Application Server.....	27
Installing Liferay in 10 Easy Steps.....	27
GlassFish 2.x.....	28
GlassFish 3.x.....	31
Jetty 6.....	34
JBoss 4.03sp1 / 4.04 / 4.05 / 4.2 / 4.3.....	36
JBoss 5.x.....	39
Oracle Application Server (OC4J).....	42
Resin 3.1.x.....	47
Resin 3.2.x.....	49
Tomcat 5.5.X.....	50
WebLogic 9 / 10.....	52
Oracle WebLogic 10.3.....	55
WebSphere 6.1.....	58
WebSphere 7.0.....	69
Making Liferay Coexist with Other Java EE Applications.....	74
SUMMARY.....	75
3. Configuration.....	77
LIFERAY'S USER INTERFACE.....	77

Navigating Liferay.....	78
Navigating the Control Panel.....	80
PORTAL ARCHITECTURE.....	81
Users.....	82
User Groups.....	82
Roles.....	82
Organizations.....	83
Communities.....	83
USING THE CONTROL PANEL.....	84
Adding Users.....	84
User Management.....	86
Organizations.....	87
Communities.....	89
Community Provisioning.....	90
User Groups.....	92
User Groups and Page Templates.....	93
Roles.....	98
Defining Permissions on a Role.....	98
Special Note about the Power Users Role.....	101
GLOBAL SERVER SETTINGS.....	101
Password Policies.....	101
Settings.....	102
General.....	102
Authentication: General Settings.....	102
Authentication: LDAP.....	103
Single Sign-On.....	109
Authentication: Central Authentication Service (CAS).....	109
Authentication: NTLM.....	110
Authentication: OpenID.....	111
Authentication: OpenSSO.....	111
Authentication: SiteMinder.....	112
Default User Associations.....	112
Reserved Credentials.....	112
Mail Host Names.....	113
Email Notifications.....	113
Identification.....	113
Miscellaneous: Display Settings.....	113
Monitoring.....	113
Plugins Configuration.....	113
Server Administration.....	114
Resources.....	114
Log Levels.....	114
System Properties.....	114
Portal Properties.....	115
Shutdown.....	115
OpenOffice.....	115
Portal Instances.....	116
Plugins Installation.....	116
SUMMARY.....	116
4. Liferay Collaboration Suite.....	119
SCOPES.....	119
ARCHIVED SETUPS.....	121

PERMISSIONS.....	121
SHARING.....	122
Any Web Site.....	122
Facebook.....	122
Friends.....	123
BLOGS.....	123
The Blogs Portlet.....	124
Configuring the Blogs Portlet.....	125
Aggregating Blog Entries.....	128
CALENDAR.....	129
Configuring the Calendar Portlet.....	129
Event From.....	129
Event Reminder Email.....	129
Display Settings.....	130
Using the Calendar Portlet.....	130
CHAT.....	131
MAIL.....	132
MESSAGE BOARDS.....	133
Email From.....	134
Message Added Email.....	134
Message Updated Email.....	134
Thread Priorities.....	134
User Ranks.....	134
RSS.....	135
Anonymous Posting.....	135
Ratings.....	135
Permissions.....	135
Adding Categories.....	136
Using the Message Boards.....	137
Message Board Administrative Functions.....	139
Moving Threads.....	140
Deleting Threads.....	140
Banning Users.....	140
Splitting Threads.....	140
Editing Posts.....	140
Permissions.....	141
WIKIS.....	141
Getting Started with the Liferay Wiki.....	141
Managing Wikis.....	142
Adding and Editing Wiki Pages.....	143
Page Details.....	144
Details.....	144
History.....	145
Incoming / Outgoing Links.....	145
Attachments.....	145
Navigating in the Wiki Portlet.....	145
SUMMARY.....	146
5. Advanced Liferay Configuration.....	147
The portal-ext.properties File.....	147
Properties Override.....	148
Liferay Home.....	148
Portal Context.....	148

Resource Repositories Root.....	149
Technology Compatibility Kit.....	149
Schema.....	149
Upgrade.....	149
Verify.....	150
Auto Deploy.....	150
Hot Deploy.....	152
Hot Undeploy.....	153
Plugin.....	153
Portlet.....	153
Portlet Coordination.....	154
Theme.....	154
Resource Actions.....	155
Model Hints.....	155
Service Builder.....	155
Spring.....	155
Hibernate.....	156
JDBC.....	158
Custom SQL.....	160
Database.....	160
Ehcache.....	160
JavaScript.....	161
SQL Data.....	164
Company.....	165
Users.....	166
Groups and Roles.....	167
Organizations.....	169
Languages and Time Zones.....	170
Look and Feel.....	172
Request.....	172
Session.....	172
JAAS.....	174
LDAP.....	175
CAS.....	177
NTLM.....	177
OpenID.....	178
OpenSSO.....	178
SiteMinder.....	178
Authentication Pipeline.....	179
Auto Login.....	181
SSO with MAC (Message Authentication Code).....	182
Passwords.....	183
Permissions.....	184
Captcha.....	185
Startup Events.....	185
Shutdown Events.....	186
Portal Events.....	186
Login event.....	186
Logout event.....	187
Default Landing Page.....	187
Default Logout Page.....	187
Default Guest Public Layouts.....	187

Default User Private Layouts.....	188
Default User Public Layouts.....	189
Default Admin.....	190
Layouts.....	190
Default Settings Layouts.....	191
Portlet URL.....	195
Preferences.....	196
Struts.....	196
Images.....	196
FileSystemHook.....	197
Editors.....	197
Fields.....	197
Mime Types.....	197
Amazon.....	198
Browser Launcher.....	198
Control Panel.....	198
Instant Messenger.....	199
Lucene Search.....	199
SourceForge.....	201
Value Object.....	202
Communication Link.....	202
Content Delivery Network.....	203
Counter.....	203
Lock.....	203
JBI.....	203
JCR.....	203
Live Users.....	204
Lock.....	204
Mail.....	204
OpenOffice.....	206
POP.....	206
Quartz.....	207
Scheduler.....	207
Search Container.....	207
Sharepoint.....	207
Social Bookmarks.....	208
Velocity Engine.....	208
Virtual Hosts.....	209
HTTP.....	209
Servlet Filters.....	210
Upload Servlet Request.....	211
Web Server.....	211
WebDAV.....	211
Main Servlet.....	212
Axis Servlet.....	212
JSON Tunnel Servlet.....	212
Liferay Tunnel Servlet.....	212
Spring Remoting Servlet.....	212
WebDAV Servlet.....	213
Widget Servlet.....	213
Admin Portlet.....	213
Announcements Portlet.....	213

Asset Publisher Portlet.....	214
Blogs Portlet.....	214
Calendar Portlet.....	214
Communities Portlet.....	215
Document Library Portlet.....	215
Image Gallery Portlet.....	216
Invitation Portlet.....	217
Journal Portlet.....	217
Journal Articles Portlet.....	219
Journal Content Search Portlet.....	219
Message Boards Portlet.....	219
My Places Portlet.....	220
Navigation Portlet.....	220
Nested Portlets Portlet.....	221
Portlet CSS Portlet.....	221
Shopping Portlet.....	221
Software Catalog Portlet.....	222
Tags Compiler Portlet.....	222
Tags Portlet.....	222
Tasks Portlet.....	223
Translator Portlet.....	223
Wiki Portlet.....	223
PLUGIN MANAGEMENT.....	225
Portlets.....	225
Themes.....	226
Layout Templates.....	227
Hook Plugins.....	227
Web Plugins.....	227
Installing Plugins from Repositories.....	227
Installing Plugins Manually.....	230
Plugin Troubleshooting.....	231
Liferay Configuration Issues.....	232
The Container Upon Which Liferay Is Running.....	233
Changing the Configuration Options in Multiple Places.....	235
How Liferay Is Being Launched.....	235
Creating Your Own Plugin Repository.....	236
The Software Catalog.....	237
Manually Creating A Software Catalog.....	244
Connecting to a Software Catalog.....	244
LIFERAY SERVICES ORIENTED ARCHITECTURE.....	244
Accessing Liferay's WSDL.....	247
6. Enterprise Configuration.....	249
LIFERAY CLUSTERING.....	250
All Nodes Should Be Pointing to the Same Liferay Database.....	251
Document Library Configuration.....	251
Jackrabbit Sharing.....	252
Search Configuration.....	253
Pluggable Enterprise Search.....	253
Lucene Configuration.....	255
Hot Deploy.....	256
DISTRIBUTED CACHING.....	256
Hibernate Cache Settings.....	258

Clustering Jackrabbit.....	259
WORKFLOW.....	259
Installation and Test.....	260
Using Different Databases.....	261
How the Workflow Portlet Works.....	261
Process Definitions.....	261
Integrating with Users, Communities, and Roles.....	261
Data Types and Error Checking.....	263
Sample Process Definitions.....	265
Warning Messages.....	266
Administration.....	267
Deploying Workflows.....	267
Managing Instances.....	268
Managing Tasks.....	270
Future Enhancements.....	271
Logging.....	271
Customizable Front-End.....	271
File Upload Data Type.....	272
Frequently Asked Questions.....	272
How do you write a new process definition?.....	272
Why are there “Duplicate File” exceptions when I change databases for jBPM?.....	272
DEPLOYING A CUSTOMIZED LIFERAY.....	272
Deploying Directly on the Server.....	273
Deploying from a Client Machine.....	274
PERFORMANCE TUNING.....	274
Memory.....	274
Garbage Collection.....	275
Properties File Changes.....	277
Servlet Filters.....	278
Portlets.....	278
Read-Writer Database Configuration.....	278
Database Sharding.....	279
7. Maintaining A Liferay Portal.....	283
LIFERAY MONITORING USING GOOGLE ANALYTICS.....	283
BACKING UP A LIFERAY INSTALLATION.....	284
Source Code.....	284
Liferay's File System.....	285
Database.....	285
LIFERAY'S LOGGING SYSTEM.....	285
UPGRADING LIFERAY.....	287
Liferay Upgrade Procedure.....	287
Upgrade Steps.....	288
Upgrading Liferay 4.3 to Liferay 4.4.....	288
Prerequisite	288
If Your Developers Have Customized Liferay.....	288
Upgrading Liferay 4.4 to Liferay 5.0.....	289
Prerequisite	289
If Your Developers Have Customized Liferay.....	289
Converting wiki pages (optional)	289
Upgrade Troubleshooting	290
Upgrading Liferay 5.0 to Liferay 5.1.....	290

Changes in configuration properties	290
How to keep the old values	290
What has been changed?	290
If Your Developers Have Customized Liferay.....	290
Upgrading Themes	291
The Parent Element of the Dock May Change Positioning When Upgrading.....	292
The Class Names for Different UI Components Have Changed .	292
Change in Theme CSS Fast Load.....	292
Change in Javascript Fast Load.....	293
Upgrading PHP Portlets	293
Javascript changes	293
Upgrading From Liferay 5.1 to Liferay 5.2.....	294
Prerequisite	294
Changes in configuration properties	295
Theme Upgrade	295
API Changes	296
8. Appendix: Documentation License.....	299
CREATIVE COMMONS LICENSE.....	299
License.....	299
Creative Commons Notice.....	305
9. Colophon.....	307
Index.....	310

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, or anything in between.

This book was written with the server administrator in mind. It is a guide for anyone who wants to get a Liferay Portal server up and running, and will guide you step-by-step through the installation and configuration process. 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 hopefully 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. You'll see how to use Liferay's collaboration suite of portlets to enable your users to work together more efficiently and more easily. From there we go into advanced administration topics and enterprise configuration, including clustering and integrating Liferay with other services. 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 Third Edition

Certainly, Liferay Portal has not stood still since the last edition was written. This edition has been updated so that it covers Liferay Portal up to version 5.2. Chapter 5 (*Advanced Liferay Configuration*) has been completely revamped to that it covers all of the new portal properties, and the rest of the book has been exhaustively gone through and updated.

The chapter on Portal Administration (Chapter 3) has been overhauled so that it covers Liferay administration using the new Control Panel. This chapter also goes over portal design, listing the things you might want to consider as you build your web site on Liferay Portal.

For this edition of the book, a new chapter on Liferay's unparalleled collaboration suite of portlets has been provided. This chapter will guide you through enabling your users to collaborate using the robust set of tools that Liferay provides: blogs, calendar, chat, mail, message boards, and wikis.

Other chapters have been expanded to include additional information. For example, Chapter 7 (*Maintaining a Liferay Portal*) now covers database sharding among other things, and an updated section on upgrading Liferay.

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 like this.

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 and to indicate a label or a name of a Java class.

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/web/guest/services> for further information about the services we can provide.

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/web/guest/about_us/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

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. A couple of months later, it was rehashed electronically in outline form among a small group of Liferay employees until the final list of content was considered complete. This seems like a world away now, as so much has changed, not only with the product, but also with the company, as Liferay has grown so much in the intervening years.

For this third edition, I had a target deadline to meet: the book had to be ready for the release of the 5.2 Enterprise Edition of the product. A lot of hard work went into polishing the material and making it as error-free as possible. Of course, no one is perfect, and so if we find any problems with the material, they will be corrected on the errata page for the book in Liferay's wiki (<http://wiki.liferay.com>).

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 my job would be a lot more difficult were it not for their patience with me when I interrupt their work with some (pretty dumb, sometimes) questions. So special thanks are due to Ray Auge, Nate Cavanaugh, Brian Chan, Alex Chow, Bruno Farache, Jorge Ferrer, and Mike Young.

I'd also like to thank my daughter Julia for checking in on me from time to time and bringing some good cheer with her. 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. I couldn't do any of this without her.

Rich Sezov

<http://www.liferay.com/web/rsezov/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.

1. Runs on all major application servers and servlet containers, databases, and operating systems, with over 700 deployment combinations
2. JSR-286 Compliant
3. Out-of-the-box usability with over 60 portlets pre-bundled.
4. Built in Content Management System (CMS) and Collaboration Suite
5. Personalized pages for all users
6. Benchmarked as among the most secure portal platforms using LogicLibrary's Logiscan suite

Created for the enterprise, Liferay Portal provides a virtual space where you can centralize, share and collaborate.

Built with the end user in mind, Liferay Portal's award winning user interface is easy enough to master by even the least technical of users.

Liferay Portal also remains one of the most popular portal technologies within the developer community with an ever-growing list of features that help your IT team deploy business solutions with minimal time and effort.

Created for the Enterprise

Personalization and easy customization

Give the right people the right access to the right applications and documents! A highly granular permissions system allows you to customize the user experience at

the organizational and personal level.

Workflow adaptable

Liferay's technology is built to quickly adapt business and organizational changes, ensuring minimal downtime in today's fast-changing market.

Branding friendly

Liferay Portal is coded to easily adapt to your organization's desired branding and look and feel.

Flexible organization

Liferay Portal is highly scalable for large, growing user bases. It accommodates the functional needs of even the most complex of enterprises--For example, your sub organizations can each be given its own portal experience with unique URL, login, look and feel and security permissions!

Built With The End-User In Mind

Liferay Portal's award winning user interface offers dynamic, intuitive and time saving features that fosters user adoption across your entire organization.

AWARD-WINNING UI

Liferay Portal offers dynamic, intuitive and time saving features that foster user adoption across your entire organization. We were the first portal to introduce drag-and-drop portlet re-positioning and continue to deliver innovative usability features for even the least technical of users!

ONE-CLICK LOOK AND FEEL CHANGES

Deploy a new look and feel with one click without having to touch any code!

WEB OS

Work with Liferay's Document Library like a network drive on your desktop with familiar folders. An optional free-floating portlet theme mimics the look and feel of your desktop environment.

Technology to Help Developers

Protect your development investment. Liferay Portal's active and mature community always keeps the developer experience in mind.

Standards Compliant

Liferay Portal complies with key industry standards, making it easy to work and integrate with.

Ready Integration

Partnerships with leading Open Source players such as Alfresco, ICEsoft, Spring-Source, and Pentaho ensure unparalleled integration and support for these widely-used technologies.

Liferay Plugins Catalog

This exclusive feature of Liferay Portal keeps track of new versions of the software and instantly updates the technology without troublesome restarts.

Liferay supports ALL major app servers, databases, and operating systems with over 700 deployment configurations.

Please see our deployment matrix for more details.

Supported Technologies

Technologies Used:

- Apache ServiceMix
- ehcache
- Hibernate
- ICEfaces
- Java J2EE/JEE
- jBPM
- JGroups
- jQuery JavaScript Framework
- Lucene
- MuleSource ESB
- PHP
- Ruby
- Seam
- Spring & AOP
- Struts & Tiles
- Tapestry

- Velocity

Standards:

- AJAX
- iCalendar & Microformat
- JSR-286
- JSR-127
- JSR-170
- Seats on the JSR-286 (Portlet 2.0) and JSF-314 (JSF 2.0) committees
- OpenSearch
- Open platform with support for web services including:
 - JSON
 - Hessian
 - Burlap
 - REST
 - RMI
 - WSRP

- WebDAV

Languages

Liferay supports I18N for any language out-of-the-box and ships with default translations for 22 languages. Most of these translations are maintained by native

speakers of the languages from Liferay's open source community. Additional languages can be added very easily, and this has been done many times over the life of the product.

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 Standard Edition (SE) and Liferay Portal Enterprise Edition (EE). SE is the same Liferay Portal that has been available for years: frequently updated and bursting with the latest features, the standard edition of Liferay Portal is offered for free under the business-friendly MIT open source license. 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 SE, 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 site is 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 cli-

ents.

Obtaining Liferay

The SE 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 will be provided to you as a result of your support subscription. You will be provided with 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.

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. Having a JDK (Java Development Kit) already installed is a prerequisite to running any of the bundles.

Please note that Liferay is not able to provide application server bundles for proprietary application servers such as WebLogic, WebSphere, or Oracle Application Server 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*.

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.



Illustration 1: Bundle directory structure

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 uncompressing 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 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 startup, 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.

The screenshot shows the Liferay 5.2 default page. At the top, there's a dark navigation bar with the Liferay logo and the tagline "Enterprise Open Source For Life". Below this is a "Welcome" message. The main content area is divided into several sections:

- Welcome to Liferay:** A introductory paragraph stating that Liferay is a complete all-in-one open source enterprise portal solution used by over 100,000 organizations worldwide.
- INSTANT WEBSITES:** A section titled "Use Liferay Web Content Manager to create websites quickly with minimal effort and cost." It lists features like creating content, layouts, and custom branding, and mentions that Liferay is used by Benetton, Lufthansa Flight Training, and more.
- SOCIAL NETWORKS:** A section titled "Unlock the latent value in your enterprise's community using Liferay's Social Networking capabilities." It lists features like customizable personal pages, expanding networks by project group, and adding friends.
- SHARED WORKSPACES:** A section with a dark background.
- SOA, ENTERPRISE INTEGRATION, AND MASHUPS:** A section with a dark background.
- 7 Cogs Inc. Demo:** A featured section for a sample website built in Liferay, with a "CLICK HERE" button. It includes a small text box explaining that this is a demo site and provides a link to upgrade to Liferay 5.2.
- Current Users:** A section at the bottom right showing a list of users, with "Bruno (Admin)" visible.

Illustration 2: Default Page in the Liferay Bundles

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.

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

Eventually, you will want to install Liferay onto a real server, after you've completed your evaluation and you want to begin building your web site. 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 this section, therefore, will focus on installing Liferay onto an already-established application server.

Sample Data

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 included 7 Cogs applications from the bundle.

There are three applications included in the bundle that you will need to remove. These three modify Liferay's default behavior in order to make the 7 Cogs application work properly. Because we want to revert Liferay's behavior back to its defaults for a clean install, you will want to remove:

- **sevendcogs-hook**
- **sevendcogs-theme**
- **wol-portlet**

If you forget to undeploy these three applications 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 these three applications undeployed before setting up your server. Use your application server's method for undeploying applications in order to remove them.

Liferay Home

In Liferay 5.2.0 and higher, there is a new folder defined as *Liferay Home*. This

folder for most application servers is one folder higher than the location of the application server itself. In the case of a server which defines different domains for different instances of the server, this folder may be one folder up from the domain in which Liferay is running.

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 will be fully documented in Chapter 5: *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 through various plugin installs which may create tables of their own and through upgrades of the platform. 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 SE 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 Journal 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 2 and 3
- JBoss+Tomcat 4.2
- JBoss+Tomcat 5.0
- Jetty
- JOnAS+Jetty
- JOnAS+Tomcat
- Resin
- Tomcat 5.5
- 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 redeployment.
- 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 5 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=smtp
```

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. Generally, this is one folder up from where your application server is installed. 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.

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.

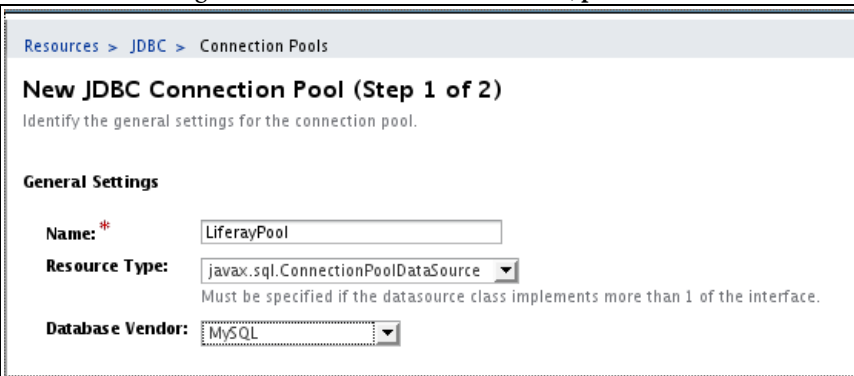


Tip: Note that Liferay 5.x *requires* JDK 1.5 or greater. Do not attempt to install Liferay 5.x on an application server that runs under Java 1.4 or lower; it will not work. If you are running an application server that ships with a JDK and that JDK is 1.4 or lower, you will need to upgrade your application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

GLASSFISH 2.X

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

1. Download the latest Liferay Portal .war file and dependencies.
2. Copy the dependencies .jars into `$GLASSFISH_HOME/domains/domain1/lib`, where `$GLASSFISH_HOME` is the directory where Glassfish is installed.
3. Copy `xercesImpl.jar` and JDBC driver into the same place.
4. Start Glassfish if it hasn't already been started. Go to the Glassfish Administration Console at <http://localhost:4848>.
5. Default login credentials are **user name:** *admin*; **password:** *adminadmin*.



Resources > JDBC > Connection Pools

New JDBC Connection Pool (Step 1 of 2)

Identify the general settings for the connection pool.

General Settings

Name: *

Resource Type: Must be specified if the datasource class implements more than 1 of the interface.

Database Vendor:

Illustration 3: Glassfish JDBC Connection Pool

DATABASE CONFIGURATION

If you want Glassfish to manage your data source, follow the instructions below. If you want Liferay to manage your data source, you can skip this section.

1. Under *Other Tasks*, select *Create New JDBC Connection Pool*.
2. 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*.
3. 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
```

If your database is not on the same server as Glassfish, substitute your database server's host name for *localhost* above.

4. Click *Add Property*, and add a property called *user* with a value of the user name to connect to the database.
5. Click *Add Property* again, and add a property called *password* with a value of the password to connect to the database.
6. Click *Finish*.
7. 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.
8. Click *JDBC Resources*. You will see a list of JDBC Resources by JNDI Name.
9. Click *New*.
10. Make the JNDI Name *jdbc/LiferayPool* and select the *LiferayPool* you created earlier.
11. Click *OK*.

Resources > JavaMail Sessions

New JavaMail Session

A JavaMail session resource represents a mail session in the JavaMail API.

General Settings

JNDI Name: * mail/MailSession
A unique identifier, contain only alphanumeric, underscore, dash, or dot characters

Mail Host: * localhost
DNS name of the default mail server

Default User: * glassfish
User name to provide when connecting to a mail server, must contain only alphanumeric, underscore, dash, or dot characters

Default Return Address: * glassfish@liferay.com
E-mail address of the default user

Description:
Makes it easier to find this session later

Status: Enabled

Illustration 4: Glassfish Mail Session

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.
4. Click *OK*.

DEPLOY LIFERAY

1. Click *Application Server* at the top of the tree hierarchy on the left.
2. Click *JVM Settings* -> *JVM Options*.
3. Click *Add JVM Option*, and enter the following:

```
-Dcom.sun.enterprise.server.ss.ASQuickStartup=false
```
4. Click *Save*.
5. Log out of the Administration Console and stop Glassfish.
6. Deploy Liferay by copying the *liferay-portal-x.x.x.war* file you downloaded in step 1 into the *\$GLASSFISH_HOME/domains/domain1/autodeploy* directory.
7. 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.

8. Start GlassFish. When Liferay finishes starting, open <http://localhost:8080> in your browser. You should see the default Liferay home page.

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.

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.

1. Go to the GlassFish console URL: <http://localhost:4848>.

Resources > JDBC > Connection Pools

New JDBC Connection Pool (Step 1 of 2)

Identify the general settings for the connection pool.

General Settings

Name: *

Resource Type: ▼
 Must be specified if the datasource class implements more than 1 of the interface.

Database Vendor: ▼

Illustration 5: Glassfish JDBC Connection Pool

- Under *Other Tasks*, select *Create New JDBC Connection Pool*.
- 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*.
- 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
```

- If your database is not on the same server as Glassfish, substitute your database server's host name for *localhost* above.
- Click *Add Property*, and add a property called *user* with a value of the user name to connect to the database.
- Click *Add Property* again, and add a property called *password* with a value of the password to connect to the database.
- Click *Finish*.
- 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.
- Click *JDBC Resources*. You will see a list of JDBC Resources by JNDI Name.
- Click *New*.
- Make the JNDI Name *jdbc/LiferayPool* and select the *LiferayPool* you created earlier.
- Click *OK*.

MAIL CONFIGURATION

At the time of this writing, JavaMail is not yet implemented in GlassFish 3. For this reason, you will have to use the mail session that is provided by Liferay.

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
```

GlassFish 3 has not yet implemented JavaMail, so you do not have the option to use one via JNDI.

Save and close the file.

2. Go to the GlassFish console URL: <http://localhost:4848>
3. Click *Web Applications* in the tree on the left.
4. Click the *Deploy* button.
5. Click *Browse* and browse to the location of the Liferay .war file.
6. Leave the rest of the defaults and click *OK*.

Liferay will be deployed and started automatically.

Illustration 6: Deploying Liferay in GlassFish 3

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. Create a `$JETTY_HOME/lib/ext` directory and copy the dependencies to it.
5. Edit `$JETTY_HOME/extra/etc/start-plus.config`.

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

6. Create a data source bound to `jdb/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>jdb/LiferayPool</Arg>
      <Arg>
        <New class="org.enhydra.jdbc.standard.StandardXDataSource">
          <Set name="DriverName">com.mysql.jdbc.Driver</Set>
          <Set name="Url">jdbc:mysql://localhost/lportal?useUnicode=true&characterEncoding=UTF-8</Set>
        </New>
      </Arg>
    </New>
  </Arg>
</Call>
```

```

        <Set name="User"></Set>
        <Set name="Password"></Set>
    </New>
</Arg>
<Arg>
    <New class="org.enhydra.jdbc.pool.StandardXAPoolData-
Source">
        <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-{$version}-bin.jar` and copy to to `$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 `xercesImpl.jar` and `xml-apis.jar`.
15. Copy `$JETTY_HOME/webapps/root/WEB-INF/lib/commons-logging.jar` to `$JETTY_HOME/ext` (overwriting existing one).
16. Create batch file.
 1. Create a directory `$JETTY_HOME/bin`.
 2. Create `run.bat` (Note, this is for Windows platform. For other plat-

forms, configure accordingly).

```
@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 `-Xmx512m`.

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

JBoss 4.03SP1 / 4.04 / 4.05 / 4.2 / 4.3

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

2. Download and install JBoss AS into your preferred directory. From now on, the directory where you installed JBoss will be referred to as `$JBOSS_HOME`.
3. Download the Liferay Portal .war file and dependencies.
4. Edit `$JBOSS_HOME/server/default/deploy/jboss-web.deployer/conf/web.xml`. Replace the default servlet (lines 79-91) :

```
<servlet>
    <servlet-name>default</servlet-name>
    <servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class>
    <init-param>
        <param-name>debug</param-name>
        <param-value>0</param-value>
    </init-param>
    <init-param>
        <param-name>listings</param-name>
        <param-value>>true</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>
```

with:

```

<servlet>
  <servlet-name>default</servlet-name>
  <servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class>
  <init-param>
    <param-name>debug</param-name>
    <param-value>0</param-value>
  </init-param>
  <init-param>
    <param-name>listings</param-name>
    <param-value>>false</param-value>
  </init-param>
  <init-param>
    <param-name>input</param-name>
    <param-value>4096</param-value>
  </init-param>
  <init-param>
    <param-name>output</param-name>
    <param-value>4096</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

```

5. Go to `$JBOSS_HOME/server/default/lib/`, download `mysql-connector-java-
{version}-bin.jar` and copy to this directory. This is the JDBC driver for MySQL. If you are using a different database, copy the appropriate driver.
6. Put the Liferay Portal dependency jars into `$JBOSS_HOME/server/default/lib`.

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 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

Configure JAAS. Edit `$JBOSS_HOME/server/default/conf/login-config.xml` and comment out the entire XML for policy other in lines 140-156.

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

Deploy the Liferay Portal .war file.

- Create new directory `$JBOSS_HOME/server/default/deploy/liferay-portal.war`.
- Unzip `liferay-portal-x.x.x.war` to directory.
- Go into `$JBOSS_HOME/server/default/deploy/liferay-portal.war/lib`.
 - Move `dom4j.jar`, `jaxen.jar` to `$JBOSS_HOME/lib`.
 - Move `commons-collections.jar` goes to `$JBOSS_HOME/server/default/lib`.

- remove `hibernate3.jar`, `jboss-hibernate.jar` from `JBOSS_HOME/server/default/lib`.

Edit `JBOSS_HOME/server/default/deploy/jbossjca-service.xml`. Change the Debug attribute in line 63 from true to false:

```
<attribute name="Debug">false</attribute>
```

Edit `JBOSS_HOME/server/default/deploy/jms/jbossmq-destinations-service.xml`. Clear out text between server tags:

```
<?xml version="1.0"?>
<server>
</server>
```

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.

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. Go to `$JBOSS_HOME/server/default/lib/`. Download `mysql-connector-java-
{version}-bin.jar` and copy to this directory. (This is the JDBC connector for MySQL. Use the appropriate driver for your database.)
4. Download Liferay's Portal Dependencies. Unzip to `$JBOSS_HOME/server/default/lib/`.
5. 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 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 `$JBOSS_HOME/server/default/deploy/ROOT.war`
2. Unzip the Liferay .war file to the ROOT.war directory.
3. Go to `$JBOSS_HOME/server/default/deploy/ROOT.war/lib`.
4. Remove `jaxen.jar`, `jaxrpc.jar`, `stax.jar`, `xercesImpl.jar`, `xml-apis.jar` from `$JBOSS_HOME/server/default/deploy/ROOT.war/WEB-INF/lib`
5. Navigate to the Liferay Home folder, which is one folder above JBoss's install location.
6. 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&characterEn-
coding=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.

ORACLE APPLICATION SERVER (OC4J)

Liferay Home is located in the home folder of the user running OC4J.

These instructions assume you have an installed OC4J container 10.1.3.4.0 in a folder that will be referred to as *\$OC4J_HOME*.

Unzip the dependencies archive (which is downloadable from Liferay's web site) into *\$OC4J_HOME/j2ee/home/applib*. Put your JDBC driver here as well.

OC4J contains a startup script in the *bin* folder of the application server directory. You will need to edit this file and change some of the arguments. On Windows systems, the filename of the script is *oc4j.cmd*. On Linux / Unix systems, the filename of the script is *oc4j*.

Find the line in this file that contains the JVMARGS variable:

```
JVMARGS=$OC4J_JVM_ARGS
```

Modify the line so it reads like the following:

```
JVMARGS=$OC4J_JVM_ARGS -Doc4j.jmx.security.proxy.off=true -Xms512m -Xmx512m -XX:MaxPermSize=256m
```

DATABASE AND CONNECTION POOL

If you are using Liferay's built-in data source, you can skip this section.

1. Click *Administration* -> *JDBC Resources*.

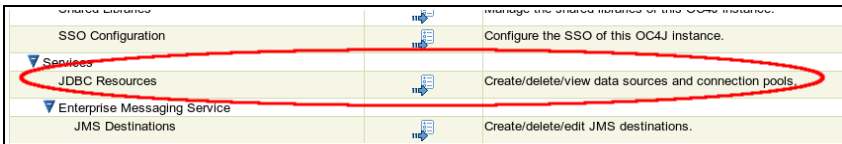


Illustration 7: OC4J: JDBC Resources

2. Under *Connection Pools*, click *Create*.
3. Select *New Connection Pool* and click *Continue*.
4. Give it a name, such as *LiferayPool*. If you are not using an Oracle database, you will have to specify the connection factory class manually. For example,

the connection factory class for MySQL is `com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource`.

- Fill out the JDBC URL to your database. You should have already configured your database and optionally run the Liferay create script.

The screenshot shows the Oracle Enterprise Manager 10g Application Server Control interface. The breadcrumb path is "OC4J: home > JDBC Resources > Connection Pool: 'LiferayPool'". The interface has three tabs: "Home", "Attributes", and "Proxy Interfaces".

Attributes:

- Name: **LiferayPool**
- Connection Factory Class: **com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource**
- * URL: `jdbc:mysql://localhost:3306/oc4j/liferay?useUnicode=true&`

Credentials:

- Username:
- Use Cleartext Password
 - Password:
- Use Indirect Password ^(D)
 - Indirect Password:
 - example: Scott, customers/Scott

Connection Factory Properties:

Specify any properties needed by the connection factory here.

Name	Value
(No properties found)	
Add Another Row	

At the bottom, there are three tabs: "Home", "Attributes", and "Proxy Interfaces".

Illustration 8: OC4J: Connection Pool

- Click *Finish*. Under *Data Sources*, click *Create*. Select *Managed Data Source* (default) and click *Continue*.
- Give it a name (*LiferayDataSource*) and a JNDI location of *jdbc/LiferayPool*.
- Select the Connection Pool you created in the previous step and then click *Finish*.

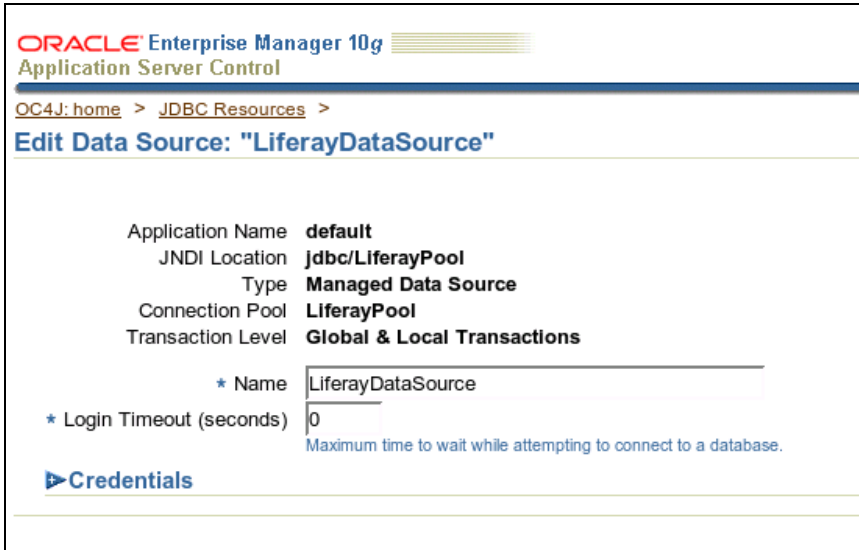


Illustration 9: OC4J: Liferay Data Source

- Click the *Test Connection* icon. If you are not using an Oracle database, change the test SQL to something like `SELECT * from User_`, and then click Test. You should get a message saying *Connection to "LiferayDataSource" established successfully*. If not, go back and check your settings.
- Click the *OC4J: home* link in the top left of the page.

INSTALLING LIFERAY

- Click *Applications*.
- Click *Deploy*. Leave the default selected under Archive, and click the *Browse* button. Browse to where your Liferay .war file is.

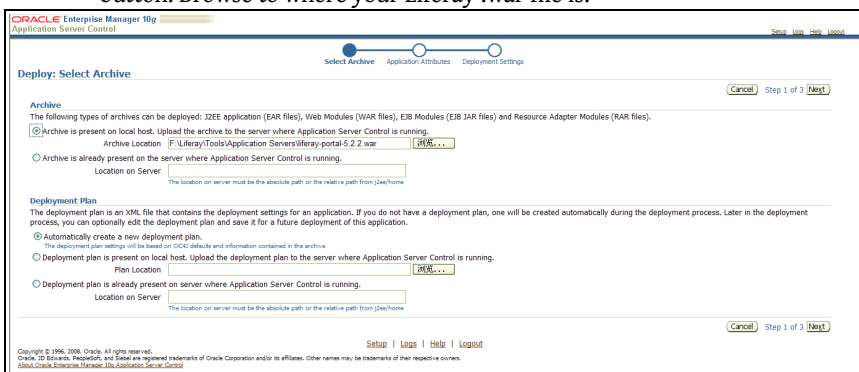


Illustration 10: Deploying the Liferay .war file on OC4J

- With the location of the Liferay .war file in the Archive Location field, click *Next*. Wait a while while OC4J uploads the Liferay .war file.

4. The next screen allows you to give the application a name and set its context root. Use *Liferay* for the name and */portal* as the context root. Click *Next*.
5. Click the *Next* link at the bottom right of the page.
6. Click the *OK* button at the top of this page. You will be brought back to the previous screen. Click the *Deploy* button at the top right of the screen. OC4J will then deploy Liferay.

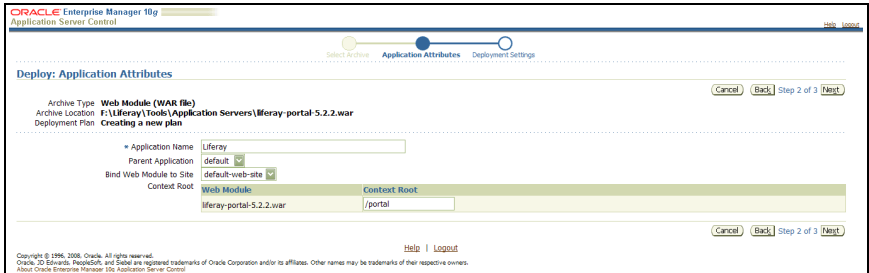


Illustration 11: Setting the deployment options on OC4J

7. Click the *Return* button at the top of the page.
8. Next, navigate to the Liferay Home folder. Create a file called *portal-ext.properties* in this folder. Add a directive for Portal Context, then give it a value of */portal*.

```
portal.ctx=/portal
```

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/jettyportal?
useUnicode=true&characterEncoding=UTF-8&useFastDateParsing=false
jdbc.default.username=root
jdbc.default.password=root
```

If you are using OC4J'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 OC4J's mail session, add the JNDI name instead:

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

Save and close the file.

9. Shut down and restart the server. In your browser, go to <http://localhost:8888/portal>. You should see Liferay's default home page.

```
09/04/28 15:06:54 Loading code-source:/F:/Liferay/Tools/Application Servers/oc4j_extended_101340/j2ee/home/applications/Liferay/liferay-portal-5.2.2/WEB-INF/lib/portal-impl.jar!/system.properties
09/04/28 15:06:55 Loading code-source:/F:/Liferay/Tools/Application Servers/oc4j_extended_101340/j2ee/home/applications/Liferay/liferay-portal-5.2.2/WEB-INF/lib/portal-impl.jar!/portal.properties
09/04/28 15:06:55 Loading file:/F:/Liferay/Tools/Application Servers/oc4j_extended_101340/j2ee/home/applications/Liferay/liferay-portal-5.2.2/WEB-INF/classes/portal-ext.properties
15:07:02,984 INFO [DialectDetector:64] Determining dialect for MySQL 5
15:07:03,125 INFO [DialectDetector:97] Using dialect org.hibernate.dialect.MySQLDialect
09/04/28 15:07:05 Loading code-source:/F:/Liferay/Tools/Application Servers/oc4j_extended_101340/j2ee/home/applications/Liferay/liferay-portal-5.2.2/WEB-INF/lib/portal-impl.jar!/captcha.properties
15:07:06,359 INFO [PortalImpl:243] Portal lib directory /F:/Liferay/Tools/Application Servers/oc4j_extended_101340/j2ee/home/applications/Liferay/liferay-portal-5.2.2/WEB-INF/lib/
15:07:20,531 INFO [ServerDetector:104] Detected server oc4j
2009-4-28 15:07:23 com.sun.portal.container.service.ServiceManagerContextListenerImpl contextInitialized
INFO: PSC_CSPCS001 : Started initializing ServiceManager
2009-4-28 15:07:23 com.sun.portal.container.service.ServiceManager addService
INFO: PSC_CSPCS004 : Service added, Name: com.sun.portal.container.service.CoordinationService, Implementation: com.sun.portal.container.service.coordination.impl.CoordinationServiceImpl
2009-4-28 15:07:23 com.sun.portal.container.service.ServiceManager addService
INFO: PSC_CSPCS004 : Service added, Name: com.sun.portal.container.service.CachingService, Implementation: com.sun.portal.container.service.caching.impl.CachingServiceImpl
2009-4-28 15:07:23 com.sun.portal.container.service.ServiceManager addService
INFO: PSC_CSPCS004 : Service added, Name: com.sun.portal.container.service.ClientCachingService, Implementation: com.sun.portal.container.service.caching.impl.ClientCachingServiceImpl
2009-4-28 15:07:23 com.sun.portal.container.service.ServiceManager addService
INFO: PSC_CSPCS004 : Service added, Name: com.sun.portal.container.service.DeploymentService_Local, Implementation: com.sun.portal.container.service.deployment.impl.DeploymentServiceLocalImpl
2009-4-28 15:07:23 com.sun.portal.container.service.ServiceManager addService
INFO: PSC_CSPCS004 : Service added, Name: com.sun.portal.container.service.ContainerEventService, Implementation: com.sun.portal.container.service.coordination.impl.ContainerEventServiceImpl
```

```

2009-4-28 15:07:23 com.sun.portal.container.service.ServiceManager addService
INFO: PSC_CSPCS004 : Service added, Name: com.sun.portal.container.service.PolicyService, Implementation: com.liferay.portal.portletcontainer.PolicyServiceImpl
2009-4-28 15:07:23 com.sun.portal.container.service.ServiceManager addService
2009-4-28 15:07:24 com.sun.portal.container.service.ServiceManagerContextListenerImpl contextInitialized
INFO: PSC_CSPCS002 : Finished initializing ServiceManager
2009-4-28 15:07:24 com.sun.portal.portletcontainer.impl.PortletContainerContextListenerImpl contextInitialized
INFO: PSPL_PCCSPCCI0003 : Starting PortletContainer 2.1
09/04/28 15:07:24 Starting Liferay Portal Standard Edition 5.2.2
(Augustine / Build 5202 / February 20, 2009)
15:07:53,000 INFO [HotDeployUtil:64] Initializing hot deploy manager
10845041
15:07:53,359 INFO [AutoDeployDir:108] Auto deploy scanner started for
C:\Documents and Settings\Steven\liferay\deploy
09/04/28 15:07:54 Oracle Containers for J2EE 10g (10.1.3.4.0) initialized
15:10:52,875 INFO [PluginPackageUtil:1148] Reloading repositories

```

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 60-64 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.x-
  erces.jaxp.SAXParserFactoryImpl" />
<system-property javax.xml.transform.TransformerFactory="org.a-
  pache.xalan.processor.TransformerFactoryImpl" />
<system-property org.xml.sax.driver="org.apache.xerces.parsers.SAX-
  Parser" />
```

3. Go to `$RESIN_HOME` and create new directory called `common/lib`. Download `mysql-connector-java-{version}-bin.jar` and copy to this directory. This is the JDBC connector for MySQL. If you are using another database, substitute this with the appropriate driver.
4. Download the Liferay Portal Dependencies and unzip into `$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 `$RESIN_HOME/common/lib`. Copy `saxpath.jar` and `xalan.jar` to `$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
```

- 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.

- 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 `$RESIN_HOME`.
- Edit `$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"/>
```

- 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>
```

- 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.
- Download the Liferay Portal Dependencies and unzip into `$RESIN_HOME/common/lib`.
- Delete the contents of `$RESIN_HOME/webapps/ROOT`.
- Unzip `liferay-portal-x.x.x.war` to `$RESIN_HOME/webapps/ROOT`.
- Next, you will need several `.jar` files which are included as part of the Liferay source distribution. Many application servers ship with these already on the class path, but Resin does not. The best way to get the appropriate versions of these files is to download the Liferay source code and get them from there. Once you have downloaded the Liferay source, unzip it to a temporary folder.
 - Go to `$LIFERAY_SOURCE/lib/development/` Copy `saxpath.jar` to `$RES-`

`IN_HOME/common/lib.`

9. To start the server, open a command prompt, navigate to the `$RESIN_HOME` and type:
10. `java -jar lib/resin.jar start`

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

TOMCAT 5.5.X

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

1. Download and install Tomcat 5.5.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 `$TOMCAT_HOME/common/lib/ext` directory and unzip the dependencies ZIP in there. If the files do not extract to this directory, make sure they are in the correct directory by moving them there.
5. Edit `$TOMCAT_HOME/conf/catalina.properties`:

```
common.loader=
    ${catalina.home}/common/classes,\
    ... \
    ${catalina.home}/common/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 refer-

ence 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" disableUp-
             loadTimeout="true"
             URIEncoding="UTF-8"
  />
```

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

WEBLOGIC 9 / 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 `$WEBLOGIC-HOME/jrocket90_150_04/jre/lib`, then copy `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*.



Illustration 12: WebLogic: Data Sources

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 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.

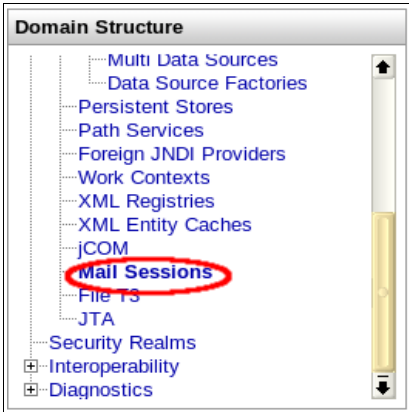


Illustration 13: WebLogic: Mail Sessions

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.
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:

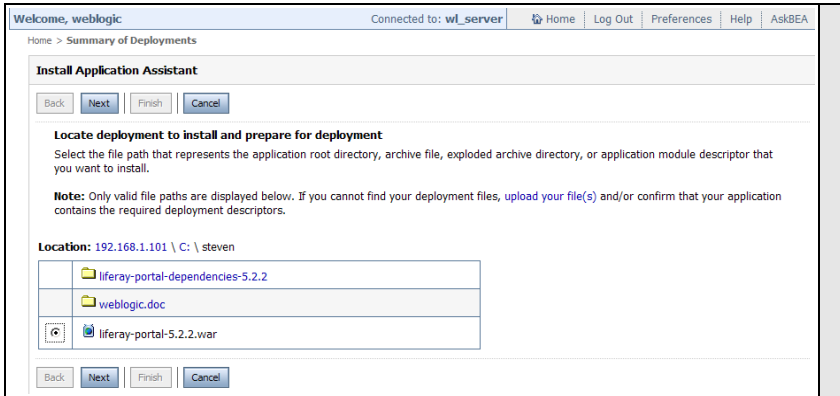


Illustration 14: Deploying Liferay on WebLogic 9.2

```
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.

Copy the dependency jars and the JDBC driver to the `domain/lib` directory.

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.

Illustration 15: 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. Select *Deployments* and click the *Install* button. Select the Liferay .war file from the file system or click the *Upload Your File(s)* link to upload it, and then click *Next*.
3. Leave *Install this deployment as an application* selected and click *Next*.
4. If the default name is appropriate for your installation, keep it. Otherwise, give it a name of your choosing and click *Next*.
5. 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 Dependen-

cies 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*.

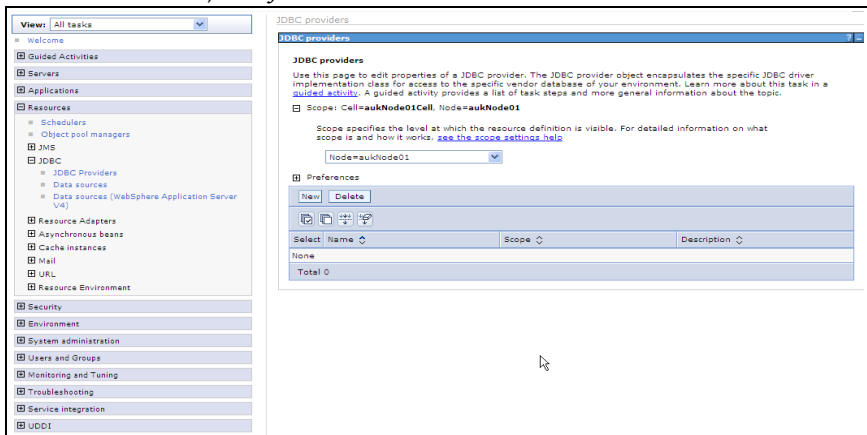


Illustration 16: WebSphere 6.1 JDBC Providers

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*.

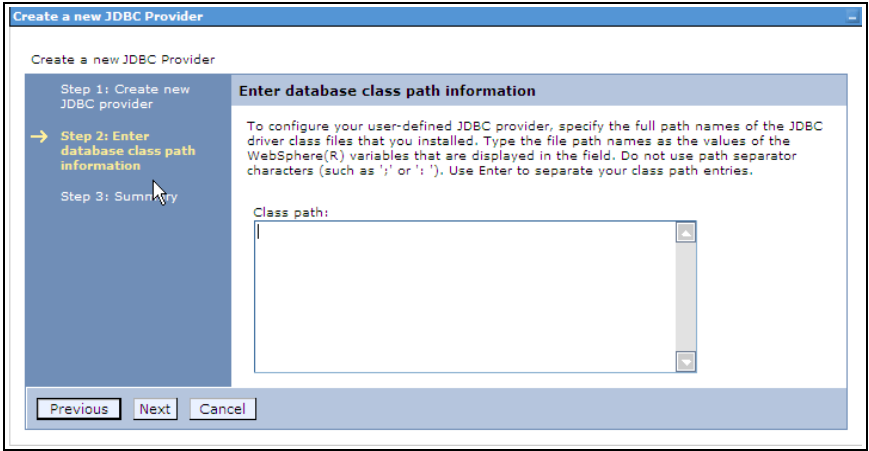


Illustration 18: Clear out anything in the class path.



Illustration 17: WebSphere 6.1 New JDBC Provider

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*.

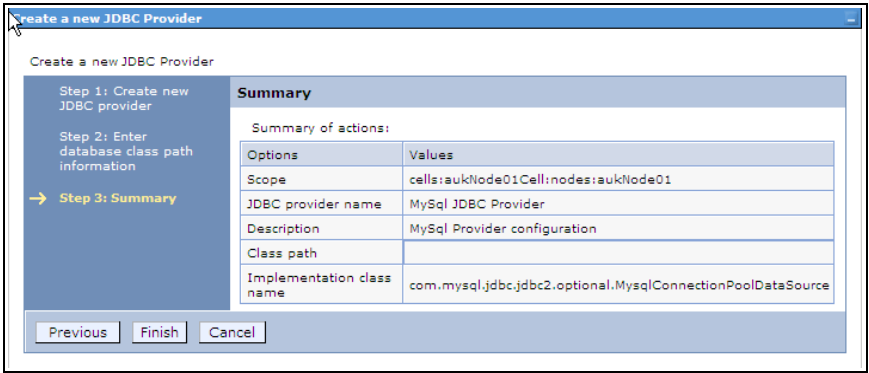


Illustration 19: JDBC Provider summary screen

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.

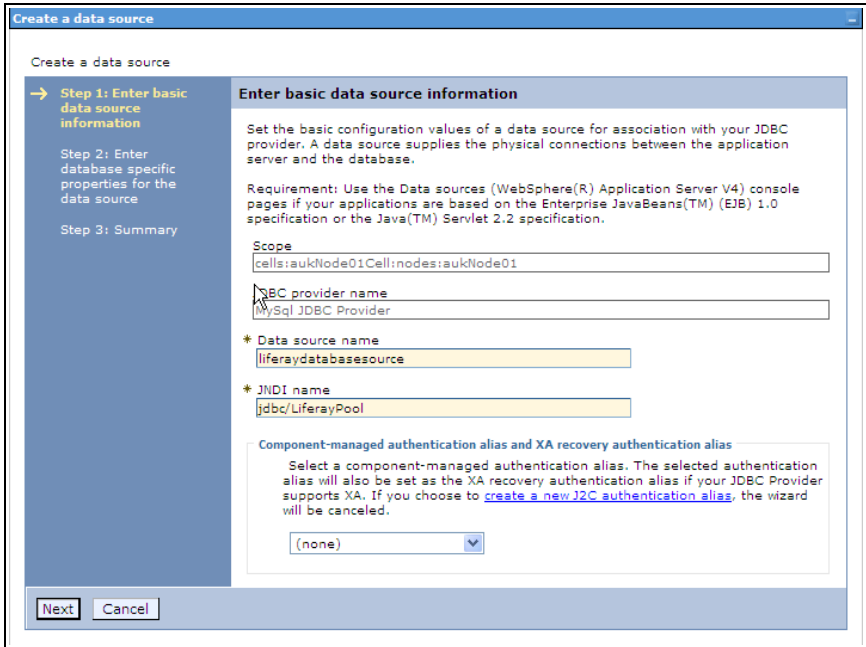


Illustration 20: Liferay data source on WebSphere 6.1

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

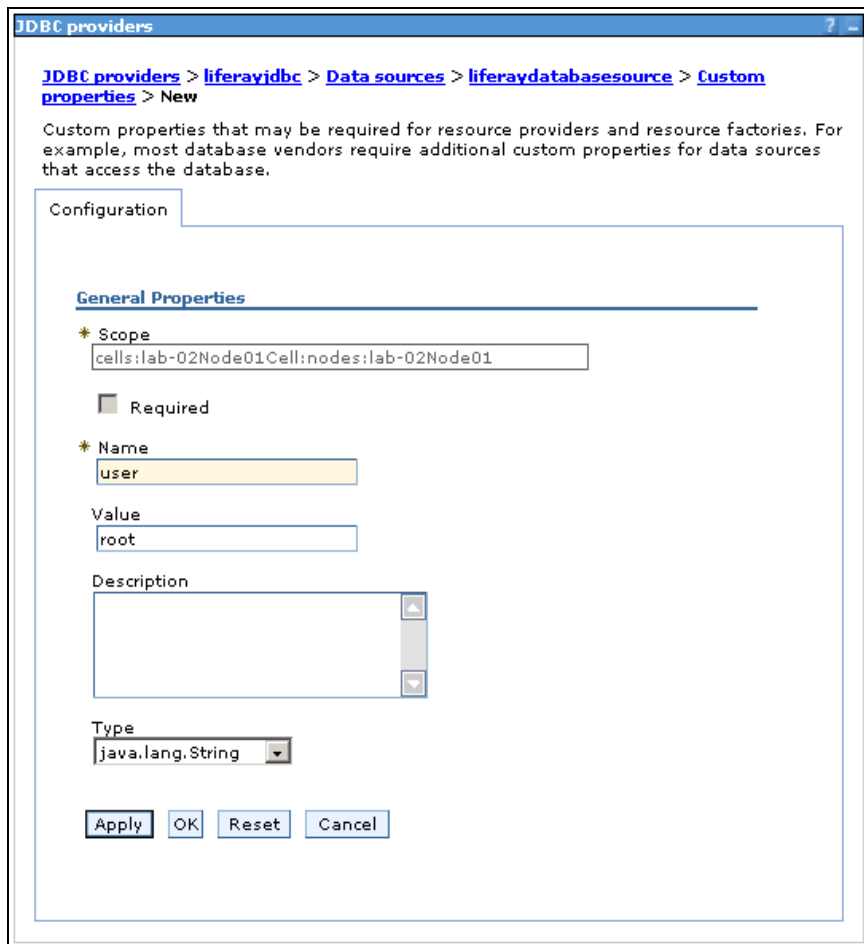


Illustration 21: WebSphere 6.1 Custom Properties Entry

20. When done correctly, custom properties should look like this:

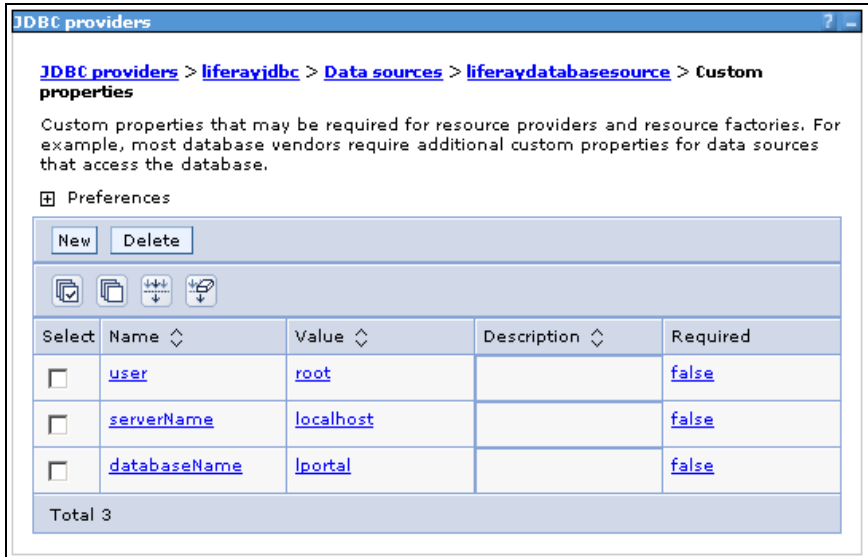


Illustration 22: Liferay Custom Properties

21. 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*.
 1. **Name:** *liferaymail*
 2. **JNDI Name:** *mail/MailSession*

[Mail Providers](#) > [Built-in Mail Provider](#) > [Mail Sessions](#) > liferaymail

Use this page to create mail sessions, which are collections of properties that define how your application sends mail and accesses the mail store. Configure mail sessions only after you configure the necessary protocol providers.

Configuration

General Properties	Additional Properties
---------------------------	------------------------------

* Scope

* Provider

* Name

* JNDI name

Description

Category

Mail transport host

Mail transport protocol

Mail transport user ID

Mail transport password

Enable strict Internet address parsing

Mail from

Mail store host

Mail store protocol

Mail store user ID

Mail store password

Enable debug mode

Illustration 23: Creating a Mail Session on WebSphere 6.1

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*.

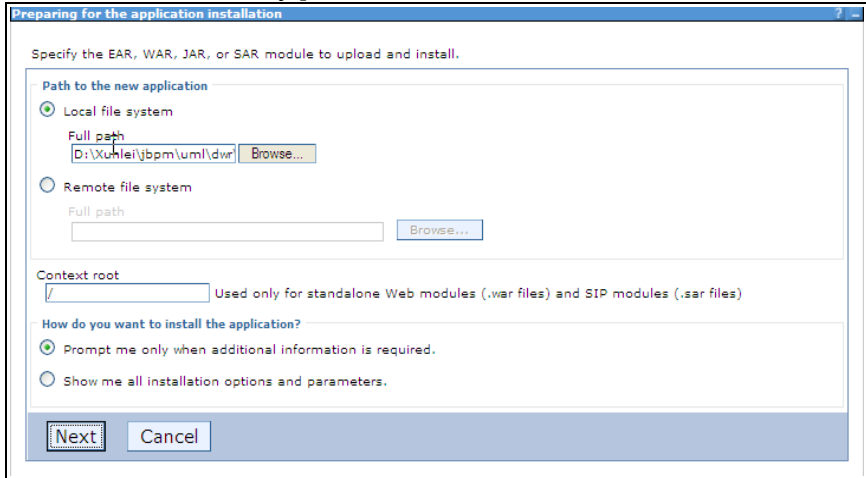


Illustration 24: Installing the Liferay .war file on WebSphere 6.1

3. Enter context root `/`.
4. Click *Next*. For Steps 1 to 3, click *Next* to apply defaults.

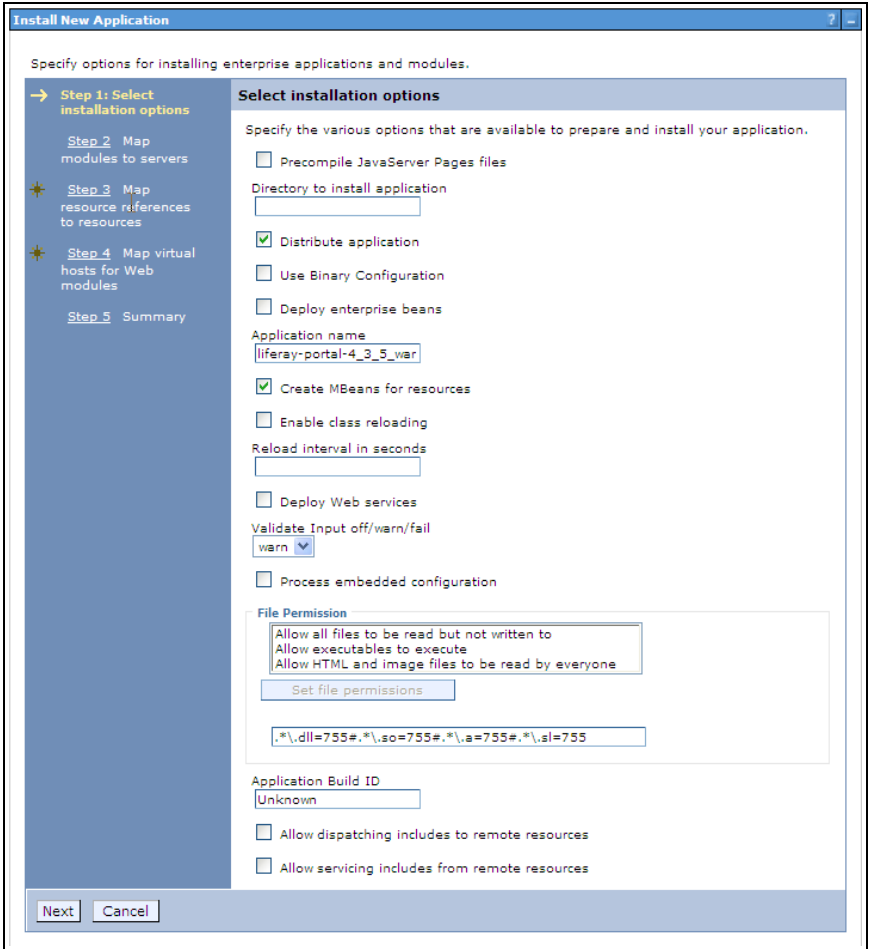


Illustration 25: Select the installation options

5. Choose the Mail Session and Data Source, and then click **Next**
6. Specify the virtual host upon which you want Liferay to run.

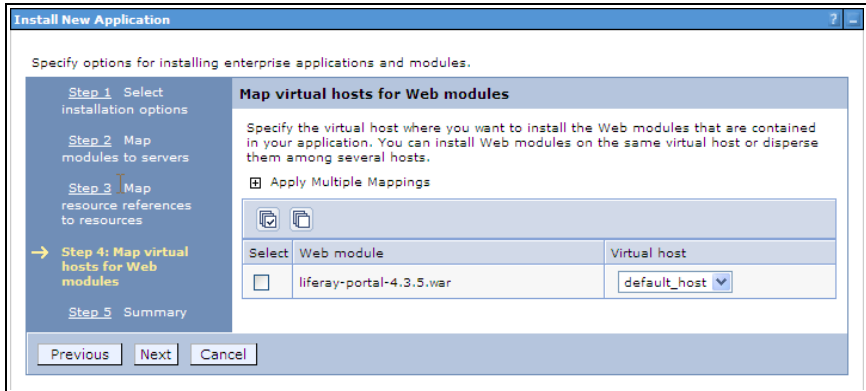


Illustration 26: Select the virtual host

7. At the Summary Screen, click *Finish*.

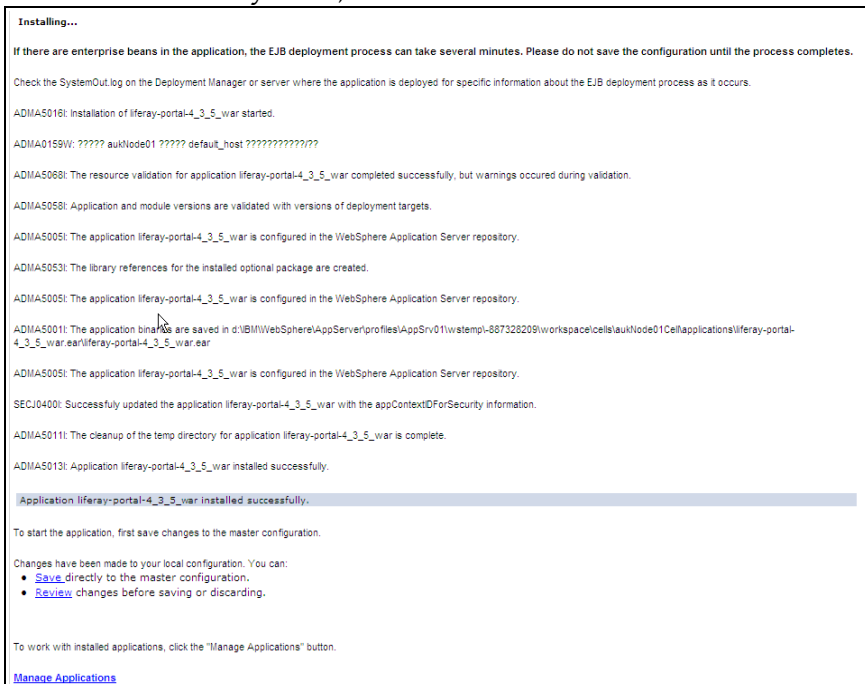


Illustration 27: WebSphere 6.1 installation process

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
```

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*.

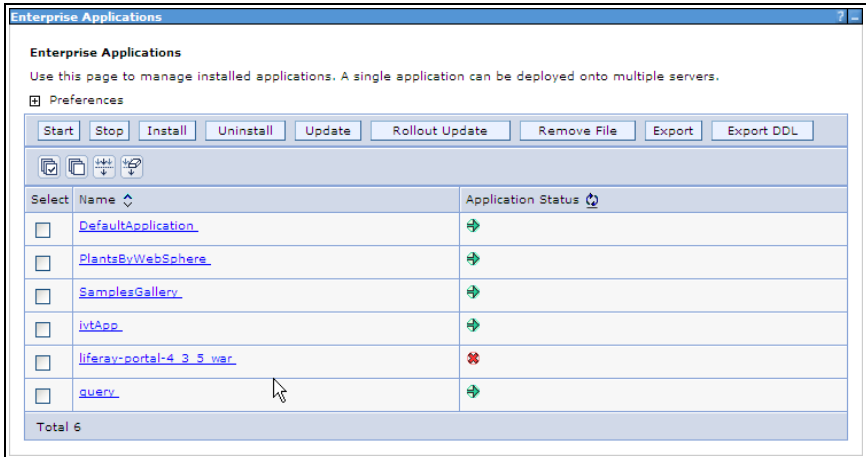


Illustration 28: Starting Liferay on WebSphere 6.1

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

WEBSHERE 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 *websphere/appserver/lib/ext*.
 1. Dependency libraries (Liferay Portal Dependencies)
 2. JDBC Driver for your database

DATABASE CONFIGURATION

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

1. Start WebSphere.

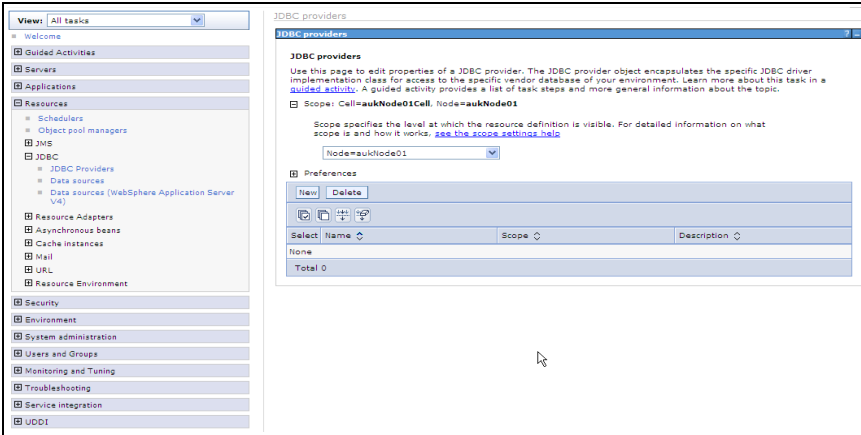


Illustration 29: WebSphere 7.0 JDBC Providers

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

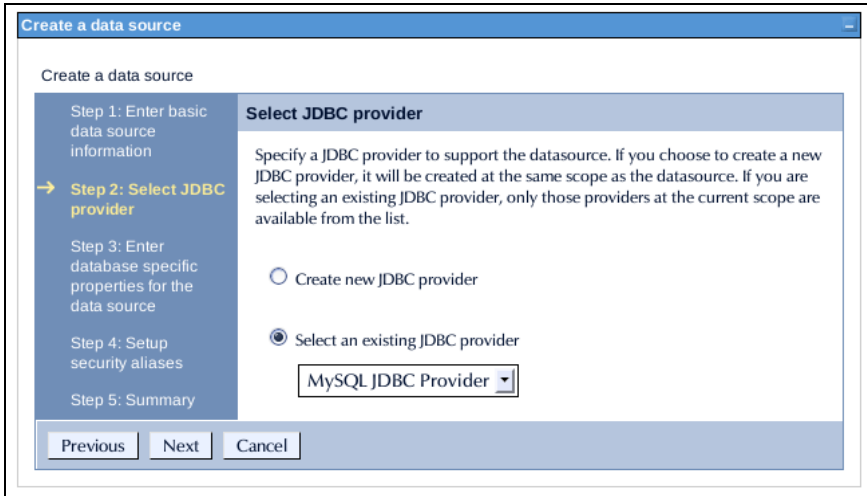


Illustration 30: WebSphere 7.0: Select JDBC Provider

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*.

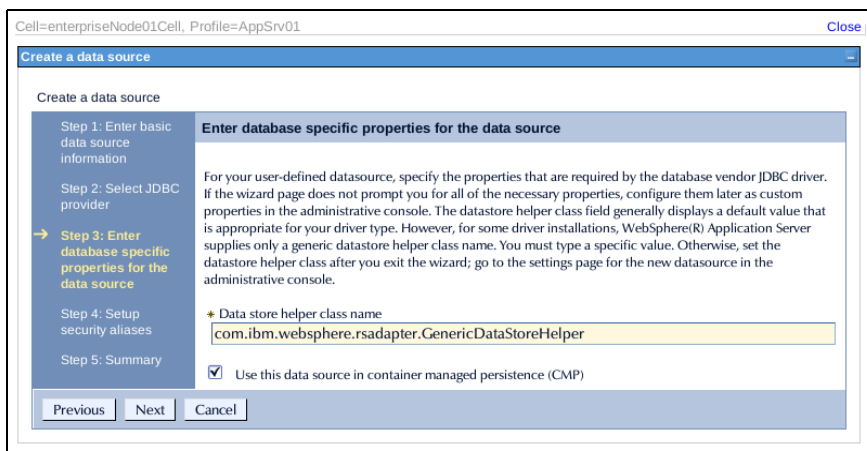


Illustration 31: WebSphere 7.0: Database Properties

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*.

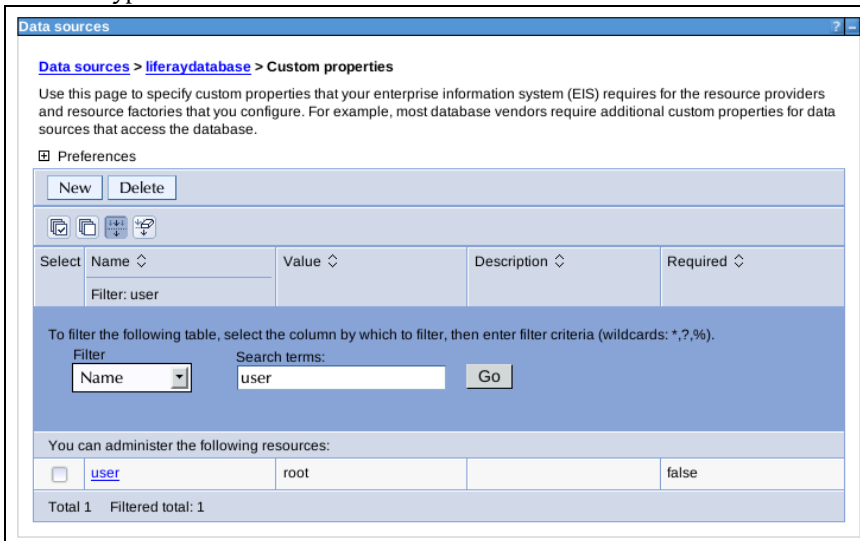


Illustration 32: Modifying data source properties in WebSphere 7

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:

```
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 deployed 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 4.

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. The choice is yours: Liferay Portal won't get in your way, and you can feel safe knowing that you have the freedom to use the software that is best for your organization.

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, 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. The heart of any portal-based web site is its portlets, because portlets are where all of the functionality is implemented. 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 the *Liferay Developer's Guide*.

Additionally, Liferay supports portlets written in other programming languages, such as PHP, Ruby, Groovy, or Python. Sample portlets written in these languages are available to download from our Sourceforge site (<http://sourceforge.net/projects/lportal>) or can be checked out from our Subversion repository (<https://lportal.svn.sourceforge.net/svnroot/lportal>).

Navigating Liferay

When you see Liferay's default interface for the first time, you will see what we call the Dock in the upper right hand corner of the screen. The Dock is the key to navigating within the supplied Liferay themes: it provides links to all the global functions a user needs, such as logging in and out and switching between various community and organization pages. By default, it contains only two links: *Home* and *Sign In*. To show these links, all you need to do is roll your mouse cursor over the Dock, and it will expand.

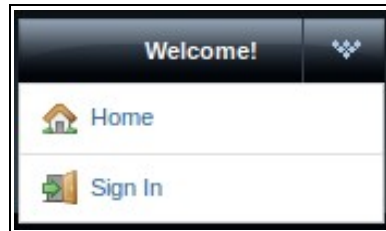


Illustration 33: The default Liferay Dock.

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 account 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

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 use 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 little has changed. The Sign In portlet now displays the name of the user (whose rather unimaginative name is Test Test) who is logged in, and the Dock now displays the text "Welcome, Test Test!" If you hover your mouse over the dock now, however, you will see that there are many more links in it:

Home: takes you to the home page.

Control Panel: takes you to the administrative functions of Liferay.

My Account: lets you edit your user information.

Sign Out: logs you out of the portal.

Add Application: opens the Add Application window which allows you to add portlets to the page.

Layout Template: displays the Layout Template window which allows you to choose different layouts for the page.

Manage Pages: takes you to the page administration screen, where you can add and delete pages, change the order of the pages, and configure several other things about pages.

Toggle Edit Controls: This will let 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.

My Places: shows you the community and organization pages in the portal to

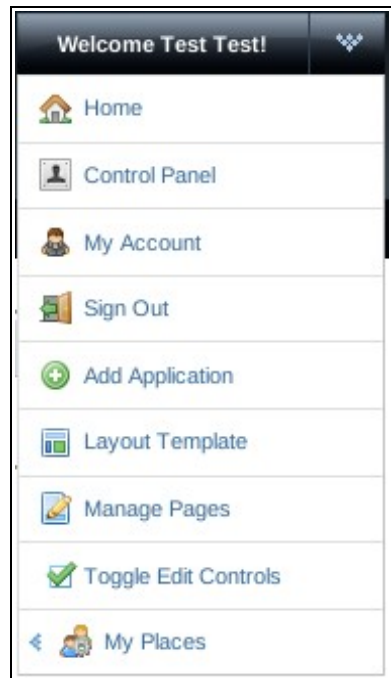


Illustration 34: Administrator's Dock

which you can navigate.

If you roll your mouse over *My Places*, the Dock will expand, showing all of the places in the portal to which you have access. Initially, the place you are on will be highlighted. You will see that you are in the Guest 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.

New in Liferay 5.2 is the *Control Panel*, a new interface which provides a single place to go for all of Liferay's administrative functions. In versions of Liferay prior to 5.2, you needed to add the administrative portlets to a page somewhere, and a good place to do that was on the Administrator's private pages. Now all you need to do is go up to the Dock and select *Control Panel*. This is a much easier and faster way to access Liferay's administrative functions and will be the user interface going forward.

With that said, the administrative portlets are still available, and you can still provide access to them if you wish.

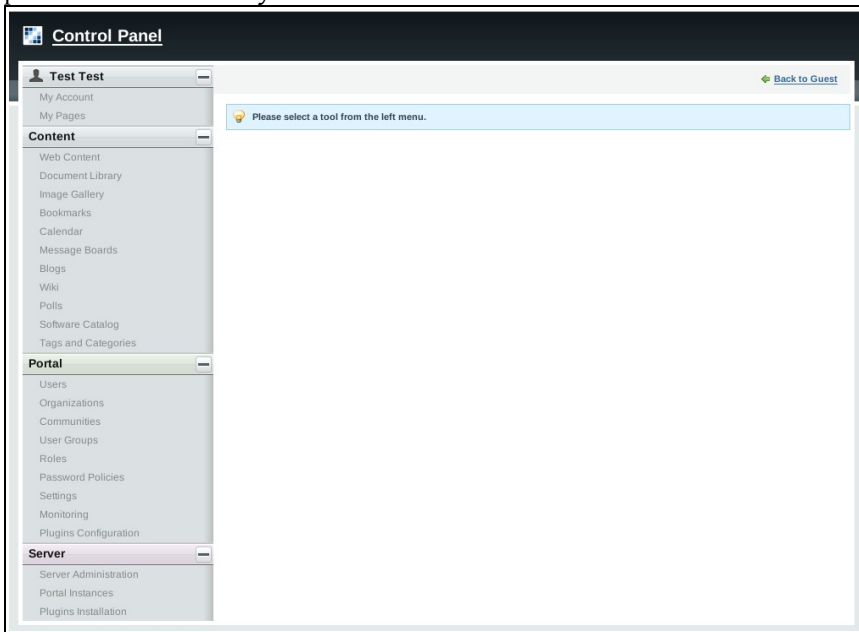


Illustration 35: Liferay's Control Panel

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.

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.

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 information about this, you can take Liferay's *Portal Developer* course or see the *Liferay Developer's Guide*.

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.

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 in the following manner:

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 collec-

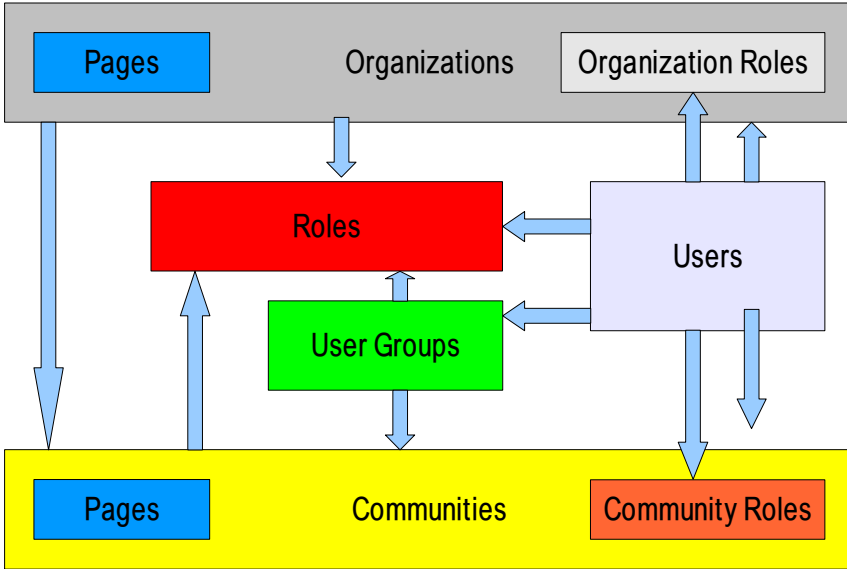


Illustration 36: Liferay portal resources

tions. Permissions do not flow through all of these collections: permissions can be assigned to roles only.

Users

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 would enable them to create blog entries in their personal space. They can be members of communities which draw together common interests. And they can have roles which describe their functions 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 scope: 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 in the Guest community, 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 to which they have 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.

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.

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 Dock and click the *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*.

Illustration 37: The Add User screen.

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.

If you have not yet set up your mail server, you will 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

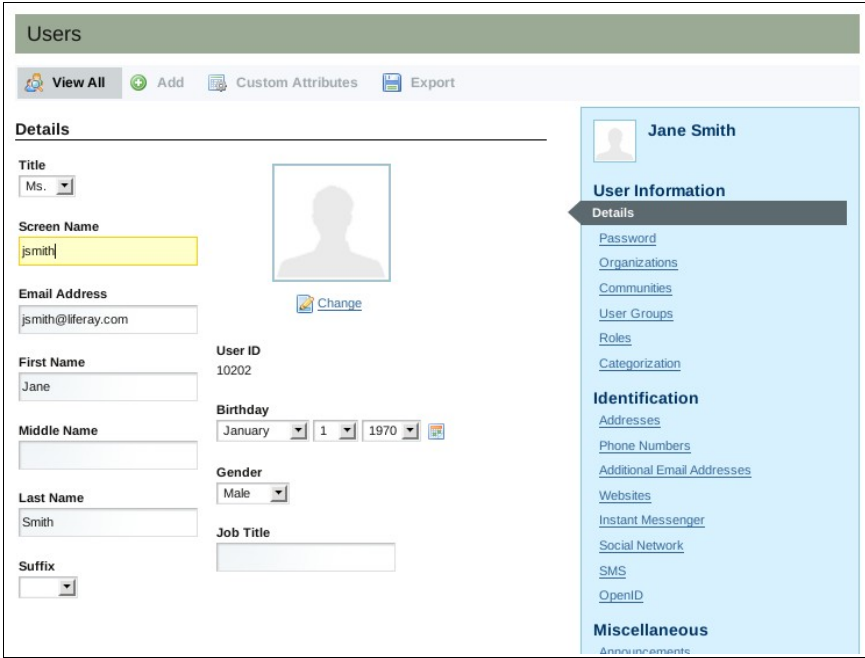


Illustration 38: Liferay's User Account editor.

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: This 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). Because 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 have a hierarchy to unlimited levels, and Users can be members of one or many organizations—inside of a hierarchy or across 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 User Groups 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.

Name: The name of the organization.

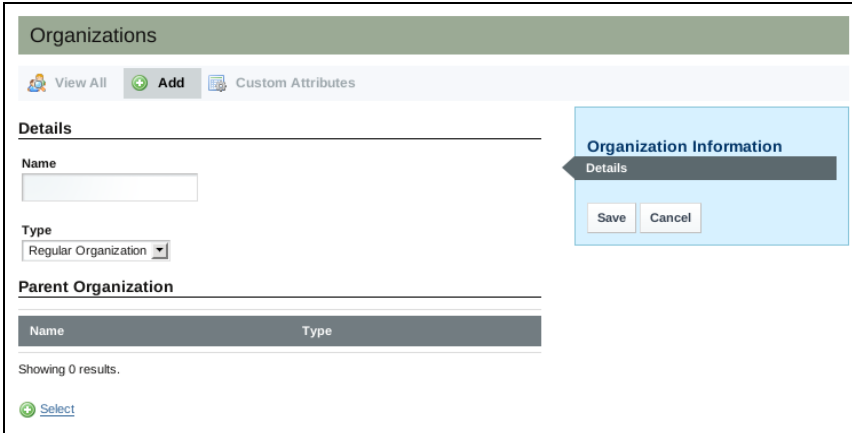


Illustration 39: Adding an organization.

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 addresses, 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 *View All* 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.

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 Suborganizations: 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.



Tip: Note that you are already a member of the organization you created, because you created it. By creating an organization, you become both a member and have the Organization Owner role, which gives you full rights to the organization.

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 called Guest, 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 a distinct role within Liferay Portal.

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 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.

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.

COMMUNITY PROVISIONING

Communities are ideal workspaces for teams to collaborate on common projects. They provide an isolated area where a group of people can place all of their data pertaining to a particular topic, and many organizations have used them for this purpose. It is a far better way to share data than using email and a shared folder on a net-

work. Instead, Liferay's Document Library portlet empowers users to access and update documents simultaneously, and all versions of the documents are preserved. A Calendar portlet can be used to keep track of the team's appointments and meetings, and can send notifications out to the team. A Wiki portlet can be used to document the project as it progresses. A Message Boards portlet can be used to keep all team discussions in one place.

To enable the ad-hoc creation of communities for this purpose, Liferay 5.2.3 and above allows portal administrators to create communities based on templates. 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.

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 the default template.

Go to the Control Panel and click *Communities*. Click the *Add* button and create a community called *DEFAULT_TEMPLATE*. Make it a private community. Once the community has been created, click *Actions* → *Manage Pages*, and then click the *Settings* tab. Select *Activate Staging*.

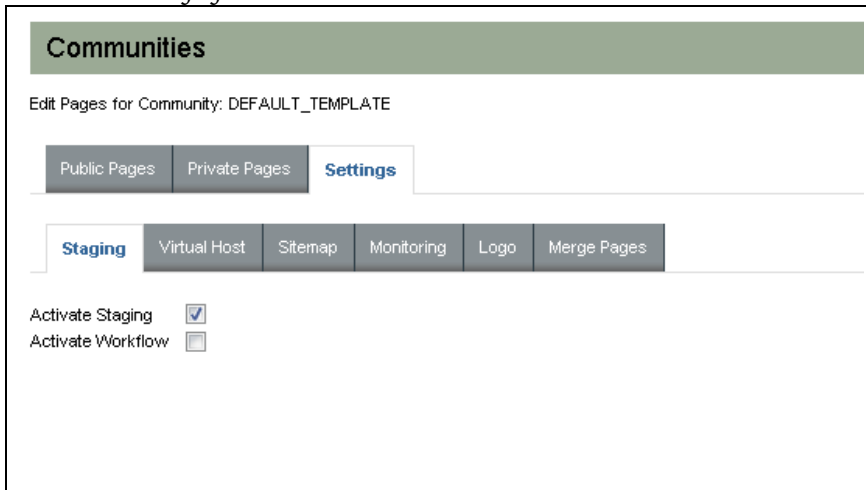


Illustration 40: Activating Staging on the community template.

Now go ahead and use the *Public Pages* or *Private Pages* tabs to create pages in the template and populate them with portlets. You can always use the *View Pages* button to view the pages you are creating, to add portlets, and to change layouts. These pages will be surrounded by a red border. This indicates that you are working in a staging environment.

That's all you need to do. Don't publish the changes to the template; leave it in staging mode. Any future communities that you create will have the same layouts and portlets as what you have placed in the *DEFAULT_TEMPLATE*.

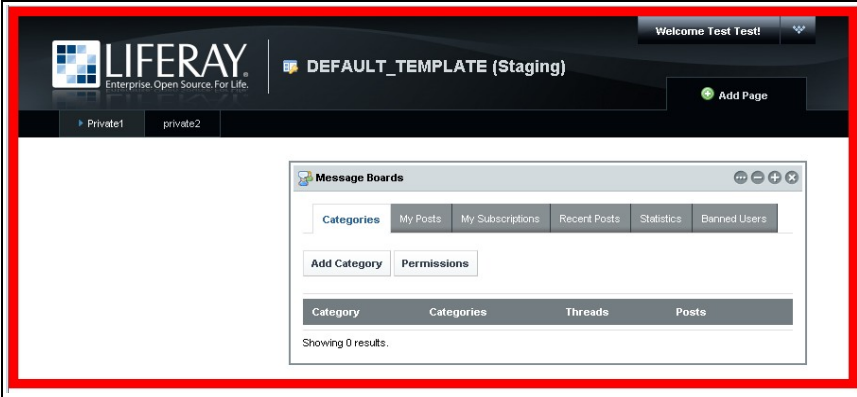


Illustration 41: Populating pages in the community template

To further refine this, you can create other kinds of templates for specific types of communities in the same manner.

If community is open, use the name `OPEN_TEMPLATE` to create a template for open communities. Use the name `RESTRICTED_TEMPLATE` to create a template for restricted communities. Use the name `PRIVATE_TEMPLATE` to create a template for private communities. The `DEFAULT_TEMPLATE` we have just created applies to all types of communities. If you want all newly created communities to have a Forums page with a Message Boards portlet on it, you would create that in the `DEFAULT_TEMPLATE`.

This feature streamlines the community creation process for administrators, making it very easy to quickly create communities for teams.

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 cre-

ate *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

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

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

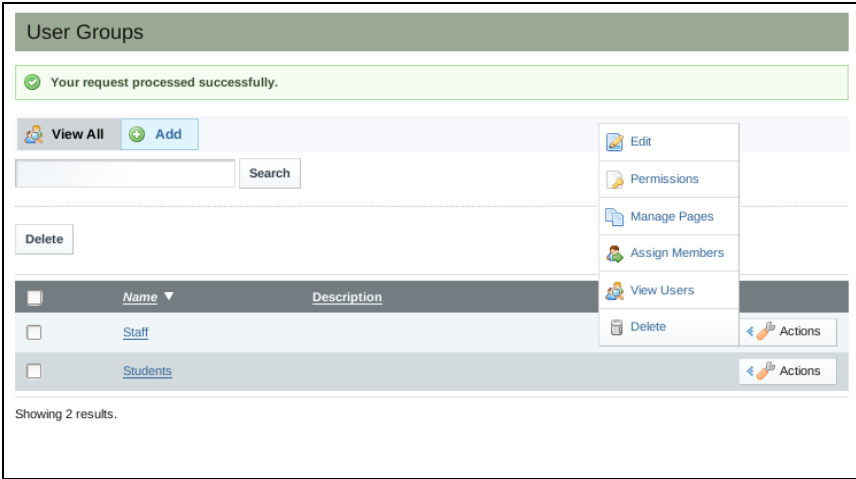


Illustration 42: Manage Pages action on a User Group.

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.

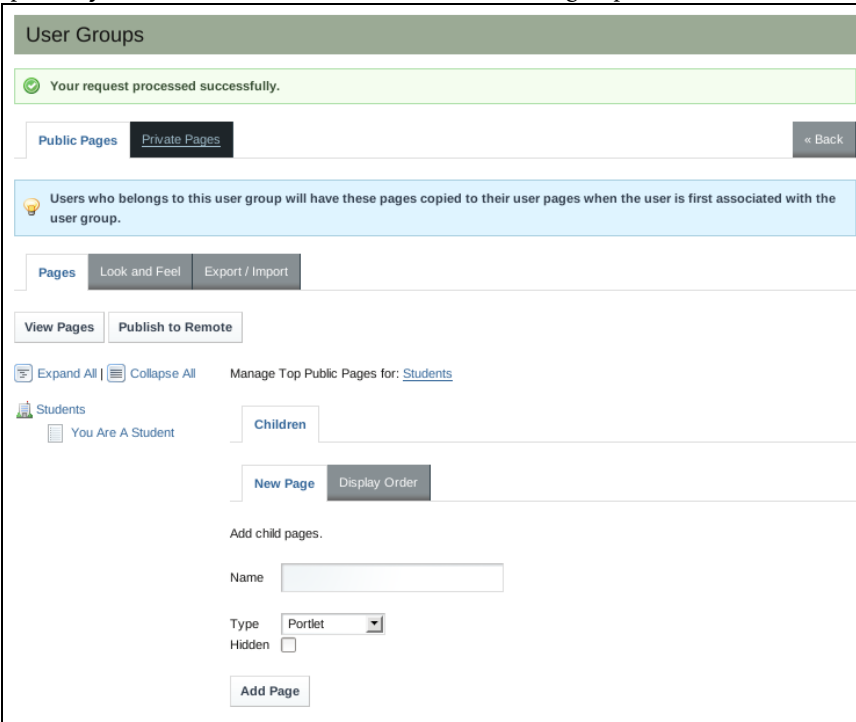


Illustration 43: Adding a Page Template

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 will be 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* action in the list of available user groups.

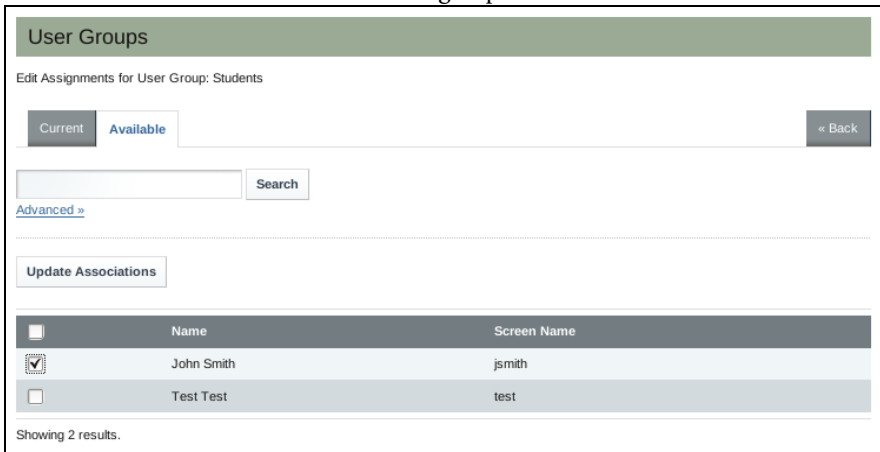


Illustration 44: Assigning Members to a User Group

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:

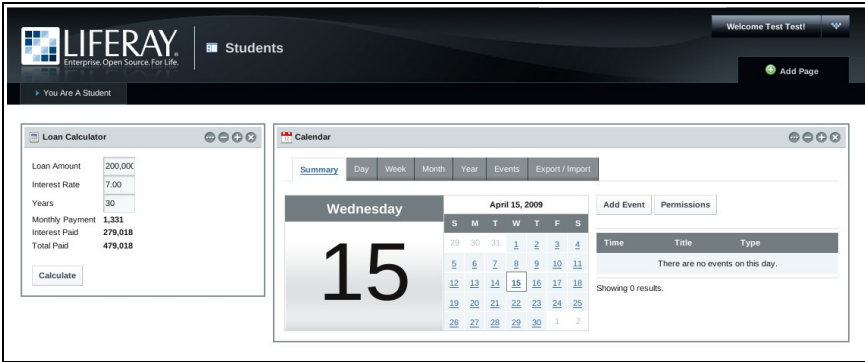


Illustration 45: Template copied to a user's page set.

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.

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 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 them 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:

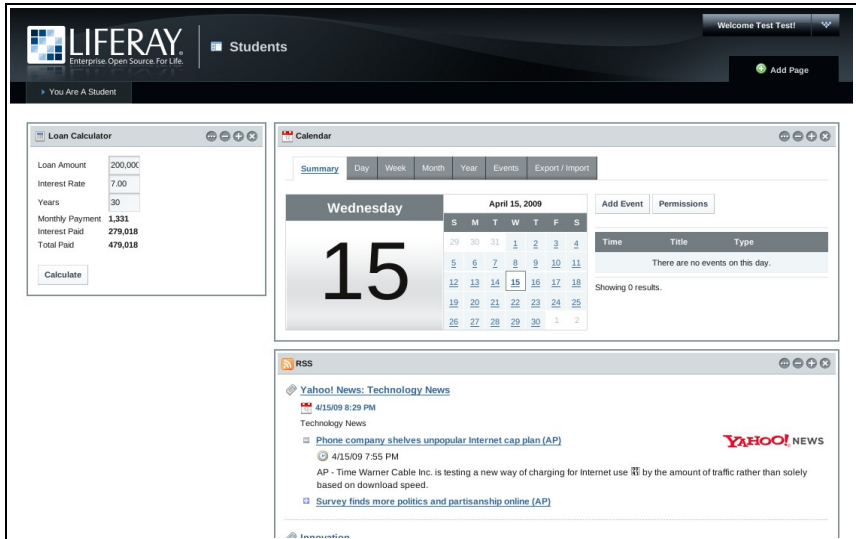


Illustration 46: Combined portlet pages.

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.

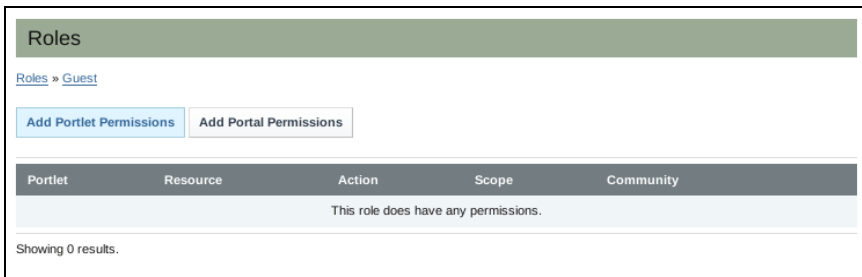


Illustration 47: Defining Permissions on a Role.

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.

When you click the *Define Permissions* action on a Portal scoped Role, you are given a choice of two kinds of permissions that can be defined for this role: *Portal Permissions* and *Portlet Permissions*. For other Roles, you only see the portlet permissions.

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.

Portlet permissions cover permissions that are defined within various portlets. Clicking the *Portlet Permissions* button brings you to a page where you can browse the names of the portlets that are currently installed in your portal. Once you choose a portlet, you can then define the actions within this portlet that the role will have permission to perform.

If we stick with our example of a *Message Boards Admin* role, we would then find the **Message Boards** portlet in the list and click on it. A new page with configurable permissions would then be displayed (see right).

Each possible action to which permissions can be granted is listed. To grant a permission, choose the scope of the permission. You have two choices: *Enterprise* and *Communities*. Granting Enterprise permissions means that permission to the action will be granted across the portal, in any community or organization where there is a Message Boards portlet.

If you choose Communities, a button appears next to the permission allowing you to choose one or more communities in which the permission will be valid. This lets you pick and choose specific communities (for a portal scoped role) in which these permissions are valid for users in this role.

Once you have chosen the permissions granted to this role, click *Save*. For a Message Boards Admin role, you would likely grant Enterprise 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 (by clicking *Add Portlet Permissions* or *Add Portal 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.

The screenshot shows a web interface for configuring roles. At the top, a green header reads "Roles". Below it, a breadcrumb trail says "Step 2 of 2. Select the scope of the action that this role can". A secondary breadcrumb trail reads "Roles » Guest » Message Boards".

The main content area is divided into four sections, each with a title and a table of actions and scopes:

- Message Boards**: A table with two rows. The first row has "Action" "Configuration" and "Scope" (dropdown). The second row has "Action" "View" and "Scope" (dropdown).
- Message Boards**: A table with two rows. The first row has "Action" "Add Category" and "Scope" (dropdown). The second row has "Action" "Ban User" and "Scope" (dropdown).
- Message Boards Category**: A table with two columns: "Action" and "Scope". There are ten rows, each with an action name and a dropdown menu for the scope.
- Message Boards Message**: A table with two columns: "Action" and "Scope". There are five rows, each with an action name and a dropdown menu for the scope.

Illustration 48: Message Boards permissions.

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 (see below on page 102), you will need to modify the permissions on certain portlets. To do this, see the section on Plugins Configuration below (page 114).

Global Server Settings

Now that you have navigated in the Control Panel, you should be pretty familiar with how it works, and hopefully now you are comfortable exploring around the various options on the left. 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 and select the 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.

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.

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.

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.

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 login IDs, connection to LDAP, and Single Sign-On.

Default User Associations: Lets you configure default membership to Roles, User Groups, and Communities for new users.

Reserved Credentials: 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 *president*.

Mail Host Names: You can add a list of other mail servers besides your main one 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. 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 login. If enabled, Liferay allows 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.
- Enable / Disable Forgot Password functionality.
- 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).
- 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 the email address they entered is a valid one they can access.

By default, all settings except for the last are enabled by default. 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, if it is not used to authenticate, it is more likely that the user will forget to update his or her email address in his or her profile. 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 has become much easier and is now 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 very detailed, so we will look at it in chunks.

GLOBAL VALUES

There are two global LDAP settings.

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 that have Liferay accounts but no LDAP accounts to log in to the portal.

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 certi-

ificate:

On the Windows 2003 Domain Controller, open the *Certificates MMC* snapin. 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 logins. By default, Liferay uses the email address as a user login 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 login 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
- First Name
- Last Name

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.

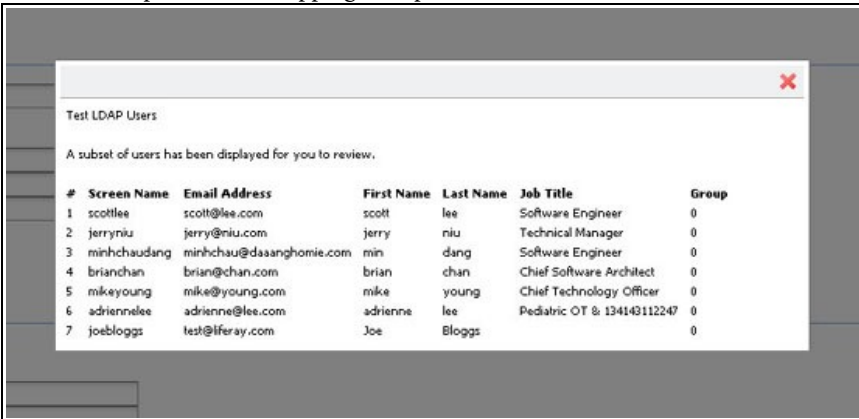


Illustration 49: Testing LDAP Users

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.

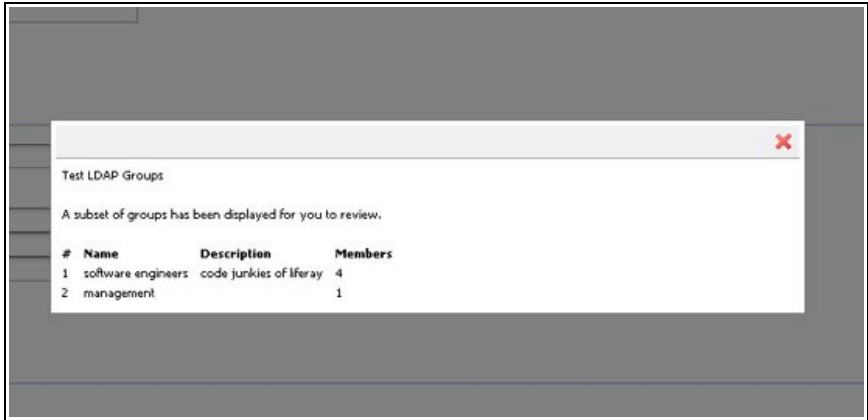


Illustration 50: Mapping LDAP Groups.

IMPORT/EXPORT

Import Enabled: Check this box to cause Liferay to do a mass import from your LDAP directory. 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, several other options will become 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.

Import Interval: The import runs on a schedule. Select how often you want the import to run. This selection 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 login, 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
```

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.

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. You can use the version that Liferay provides on our web site or the official version from the JA-SIG web site. The reason for the difference is simply that newer versions require JDK 5 and above only. We provide the older ones for use with Liferay 4.4.2 and below, so that users can have a full environment that runs on JDK 1.4.



Tip: If you are using Liferay 5.x, use the newer versions of CAS from the JA-SIG web site. If you are using Liferay 4.x, use the older Yale versions of CAS from Liferay's Sourceforge archives.

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 -keypass 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 -keypass 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% -keypass 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 finished, click *Save*. After this, when users click the *Sign In* link from the Dock, they will be directed to the CAS server to sign in to Liferay.

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.

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 OpenID4-Java (<http://code.google.com/p/openid4java/>) to implement the feature.

OpenID is enabled by default in Liferay, but can be disabled on this tab.

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: OPENSZO

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*.

DEFAULT USER ASSOCIATIONS

The *Default User Associations* link 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 used to allow users to have their own set of pages where they can add their own portlets, such as blogs, mail, or calendar. This is now provided to all users and can be modified using the *portal-ext.properties* file. You can now use the *Power Users* role to provide your own differentiation between regular users and users to whom you wish to give more privileges in the portal, or you can remove it altogether.

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.

RESERVED CREDENTIALS

The next link 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.

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. Liferay will fail over to these host names in the event that the connection to the main one fails.

EMAIL NOTIFICATIONS

There are three tabs under the *Email Notifications* link. The *Sender* tab allows you to set the portal administration 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 two tabs (*Account Created Notification* and *Password Changed Notification*) allow you to customize the email messages that are sent to users on those two 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

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.

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 has tabs for each kind of plugin you can install in Liferay. You can use these tabs to configure what portal roles have access to the plugins. For example, if you wanted to limit the use of the Message Boards portlet to just Power Users, you could use the *Portlets* tab here and remove the Users role and leave only the Power Users role in the field. To do this, simply click on the portlet you want

and enter the role names in the field provided. Any roles you enter here will be able to use this portlet.

Note that this is for basic configuration and security. If you want to disable or enable access to certain portlets completely, this is where you'd do that. For more fine-grained permissions, use the Roles section of the Control Panel and the *Actions* → *Define Permissions* button to configure more specific permissions.

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 caches:** You can send in a request to the JVM to clear a single VM cache, the cluster cache, or the database cache.
- **Reindex:** You can send in a request to regenerate all search indexes. If you are not using a Solr search server (see Chapter 6 for further information on that), 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.

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, as by modifying them too high in the hierarchy you can generate a lot more log messages than you probably need.

SYSTEM PROPERTIES

This 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.

PORTAL PROPERTIES

This 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.

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.

OPENOFFICE

Liferay Portal contains a JSR-170 compliant document repository. This repository allows users to upload documents in many formats into a folder structure that they define.

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's Document Library portlet can make use of this feature to automatically convert documents on the fly.

You would 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;" -nofirst-startwizard
```

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*.

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 the next chapter.

Summary

This chapter has described the resources in Liferay Portal that can be configured to build the site you need to build. 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 email 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 should help to bring you well on your way to becoming a seasoned Liferay Portal Administrator.

4. 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

New in Liferay 5.2 is the concept of *Scopes* as it applies to portlets instead of roles. 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 regular 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.

In a similar fashion to this, some Liferay portlets can now 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.

First of all, what do we mean by the word *scope*? Scopes are another term for a set of data that is isolated from another set of data stored in the portal database. For example, if you place a Message Boards portlet on two different communities, each Message Board has its own set of data. That is how there can be two, twenty, or 20,000 different message boards in the portal, because each has its own *scope* of data.

In previous versions of Liferay, scopes were hard-coded to be limited to organizations and communities. If you have what we would call a *non-instanceable* portlet, that's another way of saying the portlet is *scopable*. Yes, neither of those are real words, but they are an attempt to communicate the fact that each instance of the portlet has its own data set, and that data set is limited by a scope that is defined as belonging to a community or organization.

In Liferay 5.2, you can now set the scope to be for any page. This allows you to add any number of these portlets to a community or organization, and as long as they appear on different pages, they will have different sets of data. This allows you to have more than one message board per community or organization if you wish.

Unless otherwise noted, all of the portlets in this chapter support scopes. 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 one with the three dots).
2. Select *Configuration*.
3. Select the *Scope* tab.
4. Modify the Scope to be the current page.
5. Click *Save*.

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.

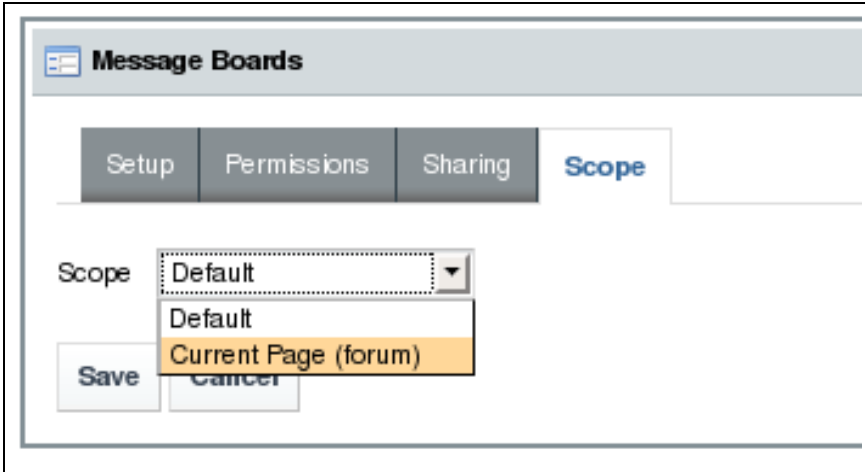


Illustration 51: Changing the scope for a portlet.

Archived Setups

Another new feature for Liferay 5.2 is Archived Setups. This means that once you configure a portlet, you can save those settings in an “archive” that the portal can keep track of. If someone then 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. If your needs are more basic, the portlets also support the standard JSR-286 permissions which define whether or not a user can see a portlet. To configure these permissions, go to the *Configuration* menu, and you will see a tab called *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*. All Java standard portlets support these permissions.

Sharing

As the concept of the web as an application itself rather than a number of islands of applications takes hold in the mindshare of the Internet's users, widgets have become very popular nowadays. This concept is sometimes referred to as "Web 2.0" and is very much enabled by widgets. So what is a widget? A widget is a small piece of code that can be included on any web site that provides a piece of functionality, 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 using its portlets as widgets. You can embed a particular instance of a portlet running on your site into another site, such as Facebook, to which you have access to add content. This opens up a whole new avenue of exposure to your web site that you would not otherwise have had.

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. Under this are three tabs.

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.

Message Boards

Categories | Recent Posts | Statistics

Search Search Messages

Showing 11 results.

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 52: Liferay's forums in Facebook

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://slashdot.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.

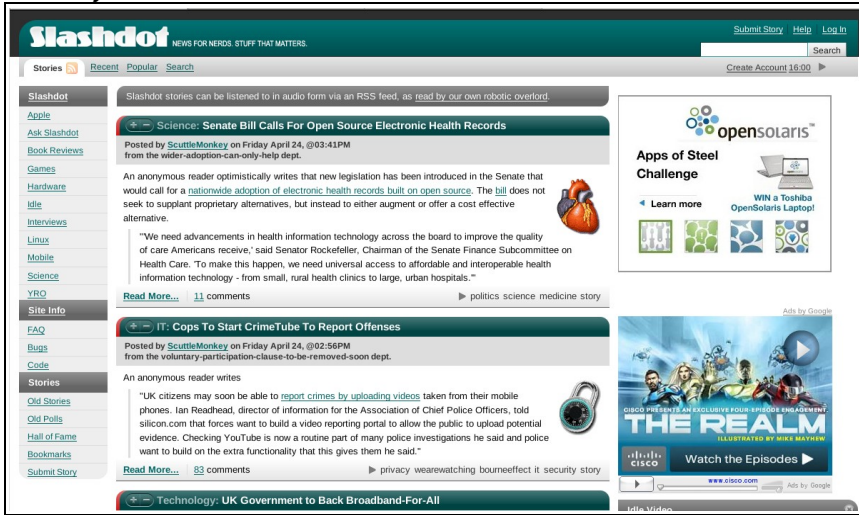


Illustration 53: Slashdot, one of the first blogs on the Internet.

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 Application 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.

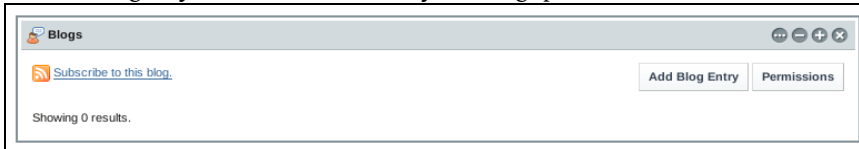


Illustration 54: Initial view of the Blogs portlet.

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 the way we want it to.

CONFIGURING THE BLOGS PORTLET

Click the menu button (the one with three dots on it), and then click *Configuration*. You will see a page allowing you to configure the various options for the Blogs portlet:

Setup Permissions Sharing Scope

Current Archived

Set the display styles used to display Blogs when viewed via as a regular page or as an RSS.

Page

Maximum Items to Display 1

Display Style Full Content

Enable Ratings

Enable Comments

Enable Comment Ratings

RSS

Maximum Items to Display 20

Display Style Full Content

Format Atom 1.0

Save

Illustration 55: Blogs Configuration

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 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.

Maximum Items to Display: Lets you configure the number of entries to appear in your RSS feed.

Display Style: Same as the display style for the web, but with the RSS feed.

Format: Lets you choose between RSS 1.0, RSS 2.0, or Atom formats for your feed.

Set the settings the way you want them and click *Save*.

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 that should have the ability to add blog entries, and then click *Save*.

Now you're ready to begin adding blog entries. Click the *Add Blog Entry* button. You will see the following data entry screen:

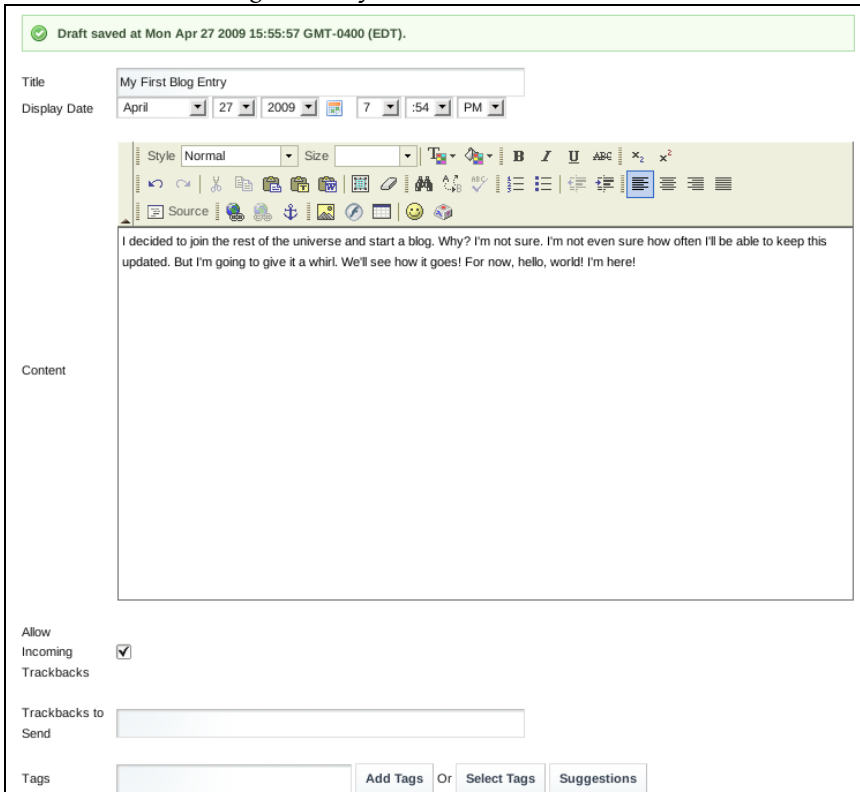


Illustration 56: Adding a Blog entry

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, as 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. These 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 Incoming 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.

Once you have finished your blog entry, click *Publish*. You'll go back to the list of entries, and you will see your entry. Here is what it looks like when the display style is set to *Abstract* and the number of entries is set to 10:

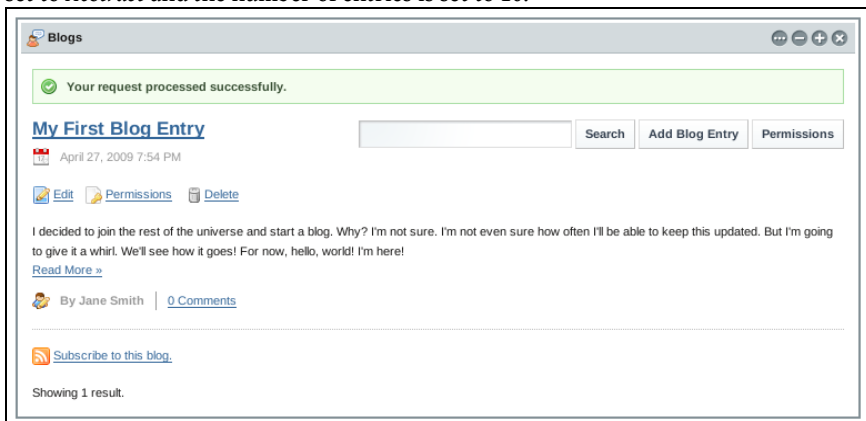


Illustration 57: First blog entry added.

You can see that in the summary view, you don't see the trackback 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 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 will give 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 Application 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:

The screenshot shows the configuration interface for the Blogs Aggregator portlet. At the top, there is a title bar with a menu icon and the text "Blogs Aggregator". Below this are four tabs: "Setup" (active), "Permissions", "Sharing", and "Scope". Underneath are two more tabs: "Current" (active) and "Archived". The main configuration area contains several settings:

- selection-method:** A dropdown menu currently set to "Users".
- Organization:** A text input field with "Select" and "Remove" buttons.
- Display Style:** A dropdown menu currently set to "Abstract".
- Maximum Items to Display:** A dropdown menu currently set to "20".
- Enable RSS Subscription:** A checkbox that is checked.

A "Save" button is located at the bottom left of the configuration area.

Illustration 58: 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.

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.

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.

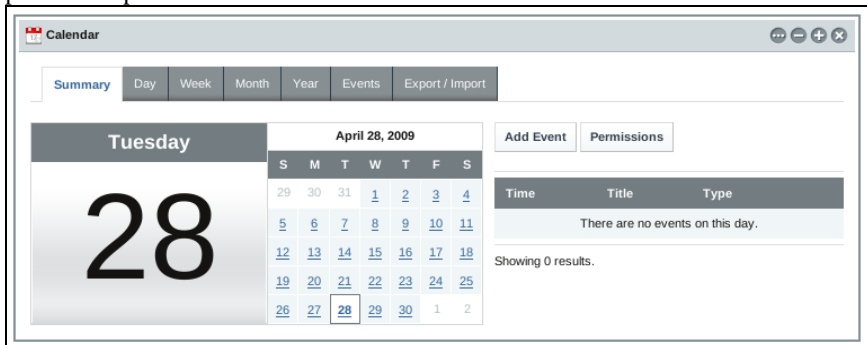


Illustration 59: The Liferay Calendar Portlet

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 categories of options: the "From" field in event reminder emails, the template for event reminder emails, and display settings for the portlet itself.

EVENT FROM

Setting the Event From name and email address allows you to define the From: field in the email messages that you receive from the Calendar portlet. 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 allow 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, or select whether it shows today's events.

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. Select 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.

Repeat: If the event repeats on a schedule, select the schedule.

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, 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 60: Liferay's Chat Portlet

The Chat portlet is very simple to use. To change the settings, click *Settings*. 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*.

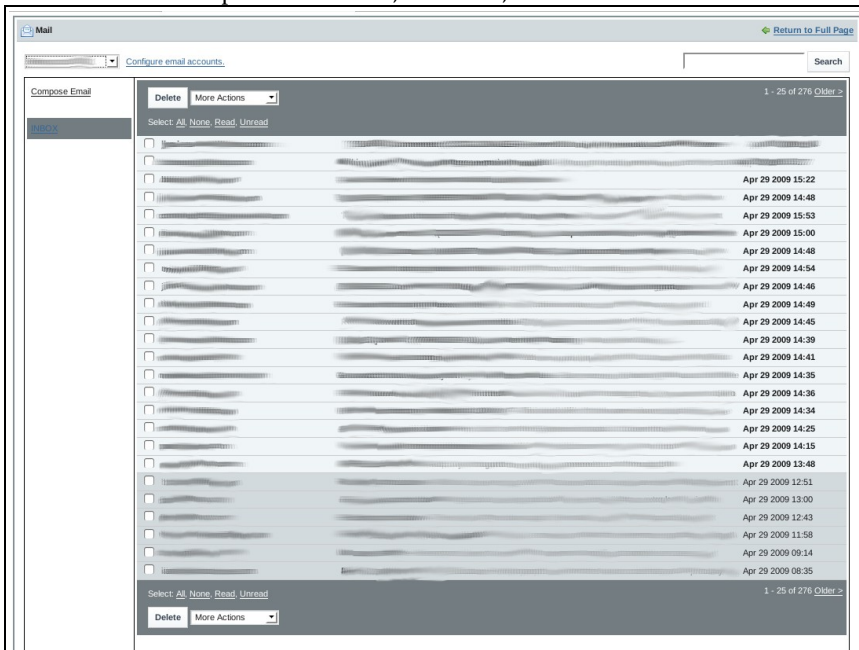


Illustration 61: Liferay Mail Portlet, with messages blurred out to protect the owner of the account.

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 an IMAP account or a Gmail account. Choose the option that you wish, and fill out the form that appears.

For an IMAP account, these fields are necessary:

Email Address: The email address which receives mail for this account.

User Name: The user name for logging into the account.

Password: The password for logging into the account.

Incoming IMAP Server: The host name for your IMAP (Internet Mail Access Protocol) server.

Incoming Port: The port upon which the IMAP 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.

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.

When finished, click *Save*. Next, click *Check Your Mail*. 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.

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 collaboration features 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*.

Email From

The first tab is labeled *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.

Anonymous Posting

This tab contains a single check box. Check the box if you want to enable anonymous posting to the message boards. Uncheck the box if users must register on your site before posting to the message boards.

Ratings

This tab also contains a single check box. Check the box if you want to allow users to rate the messages that are posted. Uncheck the box to disable this feature.

When you have finished setting up the message boards, click *Save* on the tab you are using and then click *Return to Full Page*.

Permissions

The default page that the Message Boards portlet displays has two 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

You are now ready to add categories to your message boards. Click the *Add Category* 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 tabs: *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* tab 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: *Categories*, *My Posts*, *My Subscriptions*, *Recent Posts*, and for administrative users, *Statistics* and *Banned Users*. The *Categories* tab allows users to browse the categories for an appropriate place to post a thread. 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 conversation you had once 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. 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.

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.

To post a new message thread, go to the *Categories* tab, click on a category, and then click on 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.

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

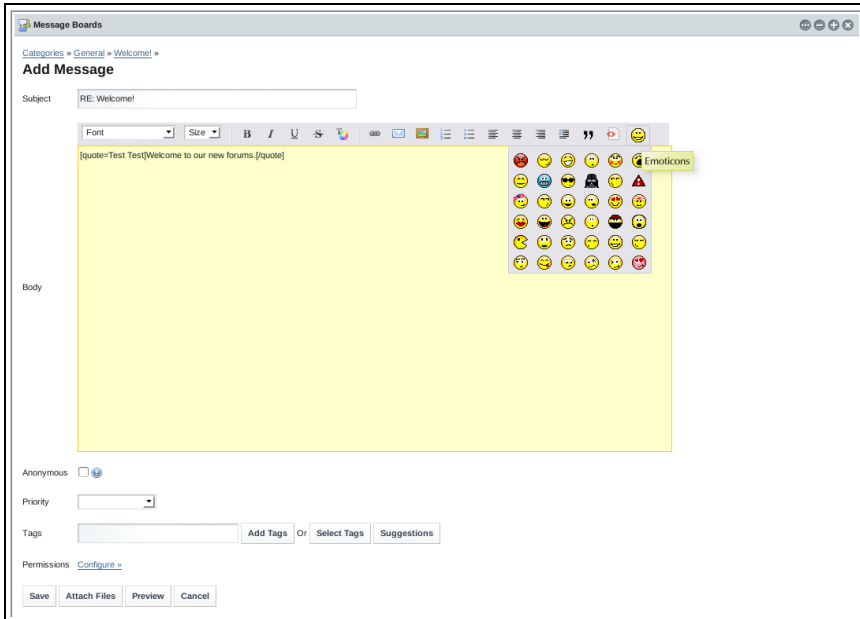


Illustration 62: Editing a message boards post. You can see the emoticons that are available in the editor.

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, 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.

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.

Roles » [Message Board Administrator](#) » [Message Boards](#)

All

Message Boards

Action

Configuration

View

Message Boards

Action

Add Category

Ban User

Message Boards Category

Action

Add File

Add Message

Add Subcategory

Delete

Move Thread

Permissions

Reply to Message

Subscribe

Update

Update Thread Priority

View

Message Boards Message

Action

Delete

Permissions

Subscribe

Update

View

Name

Showing 0 results.

Illustration 63: Message Boards permissions.

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. First click on the thread. If 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 of 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 platform.

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 for 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 Application* menu and then click *Configuration* in the portlet menu in the Wiki portlet's title bar. You should see some options which will now likely be familiar to you.

As with the Message Boards portlet, you can configure messages which come from the Wiki portlet. The *Email From* tab lets you configure a name and an email address which will be populated in the *From* field of email messages sent by the portlet. The *Page Added Email* tab lets you customize the message that is sent to subscribers when a new Wiki page has been added. The *Page Updated Email* tab lets you customize the message that is sent to subscribers when a Wiki page has been edited.

The *Display Settings* tab lets you configure how wikis and wiki pages are shown to users. You can choose which wikis are visible by moving them to a Visible or Hidden list. You can also enable comments and comment ratings on wiki pages. This allows users to interact with each other concerning edits to the content, enabling them to collaborate on changes.

The *RSS* tab allows you to set up the RSS feed for the Wiki. You can set the maximum number of items to display and whether you want to display the full content, an abstract, or just a title in the feed.

Once you have set the options the way you want them, click *Save* and then click *Return to Full Page*.

Managing Wikis

The Wiki portlet can contain many wikis. By default, it contains only one, called *Main*. At the top left of the portlet window is a small icon of a wrench. This is the *Manage Wikis* button. Click on it. 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.

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.

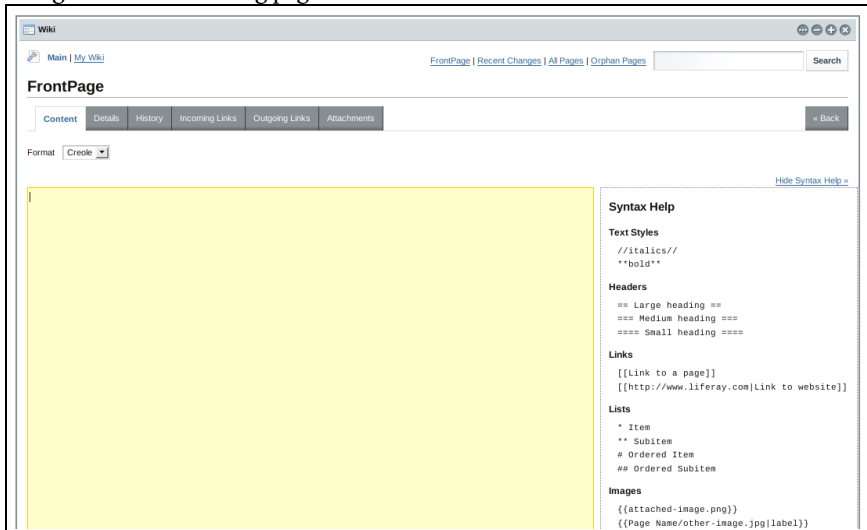


Illustration 64: Editing the default page in the wiki portlet

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:

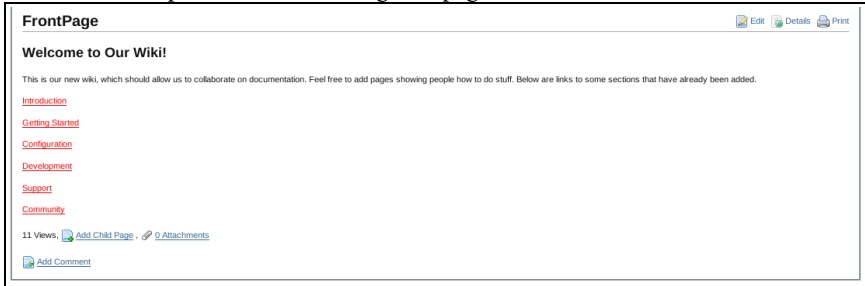


Illustration 65: Wiki text added to front page

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 generally 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 compare 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.

Summary

We have together explored all 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 IMAP 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.

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.

5. 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 configura-

tion 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=\
  com.liferay.portal.upgrade.UpgradeProcess_4_3_0,\
  com.liferay.portal.upgrade.UpgradeProcess_4_3_1,\
  com.liferay.portal.upgrade.UpgradeProcess_4_3_2,\
  com.liferay.portal.upgrade.UpgradeProcess_4_3_3,\
  com.liferay.portal.upgrade.UpgradeProcess_4_3_4,\
  com.liferay.portal.upgrade.UpgradeProcess_4_3_5,\
  com.liferay.portal.upgrade.UpgradeProcess_4_4_0,\
```

```
com.liferay.portal.upgrade.UpgradeProcess_5_0_0, \
com.liferay.portal.upgrade.UpgradeProcess_5_1_0, \
com.liferay.portal.upgrade.UpgradeProcess_5_1_2, \
com.liferay.portal.upgrade.UpgradeProcess_5_2_0, \
com.liferay.portal.upgrade.UpgradeProcess_5_2_1, \
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;
```

```
public static final int NEVER = 0;
```

```
public 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=\
com.liferay.portal.deploy.auto.HookAutoDeployListener, \
com.liferay.portal.deploy.auto.LayoutTemplateAutoDeployListener, \
com.liferay.portal.deploy.auto.PortletAutoDeployListener, \
com.liferay.portal.deploy.auto.ThemeAutoDeployListener, \
com.liferay.portal.deploy.auto.WebAutoDeployListener, \
com.liferay.portal.deploy.auto.exploded.tomcat.LayoutTemplateExplodedTomcatListener, \
com.liferay.portal.deploy.auto.exploded.tomcat.PortletExplodedTomcatListener, \
com.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 serv-

er 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=
auto.deploy.default.dest.dir=../webapps
auto.deploy.geronimo.dest.dir=${org.apache.geronimo.base.dir}/deploy
auto.deploy.geronimo-jetty.dest.dir=${org.apache.geronimo.base.dir}/deploy
auto.deploy.geronimo-tomcat.dest.dir=${org.apache.geronimo.base.dir}/deploy
auto.deploy.glassfish.dest.dir=${com.sun.aas.instanceRoot}/autodeploy
auto.deploy.glassfish-tomcat.dest.dir=${com.sun.aas.instanceRoot}/autodeploy
auto.deploy.jboss-tomcat.dest.dir=${jboss.server.home.dir}/deploy
auto.deploy.jetty.dest.dir=${jetty.home}/webapps
auto.deploy.jonas-jetty.dest.dir=${jonas.base}/webapps/autoload
auto.deploy.jonas-tomcat.dest.dir=${jonas.base}/webapps/autoload
auto.deploy.resin.dest.dir=${resin.home}/webapps
auto.deploy.tomcat.dest.dir=${catalina.base}/webapps
auto.deploy.weblogic.dest.dir=${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.web-sphere.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/trunk/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.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
```

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 start with the previous character.

```
plugin.notifications.packages.ignored=liferay/sample-jsp-portlet
```

PORTLET

Set this property for the portlet container implementation to use. The default implementation is the internal implementation and provides for the best backwards compatibility. The Sun implementation provides more features and will be the recommended implementation in the future.

```
portlet.container.impl=internal
#portlet.container.impl=sun
```

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
```

PORTLET COORDINATION

Set this property to specify how events are distributed. If the value is *ALL_PORTLETS*, then events will be distributed to all portlets that are then deployed. If the value is *ALL_PORTLETS_ON_PAGE*, then events will be distributed to all portlets that are present on a portal page.

```
portlet.event.distribution=ALL_PORTLETS_ON_PAGE
```

Set this property to specify the maximum number of events that can be generated from portlets to prevent infinite loops.

```
portlet.event.max.generation=3
```

Set this property to specify how public render parameters are distributed. If the value is *ALL_PORTLETS*, then public render parameters will be distributed to all portlets that are the deployed. If the value is *ALL_PORTLETS_ON_PAGE*, then public render parameters will be distributed to all portlets that are present on a portal page.

```
portlet.public.render.parameter.distribution=ALL_PORTLETS_ON_PAGE
```

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=\
META-INF/portal-model-hints.xml,\
META-INF/workflow-model-hints.xml,\
META-INF/ext-model-hints.xml,\
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.persistence.read.only.prefixes=\
contains,\
count,\
find,\
get

service.builder.service.read.only.prefixes=\
get,\
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*.

```
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/wsrp-spring.xml, \
\
META-INF/mirage-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/counter-hbm.xml, \
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.ClassicQueryTranslatorFactory
```

Set this property to true to enable Hibernate cache monitoring. See LPS-2056 for more information.

```
hibernate.generate_statistics=false
```

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 Hypersonic 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/hsqldb/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&characterEn-
coding=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&characterEn-
coding=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. Specifically, all JQuery scripts should go first.

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=\
  \
  #
  # JQuery scripts
  #
  \
  jquery/jquery.js,\
  jquery/cookie.js,\
  jquery/hover_intent.js,\
  jquery/j2browse.js,\
  jquery/livequery.js,\
  \
  #
  # jQuery UI 1.5
  #
  \
  jquery/ui.core.js,\
  jquery/ui.datepicker.js,\
  jquery/ui.dialog.js,\
  jquery/ui.draggable.js,\
  jquery/ui.slider.js,\
```

```
\
#
# jQuery UI 1.6
#
\
jquery/ui.color_picker.js,\
\
#
# Miscellaneous scripts
#
\
misc/class.js,\
misc/swfobject.js,\
\
#
# Liferay base utility scripts
#
\
liferay/language.js,\
liferay/liferay.js,\
liferay/util.js,\
\
#
# Liferay utility scripts
#
\
liferay/events.js,\
liferay/popup.js,\
liferay/portal.js,\
liferay/portlet.js,\
liferay/portlet_sharing.js,\
liferay/portlet_url.js,\
\
#
# Liferay modules
#
\
liferay/color_picker.js,\
liferay/dock.js,\
liferay/menu.js
```

Specify the list of everything files (everything else not already in the list of bare-bone files).

```
javascript.everything.files=\
\
#
# JQuery scripts
```

```
#
\
jquery/form.js,\
jquery/jeditable.js,\
jquery/json.js,\
jquery/livesearch.js,\
jquery/media.js,\
jquery/position.js,\
jquery/scrollTo.js,\
jquery/selection.js,\
jquery/treeview.js,\
\
#
# jQuery UI 1.5
#
\
jquery/ui.accordion.js,\
jquery/ui.droppable.js,\
jquery/ui.resizable.js,\
jquery/ui.sortable.js,\
jquery/ui.tabs.js,\
#jquery/ui.selectable.js,\
\
#
# jQuery UI 1.6
#
\
jquery/ui.autocomplete.js,\
jquery/ui.tree.js,\
\
#
# jQuery UI 1.5 Effects library
#
\
#jquery/effects.core.js,\
#jquery/effects.blind.js,\
#jquery/effects.bounce.js,\
#jquery/effects.clip.js,\
#jquery/effects.drop.js,\
#jquery/effects.explode.js,\
#jquery/effects.fold.js,\
#jquery/effects.highlight.js,\
#jquery/effects.pulsate.js,\
#jquery/effects.scale.js,\
#jquery/effects.shake.js,\
#jquery/effects.slide.js,\
#jquery/effects.transfer.js,\
\
```



```
#
# Liferay base utility scripts
#
\
liferay/layout.js,\
liferay/observable.js,\
\
#
# Liferay modules
#
\
liferay/auto_fields.js,\
liferay/dynamic_select.js,\
liferay/layout_configuration.js,\
liferay/layout_exporter.js,\
liferay/notice.js,\
liferay/navigation.js,\
liferay/panel.js,\
liferay/panel_floating.js,\
liferay/search_container.js,\
liferay/session.js,\
liferay/tags_categories_selector.js,\
liferay/tags_entries_selector.js,\
liferay/undo_manager.js,\
liferay/upload.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*.

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
```

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-as-  
sociations,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.ScreenNameGen-  
erator
```

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.ScreenNameVal-  
idator  
users.screen.name.validator=com.liferay.portal.security.auth.LiberalScreen-  
NameValidator
```

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 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 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,cus-
tom-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
```

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-at-
tributes
```

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
```

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,e1
_GR,hu_HU,it_IT,ja_JP,ko_KR,nb_NO,fa_IR,pl_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=\
    Pacific/Midway,\
    Pacific/Honolulu,\
    America/Anchorage,\
    America/Los_Angeles,\
    America/Denver,\
    America/Chicago,\
    America/New_York,\
    America/Puerto_Rico,\
    America/St_Johns,\
    America/Sao_Paulo,\
    America/Noronha,\
    Atlantic/Azores,\
    UTC,\
    Europe/Lisbon,\
    Europe/Paris,\
    Europe/Istanbul,\
    Asia/Jerusalem,\
    Asia/Baghdad,\
    Asia/Tehran,\
    Asia/Dubai,\
    Asia/Kabul,\
    Asia/Karachi,\
    Asia/Calcutta,\
    Asia/Katmandu,\
    Asia/Dhaka,\
    Asia/Rangoon,\
    Asia/Saigon,\
    Asia/Shanghai,\
    Asia/Tokyo,\
    Asia/Seoul,\
    Australia/Darwin,\
    Australia/Sydney,\
    Pacific/Guadalcanal,\
```



```
Pacific/Auckland,\nPacific/Enderbury,\nPacific/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
```

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=\
/portal/render_portlet,\
\
/document_library/get_file
```

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\ngroupId=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.lockout=retry 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.filter.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.fil-*

ters.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://openssohost.example.com:8080/opensso/UI/Login?
goto=http://portalhost.example.com:8080/c/portal/login
open.sso.logout.url=http://openssohost.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://openssohost.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.events.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=\
  /blogs/find_entry,\
  /blogs/rss,\
  /blogs/trackback,\
  \
  /bookmarks/open_entry,\
  \
  /document_library/get_file,\
```

```

\
/journal/get_article,\
/journal/get_articles,\
/journal/get_latest_article_content,\
/journal/get_structure,\
/journal/get_template,\
/journal/view_article_content,\
/journal_articles/view_article_content,\
\
/layout_management/sitemap,\
\
/message_boards/find_category,\
/message_boards/find_message,\
/message_boards/find_thread,\
/message_boards/get_message_attachment,\
/message_boards/rss,\
\
/my_places/view,\
\
/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.Open-
IdAutoLogin,com.liferay.portal.security.auth.OpenSSOAutoLogin,com.liferay.-
portal.security.auth.RememberMeAutoLogin,com.liferay.portal.security.auth.S-
iteMinderAutoLogin
```

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|emailAd-
dress>&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=0123456789ABCDEFGHIJKLMNPQRSTUVWXYZabcde-
fghijklmnopqrstuvwxyz
```

Examples:

```
passwords.regexptoolkit.length=4
passwords.regexptoolkit.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?

```
permissions.user.check.algorithm=1
```

The second algorithm (the default) does a database join and checks the permissions in one step, by calling *countByGroupsRoles*, *countByGroupsPermissions*, *countByUserRoles*, *countByUserGroupRole*, and *countByUsersPermissions* in one method.

```
permissions.user.check.algorithm=2
```

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?

ted 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?

```
permissions.user.check.algorithm=3
```

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.

```
permissions.user.check.algorithm=4
```

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
```

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
```

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.liferay.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 workaround 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.SiteMinder-
LogoutAction
#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 for 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=Welcme
```

Set the layout template id of the private layout.

```
default.user.private.layout.template.id=2_columns_ii
```

Set the portlet ids for the columns specified in the layout template.

```
default.user.private.layout.column-1=71_INSTANCE_0Y0d,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.s.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
```

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 lay-

outs 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.friendlyable=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.friendlyable[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.friendlyable[panel]=true
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.friendly[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.friendly[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.friendly[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.Layout-
TypeArticleConfigurationUpdateAction
layout.configuration.action.delete[article]=com.liferay.portal.model.Layout-
TypeArticleConfigurationDeleteAction
```

Settings for URL layouts.

```
layout.edit.page[url]=/portal/layout/edit/url.jsp
layout.view.page[url]=
layout.url[url]=${url}
layout.url.friendly[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]=${liferay:mainPath}/portal/layout?p_l_id=${link-
ToPlid}
layout.url.friendly[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_ii
```

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_l_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
```

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 tinymce-simple.

```
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
```

MIME TYPES

Input a list of comma delimited mime types that are not available by default from `javax.activation.MimetypesFileTypeMap`.

```
mime.types=\
    application/msword doc,\
```

```
application/pdf pdf,\  
application/vnd.ms-excel xls,\  
application/vnd.ms-powerpoint ppt,\  
application/x-ms-wmp wmv,\  
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 Amazons license at <http://www.amazon.com> for more information.

```
#amazon.access.key.id=  
#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
```

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 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 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/portal`.

```
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.MySQLInnoDBDialect
#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.regex.strip=
lucene.file.extractor.regex.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.perfield.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 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
```

SOURCEFORGE

```
source.forge.mirrors=\
  http://downloads.sourceforge.net,\      # Redirect
  http://internap.dl.sourceforge.net,\     # San Jose, CA
  http://superb-east.dl.sourceforge.net,\  # McLean, Virginia
  http://superb-west.dl.sourceforge.net,\  # Seattle, Washington
  http://easynews.dl.sourceforge.net,\     # Phoenix, AZ
  http://kent.dl.sourceforge.net,\        # Kent, UK
  http://ufpr.dl.sourceforge.net,\        # Curitiba, Brazil
  http://belnet.dl.sourceforge.net,\       # Brussels, Belgium
  http://switch.dl.sourceforge.net,\      # Lausanne, Switzerland
  http://mesh.dl.sourceforge.net,\        # Duesseldorf, Germany
  http://ovh.dl.sourceforge.net,\         # Paris, France
```

```
http://dfn.dl.sourceforge.net,\      # Berlin, Germany
http://heanet.dl.sourceforge.net,\    # Dublin, Ireland
http://garr.dl.sourceforge.net,\     # Bologna, Italy
http://surfnet.dl.sourceforge.net    # Amsterdam, The Netherlands
http://jaist.dl.sourceforge.net,\    # Ishikawa, Japan
http://nchc.dl.sourceforge.net,\     # Tainan, Taiwan
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.model.ContactListener
value.object.listener.com.liferay.portal.model.Layout=com.liferay.portal.model.LayoutListener
value.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
```

Value objects are cached by default. You can disable caching for all objects or per object.

For mapping tables, the key is the mapping table itself.

```
value.object.finder.cache.enabled=true
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.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_rcv_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=80
```

```
96;down_thread=false;up_thread=false):pbcast.GMS(join_timeout=5000;join_retr
y_timeout=2000;shun=false;print_local_addr=true)
```

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

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=86400000
```

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}/reposit-
ory.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:

- `com.liferay.mail.util.CyrusHook`
- `com.liferay.mail.util.DummyHook`
- `com.liferay.mail.util.FuseMailHook`
- `com.liferay.mail.util.SendmailHook`
- `com.liferay.mail.util.ShellHook`

```
mail.hook.impl=com.liferay.mail.util.DummyHook
```

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;" -nofirst-startwizard
```

This is tested with OpenOffice 2.3.x.

```
openoffice.server.enabled=false
openoffice.server.host=127.0.0.1
openoffice.server.port=8100
```

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.documentlib-  
rary.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:so-  
cial-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:so-  
cial-bookmark:url}&title=${liferay:social-bookmark:title}  
social.bookmark.post.url[technorati]=http://technorati.com/cosmos/search.htm  
l?url=${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.ServletVelo-  
cityResourceListener,com.liferay.portal.velocity.JournalTemplateVelocityRe-  
sourceListener,com.liferay.portal.velocity.ThemeLoaderVelocityResourceL-  
istener,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.LiferayRe-  
sourceManager  
velocity.engine.resource.manager.cache=com.liferay.portal.velo-  
city.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 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
```

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 cache filter will cache content. See *ehcache.xml* to modify the cache expiration time to live.

```
com.liferay.portal.servlet.filters.cache.CacheFilter=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
```

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 minifier filter is used to minify CSS and JavaScript.

```
com.liferay.portal.servlet.filters.minifier.MinifierFilter=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 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 the following to true to enable programmatic configuration to let the Web-

DAV be configured for litmus testing. This should never be set to true unless you are running the litmus tests.

```
webdav.litmus=false
```

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
```

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
```

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=
```

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.tpl
admin.email.user.added.body=com/liferay/portlet/admin/dependencies/email_
r_added_body.tpl
admin.email.password.sent.enabled=true
admin.email.password.sent.subject=com/liferay/portlet/admin/dependencies/ema
il_password_sent_subject.tpl
admin.email.password.sent.body=com/liferay/portlet/admin/dependencies/email_
password_sent_body.tpl
```

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/  
email_subject.tpl  
announcements.email.body=com/liferay/portlet/announcements/dependencies/  
email_body.tpl
```

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
```

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.tpl  
blogs.email.comments.added.body=com/liferay/portlet/blogs/dependencies/  
email_comments_added_body.tpl  
blogs.page.abstract.length=400  
blogs.rss.abstract.length=200  
blogs.trackback.excerpt.length=50
```

Set the interval on which the `TrackbackVerifierJob` will run. The value is set in one minute increments.

```
blogs.trackback.verifier.job.interval=5
```

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,break-  
fast,call,chat,class,club-event,concert,dinner,event,graduation,happy-  
hour,holiday,interview,lunch,meeting,movie,net-event,other,party,perform-  
ance,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.tpl
calendar.email.event.reminder.body=com/liferay/portlet/calendar/dependencies/email_event_reminder_body.tpl
```

COMMUNITIES PORTLET

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/dependencies/email_membership_reply_subject.tpl
communities.email.membership.reply.body=com/liferay/portlet/communities/dependencies/email_membership_reply_body.tpl
communities.email.membership.request.subject=com/liferay/portlet/communities/dependencies/email_membership_request_subject.tpl
communities.email.membership.request.body=com/liferay/portlet/communities/dependencies/email_membership_request_body.tpl
```

DOCUMENT LIBRARY PORTLET

Set the name of a class that implements *com.liferay.documentlibrary.util.Hook*. The document library server will use this to persist documents.

Available hooks are:

- `com.liferay.documentlibrary.util.FileSystemHook`
- `com.liferay.documentlibrary.util.JCRHook`
- `com.liferay.documentlibrary.util.S3Hook`

Examples:

```
#dl.hook.impl=com.liferay.documentlibrary.util.AdvancedFileSystemHook
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}/document_library
```

S3Hook

```
dl.hook.s3.access.key=
```

```
dl.hook.s3.secret.key=  
dl.hook.s3.bucket.name=
```

Set the maximum file size and valid file extensions for documents. 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*.

Examples:

```
#dl.file.max.size=307200  
#dl.file.max.size=1024000  
dl.file.max.size=3072000
```

A file extension of * will permit all file extensions.

You can map a GIF for the extension by adding the image to the theme's image display and document library folder. The wildcard extension of * will be ignored. For example, the default image for the DOC extension would be found in: */html/themes/_unstyled/images/document_library/doc.gif*.

Example File Extensions:

```
dl.file.extensions=.bmp,.css,.doc,.dot,.gif,.gz,.htm,.html,.jpg,.js,.lar,.od  
b,.odf,.odg,.odp,.ods,.odt,.pdf,.png,.ppt,.rtf,.swf,.sxc,.sxi,.sxw,.tar,.tif  
f,.tgz,.txt,.vsd,.xls,.xml,.zip
```

Set which files extensions are comparable by the diff tool. If OpenOffice integration is enabled, then it is also possible to compare some binary files that are can be converted to text.

```
dl.comparable.file.extensions=.css,.js,.htm,.html,.txt,.xml  
#dl.comparable.file.extensions=.css,.doc,.js,.htm,.html,.odt,.rtf,.sxw,.txt,  
.xml
```

Set folder names that will be used to synchronize with a community's set of private and public layouts. This will allow users to manage layouts using the Document Library portlet, and ultimately, via WebDAV. This feature is experimental.

```
dl.layouts.sync.enabled=false  
dl.layouts.sync.private.folder=Pages - Private  
dl.layouts.sync.public.folder=Pages - Public
```

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
```

INVITATION PORTLET

```
invitation.email.max.recipients=20
invitation.email.message.body=com.liferay/portlet/invitation/dependencies/email_message_body.tpl
invitation.email.message.subject=com.liferay/portlet/invitation/dependencies/email_message_subject.tpl
```

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=\ncom.liferay.portlet.journal.util.TokensTransformerListener,\n#com.liferay.portlet.journal.util.PropertiesTransformerListener,\ncom.liferay.portlet.journal.util.ContentTransformerListener,\ncom.liferay.portlet.journal.util.LocaleTransformerListener,\ncom.liferay.portlet.journal.util.RegexTransformerListener,\ncom.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\n#journal.transformer.regex.replacement.0=production.sample.com\n#journal.transformer.regex.pattern.1=staging.sample.com\n#journal.transformer.regex.replacement.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\njournal.email.from.address=test@liferay.com\njournal.email.article.approval.denied.enabled=false\njournal.email.article.approval.denied.subject=com/liferay/portlet/journal/dependencies/email_article_approval_denied_subject.tpl\njournal.email.article.approval.denied.body=com/liferay/portlet/journal/dependencies/email_article_approval_denied_body.tpl\njournal.email.article.approval.granted.enabled=false\njournal.email.article.approval.granted.subject=com/liferay/portlet/journal/dependencies/email_article_approval_granted_subject.tpl\njournal.email.article.approval.granted.body=com/liferay/portlet/journal/dependencies/email_article_approval_granted_body.tpl\njournal.email.article.approval.requested.enabled=false\njournal.email.article.approval.requested.subject=com/liferay/portlet/journal
```

```

/dependencies/email_article_approval_requested_subject.tpl
journal.email.article.approval.requested.body=com/liferay/portlet/journal/de
pendencies/email_article_approval_requested_body.tpl
journal.email.article.review.enabled=false
journal.email.article.review.subject=com/liferay/portlet/journal/dependen-
cies/email_article_review_subject.tpl
journal.email.article.review.body=com/liferay/portlet/journal/dependencies/e
mail_article_review_body.tpl

```

Specify the strategy used when Journal content is imported using the LAR system.

```

journal.lar.creation.strategy=com.liferay.portlet.journal.lar.JournalCre-
ationStrategyImpl

```

Specify the path to the template used for providing error messages on Journal templates.

```

journal.error.template.velocity=com/liferay/portlet/journal/dependencies/er-
ror.vm
journal.error.template.xsl=com/liferay/portlet/journal/dependencies/er-
ror.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/mes-
sageboards/dependencies/email_message_added_subject_prefix.tpl
message.boards.email.message.added.body=com/liferay/portlet/messageboards/de
pendencies/email_message_added_body.tpl
message.boards.email.message.added.signature=com/liferay/portlet/message-
boards/dependencies/email_message_added_signature.tpl
message.boards.email.message.updated.enabled=true
message.boards.email.message.updated.subject.prefix=com/liferay/portlet/mes-
sageboards/dependencies/email_message_updated_subject_prefix.tpl
message.boards.email.message.updated.body=com/liferay/portlet/messageboards/
dependencies/email_message_updated_body.tpl
message.boards.email.message.updated.signature=com/liferay/portlet/message-

```



```
boards/dependencies/email_message_updated_signature.tpl
```

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
```

MY PLACES PORTLET

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: header-Type, 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
```

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/de-
pendencies/email_order_confirmation_subject.tpl
shopping.email.order.confirmation.body=com/liferay/portlet/shopping/depend-
encies/email_order_confirmation_body.tpl
shopping.email.order.shipping.enabled=true
shopping.email.order.shipping.subject=com/liferay/portlet/shopping/dependen-
cies/email_order_shipping_subject.tpl
shopping.email.order.shipping.body=com/liferay/portlet/shopping/dependen-
cies/email_order_shipping_body.tpl
```

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
```

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.MinimalTagsAssetValidat-
or
```

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.regex=([^\[\]\%&?@]+)
#wiki.page.titles.regex=((\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 regex should remove any characters that are forbidden in the regex specified in `wiki.page.titles.regex`.

```
wiki.page.titles.remove.regex=([^\[\]\%&?@]+)
```

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[classic_wiki]=com.liferay.portlet.wiki.engines.friki.FrikiEngine
wiki.formats.configuration.main[classic_wiki]=wiki.transform
wiki.formats.configuration.interwiki[classic_wiki]=intermap.txt
wiki.formats.edit.page[classic_wiki]=/html/portlet/wiki/edit/wiki.jsp
wiki.formats.help.page[classic_wiki]=/html/portlet/wiki/help/classic_wiki.jsp
wiki.formats.help.url[classic_wiki]=http://wiki.liferay.com/index.php/Wiki_Portlet
```

```
wiki.formats.engine[creole]=com.liferay.portlet.wiki.engines.jspwiki.JSPWikiEngine
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/email_page_added_subject_prefix.tpl
wiki.email.page.added.body=com/liferay/portlet/wiki/dependencies/email_page_added_body.tpl
wiki.email.page.added.signature=com/liferay/portlet/wiki/dependencies/email_page_added_signature.tpl
```

```
wiki.email.page.updated.enabled=true
wiki.email.page.updated.subject.prefix=com/liferay/portlet/wiki/dependencies/email_page_updated_subject_prefix.tpl
wiki.email.page.updated.body=com/liferay/portlet/wiki/dependencies/email_page_updated_body.tpl
wiki.email.page.updated.signature=com/liferay/portlet/wiki/dependencies/email_page_updated_signature.tpl
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*, 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.

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, 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 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 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 the *Liferay Developer's Guide*.

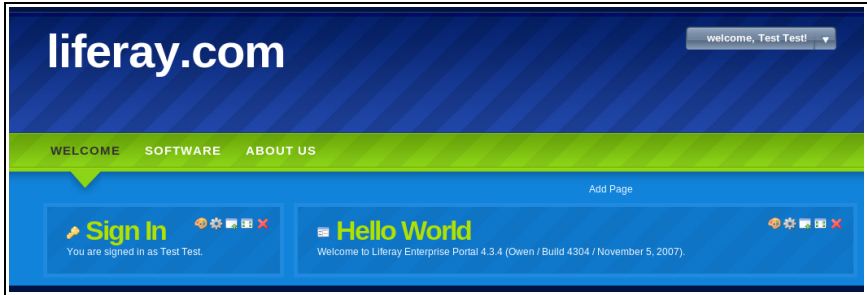


Illustration 67: Blue Pigment theme from Liferay's theme repository.

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 the *Liferay Developer's Guide*.

Hook Plugins

Hook plugins are new to 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 some 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.

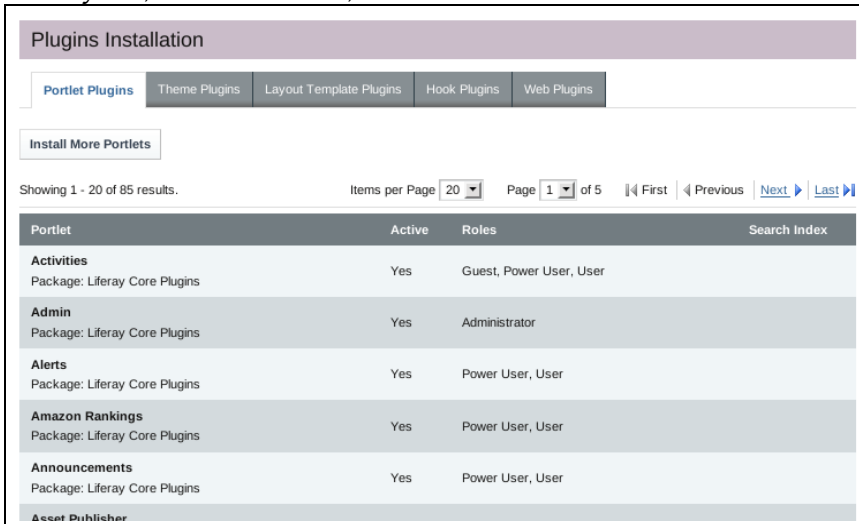


Illustration 68: Plugins Installation in the Control Panel

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.

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.

The screenshot shows the Liferay Plugin Installer interface. At the top, there are tabs for 'Browse Repository', 'Upload File', 'Download File', and 'Configuration', along with a 'Back' button. Below this, there are tabs for 'Portlet Plugins', 'Theme Plugins', 'Layout Template Plugins', 'Hook Plugins', and 'Web Plugins'. A search area includes fields for 'Keywords', 'Tag' (set to 'All'), 'Repository' (set to 'All'), and 'Install Status' (set to 'Not Installed or Out of Date'). A 'Search Plugins' button is present. Below the search area, it indicates 'Showing 1 - 20 of 84 results.' and provides pagination controls for 'Items per Page' (20), 'Page' (1 of 5), and navigation buttons for 'First', 'Previous', 'Next', and 'Last'. A table lists the following plugins:

Portlet Plugin	Trusted	Tags	Installed Version	Available Version	Modified Date ▲
WOL 5.2.2.1 ID: liferay/world-of-liferay/5.2.2.1/war This is the World of Liferay portlet that integrates social networking features.	Yes	wol	-	5.2.2.1	2/27/09 5:49 PM
Web Form 5.2.2.1		web			2/27/09 5:49

Illustration 69: Installing Plugins

To install a plugin, choose the plugin by clicking on its name. For example, if you want to provide a handy weather forecast to your web site, you might want to install the Weather portlet. This portlet provides a handy interface to which shows the current weather for a particular zip code which your users can customize.

Find the Weather 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 screenshot shows the Liferay Plugin Installer interface for the 'Weather' portlet. It includes a 'Weather' tab and a 'Back' button. The details are as follows:

Name: [Weather \(v5.2.0.1\)](#) [Download]
 Author: Liferay, Inc.
 Types: Portlet
 Tags:
 Licenses: MIT (Open Source)
 Liferay Versions: 5.2.0+
 Repository: <http://plugins.liferay.com/community> (Untrusted)

Short Description: This adds the Weather portlet to Liferay.
 Change Log: Adapted to the latest version of Liferay.

An 'Install' button is located at the bottom left.

Illustration 70: Adding the Weather Portlet to Liferay

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:

```
16:16:19,450 INFO [AutoDeployDir:183] Processing web-form-portlet-5.2.2.1.war
16:16:19,452 INFO [PortletAutoDeployListener:77] Copying portlets for /opt/liferay/liferay-portal-5.2.2/deploy/web-form-portlet-5.2.2.1.war
  Expanding: /opt/liferay/liferay-portal-5.2.2/deploy/web-form-portlet-5.2.2.1.war into /opt/liferay/liferay-portal-5.2.2/tomcat-5.5.27/temp/20090423161619617
  Copying 1 file to /opt/liferay/liferay-portal-5.2.2/tomcat-5.5.27/temp/20090423161619617/WEB-INF
  Copying 1 file to /opt/liferay/liferay-portal-5.2.2/tomcat-5.5.27/temp/20090423161619617/WEB-INF/classes
  Copying 1 file to /opt/liferay/liferay-portal-5.2.2/tomcat-5.5.27/temp/20090423161619617/WEB-INF/classes
  Copying 1 file to /opt/liferay/liferay-portal-5.2.2/tomcat-5.5.27/temp/20090423161619617/META-INF
  Copying 38 files to /opt/liferay/liferay-portal-5.2.2/tomcat-5.5.27/webapps/web-form-portlet
  Copying 1 file to /opt/liferay/liferay-portal-5.2.2/tomcat-5.5.27/webapps/web-form-portlet
  Deleting directory /opt/liferay/liferay-portal-5.2.2/tomcat-5.5.27/temp/20090423161619617
16:16:20,925 INFO [PortletAutoDeployListener:87] Portlets for /opt/liferay/liferay-portal-5.2.2/deploy/web-form-portlet-5.2.2.1.war copied successfully. Deployment will start in a few seconds.
16:16:27,993 INFO [PortletHotDeployListener:333] Unregistering portlets for web-form-portlet
16:16:28,000 INFO [PortletHotDeployListener:362] 1 portlet for web-form-portlet was unregistered
Apr 23, 2009 4:16:27 PM org.apache.catalina.startup.HostConfig checkResources
INFO: Undeploying context [/web-form-portlet]
16:16:28,408 INFO [PortletHotDeployListener:219] Registering portlets for web-form-portlet
Loading file:/opt/liferay/liferay-portal-5.2.2/tomcat-5.5.27/temp/6-web-form-portlet/WEB-INF/classes/portlet.properties
16:16:28,570 INFO [PortletHotDeployListener:298] 1 portlet for web-form-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.

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 have been 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.

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, 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.

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*.

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 portlet (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 71: 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 portlet 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.

THE CONTAINER UPON WHICH LIFERAY IS RUNNING

There are some containers, such as Oracle Application Server® and 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 location 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.

HOW LIFERAY IS BEING LAUNCHED



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, and this property is no longer detected via the relative path from the server launch location.

In versions of Liferay prior to 4.3.5, the default value of the hot deploy destination directory is a relative path (e.g., *../webapps* or *../server/default/deploy*). This path is relative to the folder from which the application server is normally launched. For example, Tomcat has the pictured directory structure.

The start up and shut down scripts are in the *bin* folder. So to start Tomcat, you would normally go into the *bin* folder to run the startup script which starts Liferay running in Tomcat.

Tomcat's hot deploy folder is the *webapps* folder. This folder is on the same level the *bin* folder is on. If you are at the command prompt inside of the *bin* folder (where you started Tomcat), to get to a file in the hot deploy folder you would reference it by using two dots to go back up one folder, and then the path separator (*/*), and then the name of the folder (*webapps*). So in the default configuration, the hot deploy destination directory is relative to the folder from which the application server was launched.

If you are launching your application server from another script—perhaps as part of a cron job—and your script does not enter the folder the application server's

startup scripts are in (in this case, `<Tomcat Home>/bin`), the relative path that is set by default will not work. Instead, the path will be relative to the path from which you launched the startup script. This will cause Liferay to create—in the Tomcat example—a `webapps` folder one folder up from where the startup script was launched. Since this is not the correct hot deploy folder for your application server, you will see the *copied successfully* message in the server console, but you will never see the *registered successfully* message.

To fix this, you can do one of two things: 1) Change the relative path to an absolute path as recommended above; or 2) Change the way you launch Liferay by making sure you go into the folder where the application server's startup scripts are before you launch them. Either one of these methods should fix your hot deploy problem and will result in the successful deployment of portlet and theme plugins.

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_5_2_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.

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.

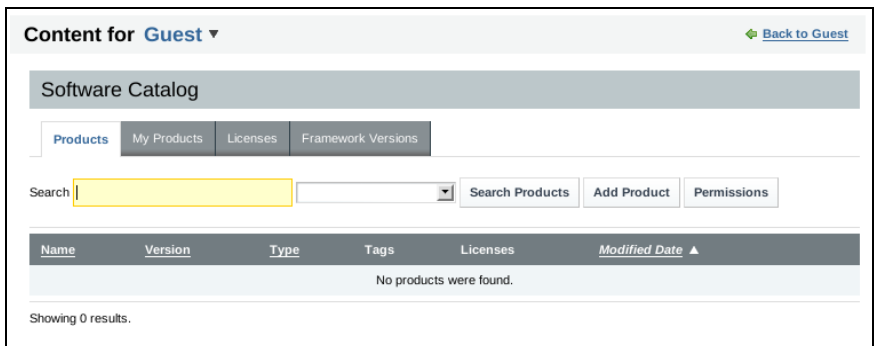


Illustration 72: Software Catalog with no products

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.

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.

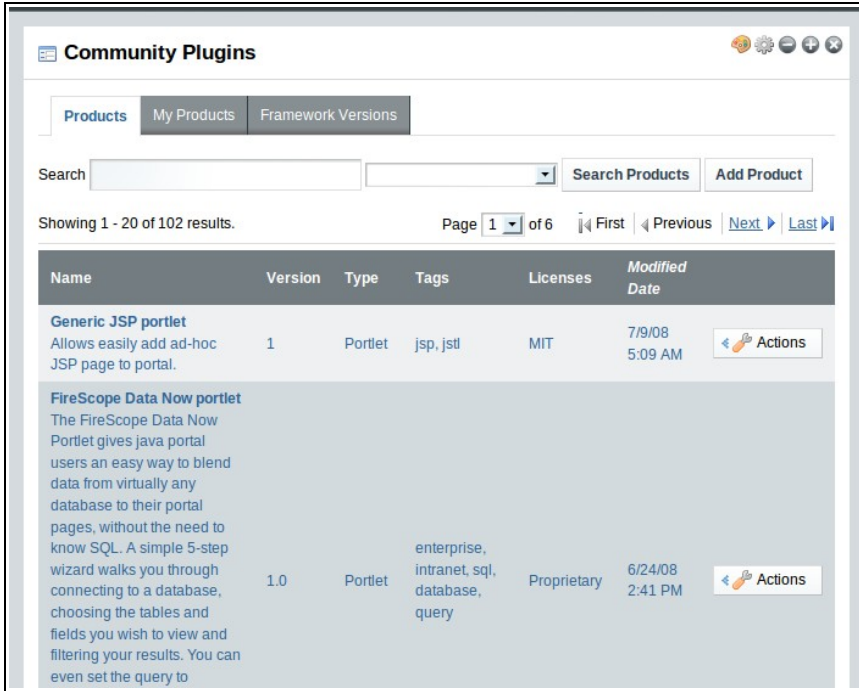


Illustration 73: Populated Software Catalog from liferay.com

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 fire-wall.

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.

The screenshot shows the 'Software Catalog' interface with a 'Product' tab selected. The form contains the following elements:

- Name:** A text input field highlighted in yellow.
- Type:** A dropdown menu set to 'Portlet Plugin'.
- Licenses:** A section with 'Recommended Licenses' (showing 'MIT License') and 'Other Licenses' (with a dropdown arrow).
- Author:** A text input field.
- Page URL:** A text input field.
- Tags:** A text input field with '(Comma delimited list)' as a hint.
- Short Description:** A large text area.
- Long Description:** A larger text area below the short description.
- Permissions:** A link labeled 'Configure »'.
- Buttons:** 'Save' and 'Cancel' buttons.
- Plugin Repository:** A section with 'Group ID' and 'Artifact ID' text input fields.
- Screenshots:** A section with an 'Add Screenshot' button.

Illustration 74: Adding a product to the Software Catalog (partial view)

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.

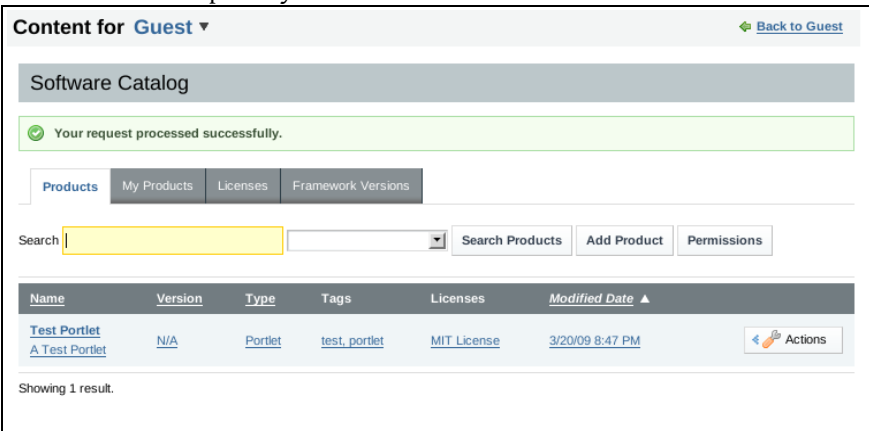


Illustration 75: Product has been added to the Software Catalog

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. For version 5.1, however, this portlet was replaced by the *World of Liferay* (WOL) portlet, which makes use of the many social networking fea-

tures which have been added to Liferay. So rather than just displaying a summary of your information, the WOL 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. For version 5.2, the WOL portlet itself has been broken down so that the social components can be installed separately from the software management components.

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, make the WOL portlets available to those who are using 5.1.x, 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 a Liferay 4.4.2 version, you will see the My Summary portlet. If you connect to Liferay's software repositories with a Liferay 5.1 version, you will see the WOL portlet. If you connect to Liferay's software repositories with a Liferay 5.2 version, 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 5.2.2 for the name, because our portlet should work on that version and higher, and <http://www.liferay.com> for the URL. Click *Save*.

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.

Software Catalog

✓ Your request processed successfully.

Product Version « Back

My Summary

Main Fields

Version Name: 1.0

Initial Version: [Empty]

Change Log: [Empty]

Supported Framework Versions: 5.2.2

Repository Fields

Download Page URL: [Empty]

Direct Download URL (Recommended): <http://www.liferay.com/portlets/my-summary-portlet-5.2.2.war>

Test Direct Download URL: No

Include Artifact in Repository: Yes

Save Cancel

Illustration 76: Adding a product version to the Software Catalog

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-5.2.2.war
    </download-url>
    <author>Rich Sezov</author>
    <screenshots/>
    <licenses>
      <license osi-approved="true">MIT License</license>
    </licenses>
    <liferay-versions/>
  </plugin-package>
</plugin-repository>
```

Save this document to a file called *liferay-plugin-package.xml* and put this file on your HTTP server where you uploaded your portlet .war file. 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. 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_5_2_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 the *Liferay Developer's Guide*, 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.

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 will be covered in the *Liferay Developer's Guide*, 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.

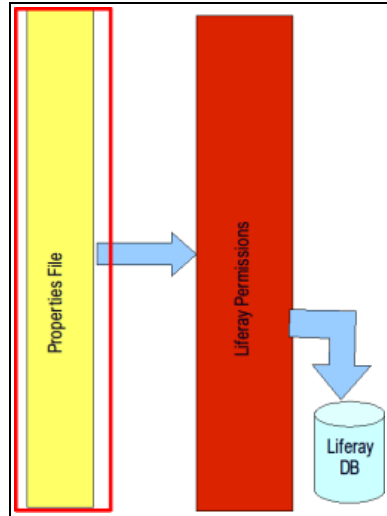


Illustration 77: Liferay SOA's first layer of security.

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.

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.

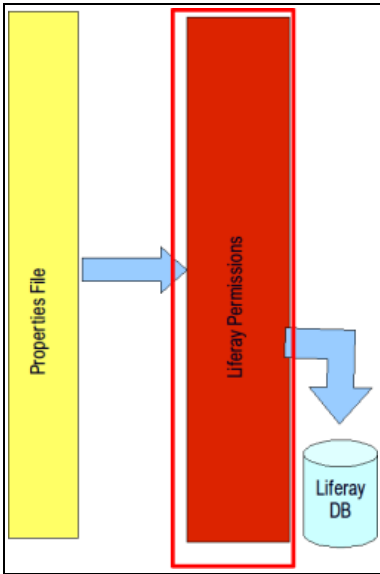


Illustration 78: Liferay SOA's second layer of security.

For example, say that a Document Library folder called *Documents* has been set up in a community. A role has been created called *Document Uploaders* 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
```



Tip: In older versions of Liferay (4.2.x and below), this password had to be the encrypted version from Liferay's database.

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 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 the *Liferay Developer's Guide*.

6. 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.

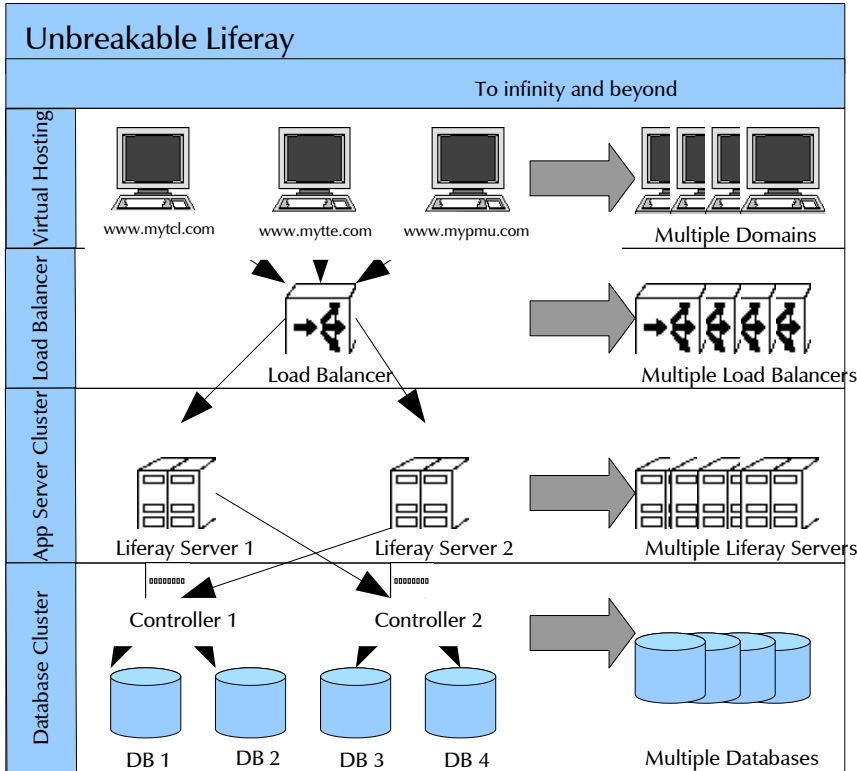


Illustration 79: "Unbreakable" Liferay architecture

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 minimum, 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

Liferay 5.2.x now defaults to using the file system for storing documents. This has proven to be the highest performing configuration for large document libraries, which is why this decision was made. You can use the file system for your clustered configuration, and Liferay's document library will prevent users from “colliding” with each other by versioning documents and locking the files before they are modified. If you have a Storage Area Network (SAN), you can configure Liferay to store documents there to take advantage of the extra redundancy. To configure the location where your documents are stored, use the following property:

```
dl.hook.file.system.root.dir=${liferay.home}/data/document_library
```

If you wish to cluster your document library configuration in a database, you can still do so using the Jackrabbit JSR-170 repository. You would also use the Jackrabbit repository if you want to have a JSR-170 compliant repository for your documents.

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 `$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).

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 5.1 and higher now 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.

Since at the time of this writing there is only one implementation of the pluggable enterprise search, we will cover how to implement this using *Solr*.

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>). *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 *catalina.sh* or *catalina.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` vari-

able:

```
-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 that there are two entries which define where the Solr server can be found by Liferay:

```
<bean id="indexSearcher" class="com.liferay.portal.search.solr.SolrIndex-  
SearcherImpl">  
    <property name="serverURL" value="http://localhost:8080/solr/select"  
</bean>  
<bean id="indexWriter" class="com.liferay.portal.search.solr.SolrIndexWri-  
terImpl">  
    <property name="serverURL" value="http://localhost:8080/solr/update"  
</bean>
```

Modify these values 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 coming 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. If you wish to have a shared index, you will need to either share the index on the file system or in the database.

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 home folder of the user that is running Liferay:

```
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 4.3.1 and higher 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 invalid-

ated 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 extension environment 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/classes/ehcache*. The files should be postfixed with *-ext.xml*. 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.RMIBoot-
strapCacheLoaderFactory" />
</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, by default you are using 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 User Home>/liferay/jack-rabbit`. 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

The workflow portlet allows a user to define any number of simple to complex business processes/workflows, deploy them, and manage them through a portal interface. The power of this portlet is that it allows users to create forms-based data entry applications that are fully integrated with Liferay's permissions system. They have knowledge of users, groups, and roles without writing a single line of code—it only requires the creation of a single XML document.

The portlet relies on Mule or Apache ServiceMix to function as an Enterprise Ser-

vice Bus (ESB) that acts as a broker between the portal and a workflow engine. Essentially, the portal provides a generic interface through which workflow services are requested via normal HTTP calls. The requests are routed through the ESB which in turn calls a workflow engine implementation that the user has defined in the ESB configuration. By default, Liferay provides an implementation of JBoss' jBPM workflow engine.

Installation and Test

To enable workflow in the portal, you will have to install three plugins: the workflow portlet itself, the jBPM web plugin, and one of the ESB web plugins. The latter two are conveniently located on the *web* tab of the *Plugins Installation* section of the Control Panel. You will need to have jBPM and *one* of the ESB plugins installed.

Once you install the plugins, follow these steps to get everything configured properly:

1. For Mule, make sure the following property in *portal-ext.properties* is set to the following:

```
jbi.workflow.url=http://localhost:8080/mule-web/workflow
```

For ServiceMix, make sure the following property in *portal-ext.properties* is set to the following:

```
jbi.workflow.url=http://localhost:8080/servicemix-web/workflow
```

2. Restart your application server if you needed to change the above property.
3. Login to the portal as the portal administrator. The default credentials are *test@liferay.com/test*.
4. Add the *Workflow* portlet to a page.
5. Click on the *Definitions* tab.
6. Click on the *Add* button.
7. Copy and paste the contents of *jbpw-war/WEB-INF/definitions/datatypes_definition.xml* into the text area and click the *Save New Version* button.
8. Click on the *Add Instance* icon.
9. From the *Instances* tab, click on the *Manage* icon next to *Enter data*.
10. Fill out the form and click the *Save* button; alternatively, you can test the various error checking capabilities by inputting incorrect values and clicking the *Save* button.
11. Eventually, enter correct values and click the *Save* button.
12. From the *Instances* tab, click on the *Manage* icon next to *View Data*.
13. Confirm that all the data was entered correctly and click the *Finished* button.
14. Confirm that the instance is now in the *End* state.

Using Different Databases

The default implementation of jBPM uses an HSQL database found in *jbpm-web.war/WEB-INF/sql/jbpm.**. To change the location of the HSQL database, change the value of the *hibernate.connection.url* property in *jbpm-web.war/WEB-INF/classes/hibernate.cfg.xml*. The location is relatively addressed from wherever the startup script for your server is located.

To use a database other than HSQL, first create the database schema using one of the SQL create scripts supplied in the *jbpm-web.war/WEB-INF/sql* directory. Then uncomment the corresponding hibernate connection properties block in *jbpm-web.war/WEB-INF/classes/hibernate.cfg.xml*.

How the Workflow Portlet Works

Since the default implementation of the workflow portlet relies heavily on the capabilities of jBPM, this section gives a technical overview of jBPM and explains how it integrates into Liferay using jBPM Process Definition Language (JPDL) formatted XML files. It does not, however, give an in depth view of jBPM. For that, please refer to the jBPM user guide (<http://docs.jboss.com/jbpm/v3/userguide>).

PROCESS DEFINITIONS

Before the workflow portlet can be used, business processes must be defined. Business processes in jBPM are defined by XML documents known as *process definitions* which are written in jBPM Process Definition Language (JPDL). This XML format specifies entities such as the process roles (known as *swimlanes*), the various states in the process (known as *nodes*), the tasks associated with each node, the roles associated with each task, the transitions from one node to the next, the variables associated with each task's form, the external actions executed on entry or exit of a node, and many others. For an in depth understanding of process definitions and JPDL, refer to JBoss' jBPM user guide at the link above.

There are three sample process definition XMLs that are packaged with the portlet. They can be found in *jbpm-web.war/WEB-INF/definitions*. We used one in the quick start section above to create a sample workflow. An explanation of each is included below.

INTEGRATING WITH USERS, COMMUNITIES, AND ROLES

In JPDL, there is the notion of process roles called *swimlanes*. Swimlanes can be associated with Liferay users, communities, and roles via the *IdentityAssignmentHandler* class.

```
<swimlane name="approver">
  <assignment class="com.liferay.jbpm.handler.IdentityAssignmentHandler"
    config-type="field">
    <type>user</type>
    <companyId>10095</companyId>
    <id>10112</id>
  </assignment>
```

```
</swimlane>
```

In the XML above, the *approver* swimlane is associated with the Liferay user that has a User ID of 10112 and belongs to a Company ID of 10095. You can also associate a Liferay user with a swimlane by email address as shown in the following XML snippet.

```
<swimlane name="shipper">
  <assignment class="com.liferay.jbpm.handler.IdentityAssignmentHandler"
  config-type="field">
    <type>user</type>
    <companyId>10095</companyId>
    <name>test.lax.2@liferay.com</name>
  </assignment>
</swimlane>
```

In the XML above, the *shipper* swimlane is associated with the Liferay user that has an email address of “test.lax.2@liferay.com” and belongs to a Company ID of 10095.

```
<swimlane name="salesman">
  <assignment class="com.liferay.jbpm.handler.IdentityAssignmentHandler"
  config-type="field">
    <type>community</type>
    <companyId>10095</companyId>
    <id>3</id>
  </assignment>
</swimlane>
```

In the XML above, the *salesman* swimlane is associated with any Liferay user that belongs to a Community with the Group ID of 3 (which defaults to the *Support* community if you are using the embedded HSQL database that comes with the Liferay Portal bundles) and Company ID of 10095. In other words, the salesman swimlane is assigned to the pool of *Support* users. If one of these users were to manage a salesman task, he/she would automatically be assigned to all other salesman tasks in the workflow.

```
<swimlane name="accountant">
  <assignment class="com.liferay.jbpm.handler.IdentityAssignmentHandler"
  config-type="field">
    <type>community</type>
    <companyId>10095</companyId>
    <name>Support</name>
  </assignment>
</swimlane>
```

The XML above shows an alternative way to associate the *accountant* swimlane with the *Support* community using the actual community’s name. Since community names must be unique per Company ID, this format accomplishes the same results as the previous XML.

```
<swimlane name="user_admin">
  <assignment class="com.liferay.jbpm.handler.IdentityAssignmentHandler"
  config-type="field">
```

```
<type>role</type>
<companyId>10095</companyId>
<id>1001</id>
</assignment>
</swimlane>

<swimlane name="user_interviewer">
  <assignment class="com.liferay.jbpm.handler.IdentityAssignmentHandler"
config-type="field">
    <type>role</type>
    <companyId>10095</companyId>
    <name>User Interviewer</name>
  </assignment>
</swimlane>
```

The two XML snippets above are very similar to the Group XML snippets. Both associate their respective swimlanes with a role, but the first XML does so using the Role ID, and the second XML does so using the role's unique name.

DATA TYPES AND ERROR CHECKING

Currently, jBPM doesn't have support for variable data types. However, data types have been dealt with in the workflow portlet by incorporating them into the names of the controller variables. The table below shows the data types supported by the portlet as well as the syntax for the variable names:

Data Type	Syntax	Description
Check-box	checkbox:name:checkedValue	name: the caption next to the checkbox checkedValue = the value assigned to the variable if the checkbox is checked
Date	date:name	name: the caption next to the date selector object
Email	email:name	name: the caption next to the text input field
Number	number:name	name: the caption next to the text input field
Password	password:name	name: the caption next to the text input field
Phone	phone:name	name: the caption next to the text input field
Radio Button	radio:name:option1,option2,...*	name: the caption next to the radio buttons option1,option2,...* : a comma-delimited list of options that represent the different radio button options
Select Box	select:name:option1,option2,...*	name: the caption next to the select box option1,option2,...* : a comma-delimited list of options that represent the different options in the select drop-down
Text	text:name	name: the caption next to the text input field
Textarea	textarea:name	name: the caption next to the textarea input field

Note that for all name and option values, the values should be entered in the XML in lower case with hyphens used between words:

```
radio:are-you-hungry:yes,no,a-little-bit
```

In addition, you should register the corresponding display values in the *Language.properties* file:

```
are-you-hungry=Are you hungry?
yes=Yes
no=No
a-little-bit=A little bit
```

This will ensure that the values are displayed correctly in the portlet to the user.

By default, all variables are readable and writable by the user. Therefore, they can be defined as follows:

```
<variable name="textarea:comments" />
```

However, if variables should only be readable or writable, or if variables are required, these must be specified in the variable definition:

```
<variable name="text:name" access="read,write,required" />
<variable name="date:birthday" access="read" />
```

Variables of data type Date, Number, Email, and Phone are validated in the service call. Also, required fields are validated to ensure a value of some kind was submitted. If invalid values are submitted, the user is returned to the original form and error messages are displayed next to the invalid input fields.

Refer to the sample definition *jbpm-web.war/WEB-INF/definitions/datatypes_definition.xml* for examples of all data types in use in a single form.

JBoss also provides a graphical JPDL editor which is implemented as an Eclipse plugin. This editor allows you to design workflows graphically, rather than using an XML editor. You can find this tool at <http://www.jboss.org/tools>.

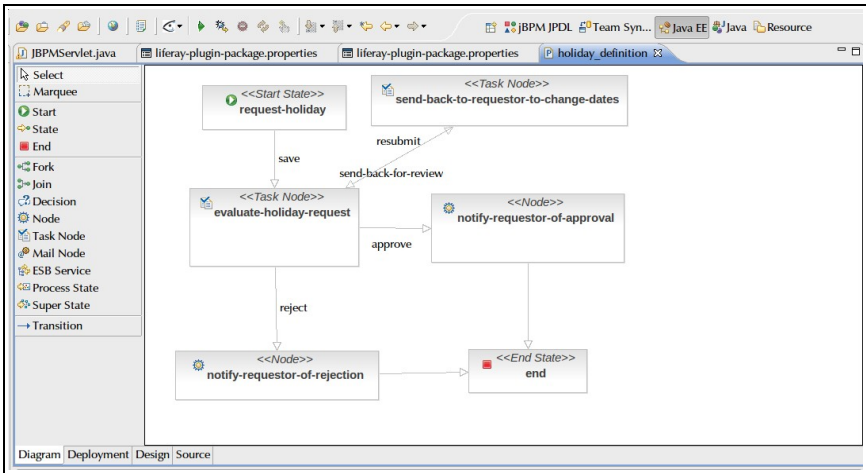


Illustration 80: The JPDL editor from JBoss

SAMPLE PROCESS DEFINITIONS

The best way to understand JPDL is to look over the 3 sample XML files included with the workflow portlet. They can be found in *jbpm-web.war/WEB-INF/definitions/*. Below is a quick summary of each:

datatypes_definition.xml – a good guide to follow to understand how to use each of the data types described in the section above

holiday_definition.xml – a simple workflow that allows an employee to make a holiday request with a start and end date, and then a manager can either approve, reject, or send the request back for review

websale_definition.xml – a more complex workflow that emulates an online auction site in which control of the workflow passes through various roles. It is the most complicated of the 3 workflows, but it demonstrates almost all of the BPM features offered by jBPM

Notes:

- The JPD L definition XMLs can be created through a graphical design tool offered by JBoss, but that is beyond the scope of this document (see <http://docs.jboss.com/jbpm/v3/gpd> for a detailed explanation) and is also beyond the scope of the portal.
- For nodes that have tasks associated with them, each of the variables in the controller will be converted to a form element that the user must update.
- For nodes that have tasks associated with them, each of the transitions out of the node is represented as a button in the task's form. The transition's *name* attribute should always be in lowercase with hyphens between words and registered in *Language.properties*. The display value is used as the button's name.
- Many of the action handler classes found in the *com.liferay.jbpm.handler* package are just place holders that output relevant text to the console. Conceivably, these classes could perform operations such as sending out emails, initiating batch processes, updating legacy systems, etc.
- The websale workflow demonstrates the following jBPM concepts, all of which are discussed in further detail in the jBPM user guide (<http://docs.jboss.com/jbpm/v3/userguide>):
 - Events
 - Beanshell scripting
 - Swimlanes
 - Tasks
 - Assigment(User/Pool)
 - Controlllers
 - Variables
 - Timers
 - Node Types
 - State
 - Task
 - Fork
 - Join
 - Decision
 - Transitions
 - Actions

WARNING MESSAGES

If you have warning messages turned on for your server, in your console you

may see some variant of the following message output several times when jBPM is called:

```
WARN [org.hibernate.engine.StatefulPersistenceContext] Narrowing proxy to
class org.jbpm.graph.node.TaskNode - this operation breaks ==
```

According to the following post on the JBoss forums (from Koen Aers, one of the key contributors to the jBPM project), this is not an error and is not of significance. He explains the reason this warning is thrown here: <http://www.jboss.com/?module=bb&op=viewtopic&t=73123>. Basically, the issue boils down to Hibernate doing lazy loading of the class. After a query, a collection is returned which holds a collection of stubs of the class. When a particular instance is retrieved, Hibernate goes and gets it from the database and replaces the stub with an actual instance, thereby breaking the == operator.

Administration

Once you have defined your business processes by successfully writing process definitions for them, the next step is to deploy your business processes. Once they are deployed, users can manage the life of each process instance as control is passed from one role to the next. This section is meant for end users who will actually be executing predefined process definitions.

DEPLOYING WORKFLOWS

Once the user logs in to the portal and adds the workflow portlet to her page, he or she will see something similar to the following:

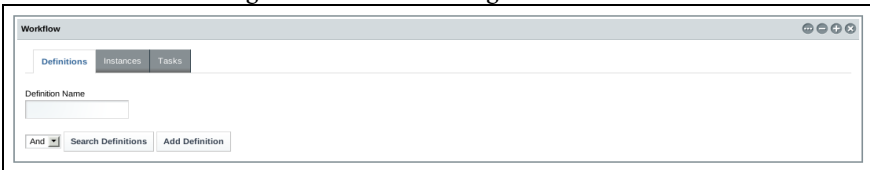


Illustration 81: Default view of the Workflow Portlet

If she clicks on the “Definitions” tab, he or she will see the following:

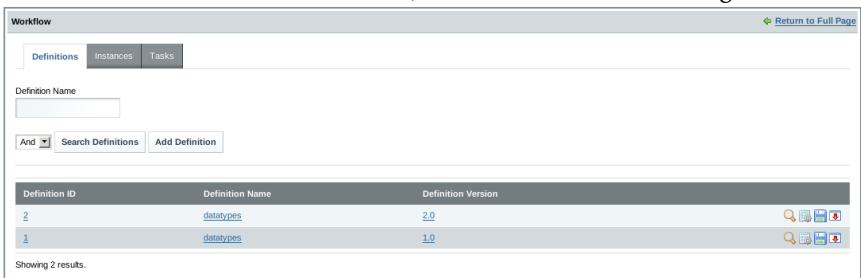


Illustration 82: Workflow Definitions

The *Definitions* tab displays all of the workflows that have been deployed in the system. To deploy a workflow, click on the *Add Definition* button. The user will see the following screen:

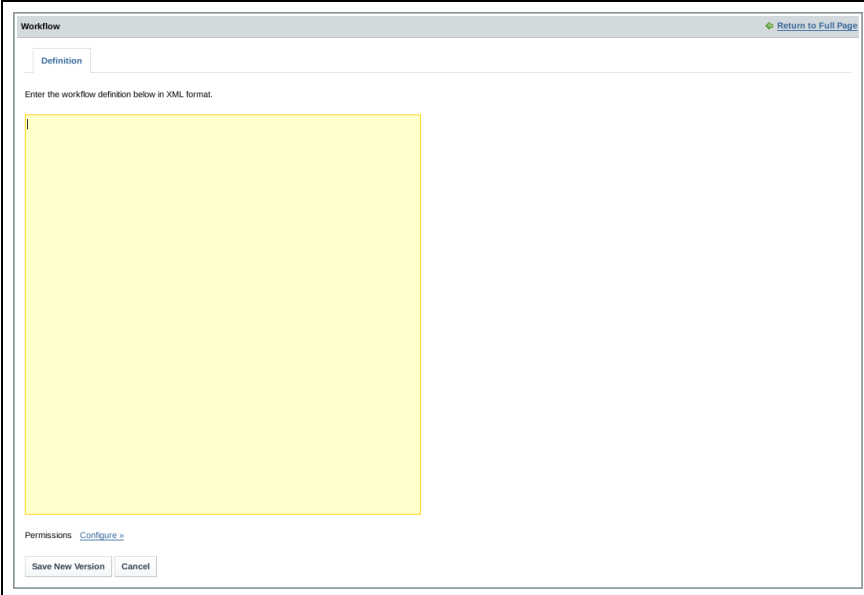


Illustration 83: Adding a Workflow Definition

At this point, the user can paste in the contents of a definition XML (see *jbpm-web.war/WEB-INF/definitions* for examples) and click the *Save New Version* button. If the XML is invalid, an error message will be displayed. If the XML is valid, it will be deployed, the user will be returned to the *Definitions* tab, and a success message will be displayed.

Because business processes may change over time, every version of the workflows is maintained. To edit an existing version, click on the *Edit* icon next to the definition name. Update the XML in the text area, and click the *Save New Version* button. The new version number will be incremented by 1 from the previous version. To start a new instance of a workflow definition, click on the *Add Instance* icon. A new instance will appear on the *Instances* tab. To view all the instances of a particular definition, click on the *View Instances* icon. Finally, the user can also search for a definition by name using the *Definition Name* input box.

MANAGING INSTANCES

After a definition is deployed and an instance of that definition is started, it is up to the user to manage the life cycle of the instance. Instance management is controlled from the *Instances* tab. Below is an example of what the user might see:

The *Instances* tab displays every instance of every version of every workflow deployed in the system. They are listed alphabetically by **Definition Name** followed by **Start Date** in descending order. The search form at the top of the screen allows the user to find specific instances to manage. In particular, the *Hide instances that have already ended* check box allows the user to display only active, running instances. The date ranges also allow the user to search by *Start Date* and/or *End Date* (**NOTE:** Date ranges are inclusive of the day. For example, if the *Start Date* range was set to January

Your request processed successfully.

Definitions Instances Tasks

Definition Name: Definition Version:

Start Date (Range): To:

End Date (Range): To: Hide instances that have already ended

And Search



Instance ID	Definition Name	Definition Version	Start Date	End Date	State
1	datatypes	2.0	5/8/09 6:03 PM	5/8/09 6:43 PM	End
2	websale	1.0	5/8/09 6:51 PM	N/A	perform-shipping-and-payment ship-item perform-shipping-and-payment

Showing 2 results.

Illustration 84: Workflow Instances

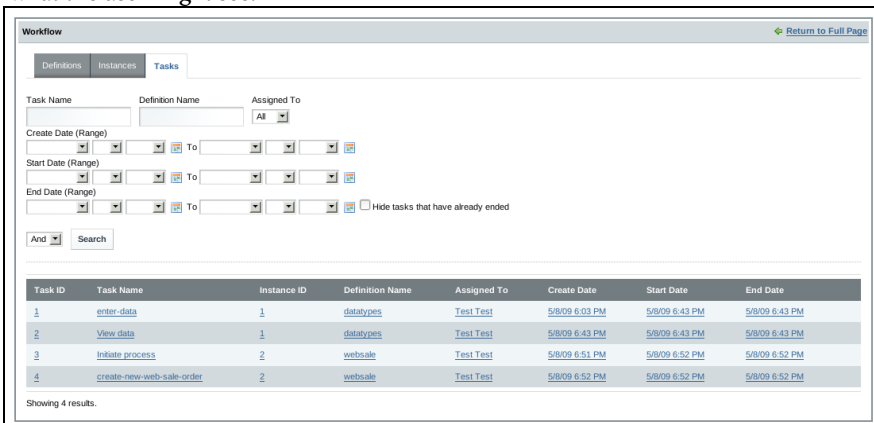
23, 2008 – January 23, 2008, then only instances that were started between January 28, 2008, 12:00am to January 28, 2008, 11:59pm would be displayed). The first row for each instance describes the state of the instance. Any subsequent rows in the instance define tasks associated with the current state. Often times, the current state and current task have the same name. In the example screen shot below, notice that web sale version 1.0 is currently in the “Perform shipping and payment” state, and it has two outstanding tasks associated with it – “Wait for shipment to be delivered” and “Wait for money.”

The right-most column in the results table displays what actions the current user can perform on the given instance in its current state. The table below shows all of the possible actions and what each means:

Action	Explanation
Blank	3 possibilities: <ul style="list-style-type: none"> ● The user doesn't have the appropriate role/swimlane to perform an action on the instance in its current state ● The user doesn't have permissions to perform an action ● The instance has already ended
Manage icon ()	The user directly has the appropriate role/swimlane to perform an action or the user belongs to a group which has the appropriate role/swimlane. If the user clicks on the "Manage" icon, she will be taken to a form which must be submitted to complete the task. See section 3.3 for more details
Signal icon ()	The instance is currently in a wait state and must be "signalled" to continue. Typically, signals come from eternal processes (e.g., the arrival of a package, the successful update of a legacy system, etc.) and are not manually entered by a user. However, in the case that user intervention is required, the "Signal" icon is available.
Waiting on sibling tokens to complete	This only occurs when the process has forked into multiple subprocesses. In order for the main process to continue, all of the subprocesses must complete. As each of the subprocesses completes, they will go into this state. Once all subprocesses complete, the main process will continue like normal.

MANAGING TASKS

Task management is controlled from the "Tasks" tab. Below is an example of what the user might see:



The screenshot shows the 'Workflow' application interface. At the top, there are tabs for 'Definitions', 'Instances', and 'Tasks'. Below the tabs, there are search filters for 'Task Name', 'Definition Name', and 'Assigned To'. There are also date range filters for 'Create Date (Range)', 'Start Date (Range)', and 'End Date (Range)'. A 'Search' button is present. Below the filters, a table displays the results of the search. The table has 8 columns: Task ID, Task Name, Instance ID, Definition Name, Assigned To, Create Date, Start Date, and End Date. There are 4 rows of data shown.

Task ID	Task Name	Instance ID	Definition Name	Assigned To	Create Date	Start Date	End Date
1	enter_data	1	datatypes	Test Test	5/8/09 6:03 PM	5/8/09 6:43 PM	5/8/09 6:43 PM
2	View data	1	datatypes	Test Test	5/8/09 6:43 PM	5/8/09 6:43 PM	5/8/09 6:43 PM
3	instate process	2	web-sale	Test Test	5/8/09 6:51 PM	5/8/09 6:52 PM	5/8/09 6:52 PM
4	create new web-sale order	2	web-sale	Test Test	5/8/09 6:52 PM	5/8/09 6:52 PM	5/8/09 6:52 PM

Showing 4 results.

Illustration 85: Workflow Tasks

The Tasks tab displays every task that has either been assigned directly to the

user or to the group/role pool that the user belongs to. They are listed by **Create Date** in ascending order, and the tasks assigned directly to the user are listed before the tasks assigned to the user's pool (if the **Assigned To** column is blank, that means the task is open to the pool). The search form at the top of the screen allows the user to find specific tasks to manage. In particular, the *Hide tasks that have already ended* check box allows the user to display only active tasks. The date ranges also allow the user to search by task *Create Date*, *Start Date*, and/or *End Date*. The user can also choose to only display tasks assigned directly to her, tasks assigned to her pool, or all tasks assigned to either by using the *Assigned To* drop-down.

To start a task, all a user needs to do is click on it. The next screen that is shown will be different depending on the type of task that has been defined.

For example, if a user was using the sample *request-holiday* definition which has been provided, a form similar to the following will be displayed:

Illustration 86: Workflow Form

These task forms are generated from the control variables associated with the task and defined in the definition XML. Depending on the data type, the corresponding element is created in the form. Required fields are denoted by a red asterisk. If the user submits the form with invalid data, she will be returned to the form with error messages next to each of the invalid fields. If all data is valid and the user submits the form, she will be returned to the *Tasks* tab with a success message displayed.

Future Enhancements

LOGGING

Currently, the workflow portlet has no notion of logging other than the ability to review all of the tasks that a user has assigned to them or has completed. However, jBPM provides rather robust logging functionality so administrators/users can monitor every action that has ever been taken in a particular workflow.

The only reason logging functionality has not been built out in the current release is because the Liferay development team is not sure what the most effective logging metrics would be to the end user. If you or your organization has logging requirements, please submit them to the Liferay Forums, and we will review those requirements for possible inclusion in future versions of the Workflow portlet.

CUSTOMIZABLE FRONT-END

Though the workflow portlet's strength is that it can provide a forms-based data

entry application virtually on-the-fly, it is obvious that there is not much control over what the forms look like when they are rendered by the portlet. To address this concern, the Liferay development team plans to create style sheets and templates that can be applied to the vanilla forms. The functionality would be very similar to how XSL style sheets are currently applied to Journal Articles in the Liferay Journal Content Management System. This enhancement would give organizations flexibility in layout and UI design of their forms.

FILE UPLOAD DATA TYPE

There have already been several requests to add a file data type to provide a means for users to upload files that are associated with workflow tasks. This will definitely be a future enhancement.

Frequently Asked Questions

HOW DO YOU WRITE A NEW PROCESS DEFINITION?

The best way to learn how to write a new process definition is to use one of the sample definition XMLs (found in *jbpm-web.war/WEB-INF/definitions/*) as a starting point. In particular, *websale_definition.xml* demonstrates most of the BPM features offered by jBPM. For an exhaustive explanation of JPDL, visit JBoss' documentation at <http://docs.jboss.com/jbpm/v3/userguide/jpdl.html>. There is also a graphical JPDL designer available at <http://labs.jboss.com/jbossjbpm/downloads>.

WHY ARE THERE “DUPLICATE FILE” EXCEPTIONS WHEN I CHANGE DATABASES FOR jBPM?

Since we are using ServiceMix as the service broker for our workflow implementation (by default, we are using jBPM), we cannot rely on the workflow engine to maintain versions of our process definitions. Therefore, we maintain the process definition XMLs as system documents in our document library. The XMLs are named based on their definition IDs, and the definition IDs are maintained in the jBPM database. Therefore, if you were to switch databases to a new instance, the definition IDs would also be reset, and when the system tries to store the process definition XML, it will find a duplicate XML already exists. The only way to ensure that this exception does not occur is by clearing out the *<User Home>/liferay/jackrabbit* folder before switching databases. However, be warned that this will delete ALL the files that are stored in your Document Library. It is recommended that once you decide on a jBPM database that suits your needs, you should only use that database.

Deploying A Customized Liferay

As described in the *Installation* chapter of this book, Liferay allows for complete customization of the portal through the Extension Environment. Deploying the extension environment to a server requires one of two scenarios:

- The Liferay development tools (JDK, Ant, etc.) are installed on the server, the Liferay Portal source code is available on the server, and the extension

environment is checked out to a directory on the server.

- On a client machine which contains the Liferay development tools, a drive can be mapped or a folder mounted which points to the installation directory on the server.

Once one of these two requirements have been met, deploying to the server becomes as easy as deploying locally to the developer's machine.

Deploying Directly on the Server

Deploying the extension environment directly on the server is the recommended method. To do this, you will have to create two new configuration files which define the settings for the deployment. In the extension environment, create a file called `app.server.<username>properties`, where `<username>` is the user name of the account under which the server executable runs. If, for example, you have a Glassfish server running under the user name of `glassfish`, your file would be called `app.server.glassfish.properties`. This file will override the default values which are found in `app.server.properties`. You will need to configure two properties in this file: the server type and the server path.

The server type should be one of the following:

```
app.server.type=geronimo-tomcat
app.server.type=glassfish
app.server.type=jboss-tomcat
app.server.type=jetty
app.server.type=jonas-jetty
app.server.type=jonas-tomcat
app.server.type=resin
app.server.type=tomcat
```

The path property is similar to the server type. It should look like this:

```
app.server.<server name>.dir
```

Replace `<server name>` with the server type above. For example, if you are using Glassfish, your property would be:

```
app.server.glassfish.dir=/home/glassfish/glassfish-v2
```

The value of the property should be the fully qualified path to the server directory.

Next, create another file similar to the first one called `release.<username>.properties`. Again, substitute `<username>` with the user name the server runs under and under whose credentials you will be doing the deployment. This file will override the default values found in `release.properties`.

This file requires two properties:

```
lp.source.dir
lp.ext.dir
```

Set the value for the `lp.source.dir` property to be equal to the fully qualified directory name for where you have installed the Liferay Portal source. Set the value for the

lp.ext.dir property to be equal to the fully qualified directory name for where you have installed the Extension Environment you have checked out from your source code repository. For example:

```
lp.source.dir=/home/glassfish/lportal/portal
lp.ext.dir=/home/glassfish/lportal/ext
```

Once you have set up these two properties files, run the following Ant task:

```
ant deploy
```

Your customized Liferay will be automatically compiled and deployed to your application server.

Deploying from a Client Machine

If you will be deploying a customized Liferay from a client machine, you will need to map a drive (on Windows) or a folder on your file system (Mac or Linux) to the server. Once you have done this, follow the same procedure outlined above, with the exception that the *<username>* should be the user name of the user logged in to the client, not the user name on the server. You will not need to change the *release.<username>.properties* file; only the *app.server.<username>.properties* file will need to be modified.

If you are using a developer's machine to do the deployment, these configuration files will already exist. Modify the *app.server.<username>.properties* file to match the application server type and location of the directory in which it is installed from your mapped drive or folder. Then run the above Ant task to deploy the extension environment.

Note that this second method is not a best practice, as it enables changes to be made locally on a developer's machine which can then be deployed directly to a server without source code management being done first. It is better to have developers check in all their code, version that code, and then pull that version from your source code management software to deploy it to a server.

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.

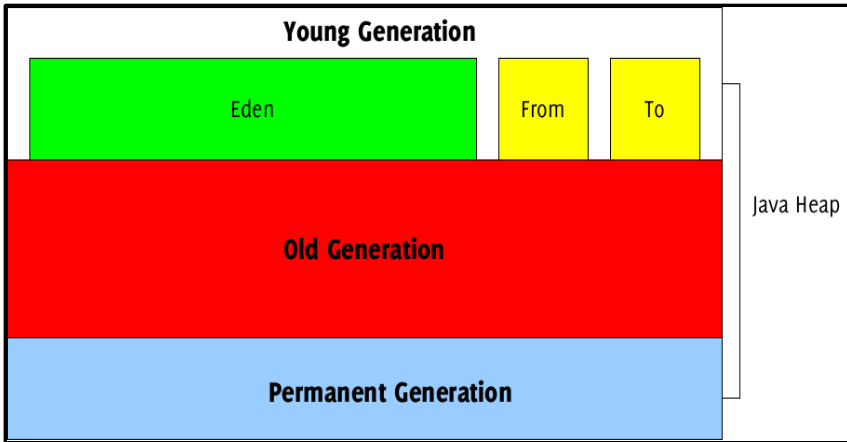


Illustration 87: Java memory

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. If you are not using the Virtual Hosting for Communities feature, comment out the Virtual Host 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 5.2.x 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/wsrp-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 *Manu-*

`alShardSelector` 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/wsrp-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.

7. 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

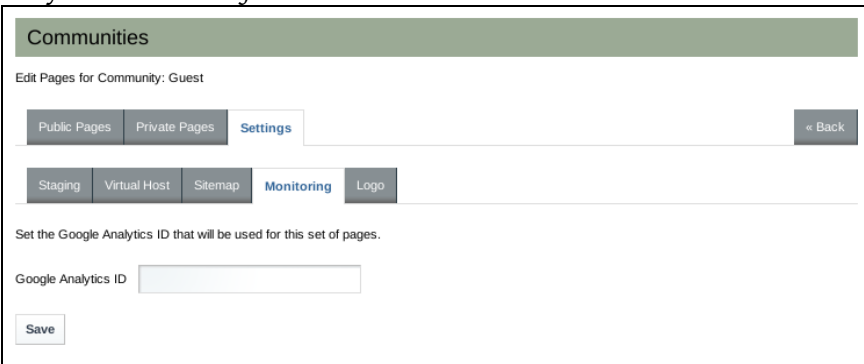
Liferay includes built-in support for Google Analytics, allowing administrators to make use of Google's tool set for analyzing site traffic data. When you sign up for Google Analytics, a snippet of code is provided which needs to be added to your web pages in order to allow Google's system to register 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.



The screenshot shows the 'Communities' management interface. At the top, there's a header 'Communities' and a sub-header 'Edit Pages for Community: Guest'. Below this are two rows of tabs: the first row has 'Public Pages', 'Private Pages', and 'Settings' (which is selected), and a '« Back' button; the second row has 'Staging', 'Virtual Host', 'Sitemap', 'Monitoring' (which is selected), and 'Logo'. Below the tabs, there's a text label 'Set the Google Analytics ID that will be used for this set of pages.' followed by a text input field labeled 'Google Analytics ID'. At the bottom left is a 'Save' button.

Illustration 88: 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 portlet or theme 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 Extension Environment, 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 1 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.

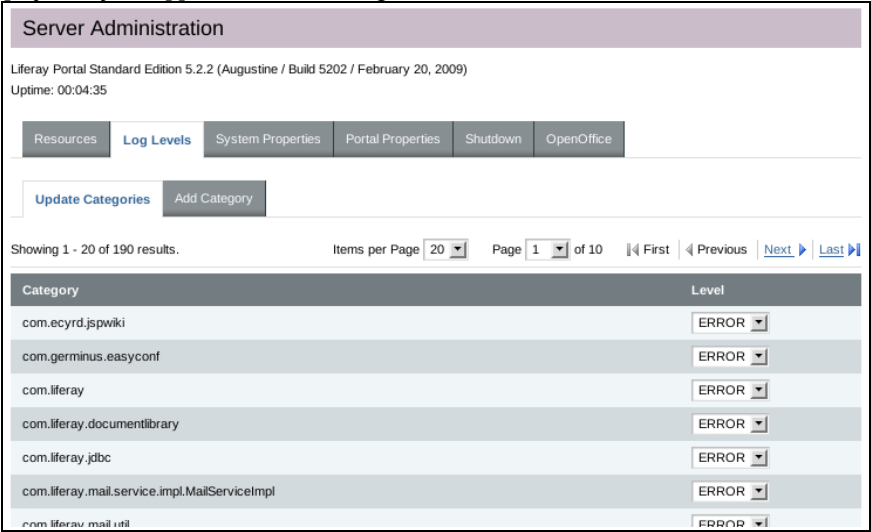


Illustration 89: Changing Logging Levels

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 working in the extension environment or have created a plugin and want to set the log level for one of your own classes, 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.

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.



Illustration 90: Adding a Logging Category

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 is possible that API changes in the new version will break your existing code. This, however, is usually fairly straightforward 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 changes are 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 4.3.x to 4.4.x, but not from 4.3.x to 5.0.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. So if you wanted to upgrade from Liferay 4.3.5 to Liferay 5.2.4, you would first run the upgrade procedure for 4.4.2, then 5.0.1, then 5.1.2, and then 5.2.4.

Liferay Upgrade Procedure

Liferay 4.3.0 and higher 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, 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 four steps to upgrade a standard Liferay installation:

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 1 or in Chapter 5 (if you have customized Liferay).
4. Start (or restart) your application server. Watch the console as Liferay starts: it should upgrade the database automatically. Review the *portal.properties* changes and re-configure your previous customizations as necessary, restarting when customizations are complete.

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.

What follows are instructions for upgrading for specific versions.

Upgrading Liferay 4.3 to Liferay 4.4

PREREQUISITE

In order to upgrade to 4.4.x, you must start at 4.3.0 or above. If you are using version 4.2.2 or below, please see the upgrade instructions on Liferay's wiki at <http://wiki.liferay.com>.

Follow the generic upgrade steps above. Make sure your Liferay .war and dependency jars are all the same version.

IF YOUR DEVELOPERS HAVE CUSTOMIZED LIFERAY

If you are deploying Liferay from an extension environment, ensure that your developers have:

1. Run Service Builder for each custom service.
2. Edited their custom `{Model}LocalServiceImpl` classes and make them extend from `{Model}LocalServiceBaseImpl`.

3. Run Service Builder again for each of the custom services after the previous change.

Many of the DTD references in the various *-ext.xml* files have changed (*liferay-portlet-ext.xml* in particular). Be sure to compare your `<!DOCTYPE` references with the main file to insure they reference the same DTD.

EXAMPLE:

Liferay 4.3.x:

```
<!DOCTYPE liferay-portlet-app PUBLIC "-//LiferayDTD Portlet Application
4.3.0EN" "http://www.liferay.com/dtd/liferay-portlet-app_4_3_0.dtd">
```

Liferay 4.4.x:

```
<!DOCTYPE liferay-portlet-app PUBLIC "-//LiferayDTD Portlet Application
4.4.0EN" "http://www.liferay.com/dtd/liferay-portlet-app_4_4_0.dtd">
```

Upgrading Liferay 4.4 to Liferay 5.0

PREREQUISITE

In order to upgrade to 5.0.x, you must start at 4.4.0 or above.

Follow the generic upgrade steps above. Make sure your Liferay *.war* and dependency *.jars* are all the same version.

IF YOUR DEVELOPERS HAVE CUSTOMIZED LIFERAY

If you are deploying Liferay from an extension environment, ensure that your developers have upgraded their Service Builder *.xml* files to the new DTDs.

The generated code is slightly different, and custom written finders and *LocalServiceImpl* methods need to be adopted. The *system-ext.properties* and *portal-ext.properties* files have moved from *ext-impl/classes* to *ext-impl/src*.

CONVERTING WIKI PAGES (OPTIONAL)

If you were using the wiki portlet, you will be happy to find many new features and improvements in 5.0—and everything is backwards compatible.

You may wish to convert your existing pages from the old Classic Wiki syntax to the new Creole (<http://www.wikicreole.org>) syntax because it's more powerful and easier to learn and use. It may also be more familiar to your users if they have experience with other popular wikis like MediaWiki or Confluence. To that end, Liferay 5.0 includes an automatic translator that will convert all of the pages in the database to Creole. To run it, edit *portal-ext.properties* and set:

```
verify.processes=com.liferay.portal.verify.VerifyWikiCreole
verify.frequency=-1
```

Start the server and let the process do the whole translation for you automatic-

ally. Remember to change those properties back to their original state before the next startup; otherwise the process will run every time you start Liferay.

UPGRADE TROUBLESHOOTING

- The parameter *p_p_action* is now called *p_p_lifecycle*. This has to be adapted for example in the FriendlyURLMappers if you want them to trigger a *processAction()*.
- You need to copy *parent-build.xml* in *ext-impl* from our Subversion repository; otherwise, some files will be missing when deploying.
- Upload in Struts portlets is broken. See <http://support.liferay.com/browse/LEP-6412> and <http://support.liferay.com/browse/LEP-6479>.

Upgrading Liferay 5.0 to Liferay 5.1

CHANGES IN CONFIGURATION PROPERTIES

Because of changes in the default configuration, Liferay now comes with several configuration files containing the defaults for previous versions. If the portal is not operating the way you are used to, it is likely that this is because a default value has been changed. You can revert to the previous behavior by following the process below.

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 pass the following system property when running Liferay:

```
java ... -Dexternal-properties=portal-legacy-5.0.properties
```

Each application server has different methods to add this system property. In Tomcat, you would modify *catalina.sh/catalina.bat* or *catalina.conf* (depending on the exact version you are using).

WHAT HAS BEEN CHANGED?

Here is a description of the most significant changes. Check the file *portal-legacy-5.0.properties* for a full list of changed properties.

- **layout.user.private.layouts.power.user.required:** It's no longer required to be Power User to have a personal community. This new property allows falling back to the old behavior.
- **permissions.user.check.algorithm:** the default value is now a new algorithm that is much faster.

IF YOUR DEVELOPERS HAVE CUSTOMIZED LIFERAY

Following is a list of API changes. If your developers used the extension environment to develop custom code review the following items and adapt your code accord-

ingly:

- Several classes have been moved from `portal-impl` to `portal-kernel` to make them available to plugins. If you were using any of those classes you will have to change the import package.
- The `JSPPortlet` class has been moved to `util-bridges` so that it can be used from plugins. In order to adapt to this change do the following:
 - Any references to `com.liferay.portlet.JSPPortlet` need to be changed to `com.liferay.util.bridges.jsp.JSPPortlet`
 - Check that the paths to your JSP files in `portlet.xml` are absolute paths (from the docroot). For example, if your `view.jsp` lives in `docroot/html/`, set your `view-jsp` to `/html/view.jsp`.
- Error handling has been changed in order to provide a better end-user experience. The following construct:

```
catch (Exception e) {
    req.setAttribute(PageContext.EXCEPTION, e);
    return mapping.findForward(ActionConstants.COMMON_ERROR);
}
```

should be replaced with

```
catch (Exception e) {
    PortalUtil.sendError(e, request, response);
    return null;
}
```

Note that you can also optionally include an `HttpServletResponse` code in the `sendError` method.

UPGRADING THEMES

There were a few changes between the Classic theme in 5.0 and the Classic theme in 5.1. However, these changes were predominantly CSS only changes, but if you built your theme from the plugins directory, and used the `_diffs` directory and placed your CSS changes in `custom.css`, you may need to make a few adjustments to your CSS.

NEW CSS FILE

There is a new file which your developers will need to include a file called `application.css`. The list of these files is located in `/css/main.css`. At the top, right after `base.css`, you would add this line:

```
@import url(application.css);
```

This file includes all of the styling that is used for application elements, such as dialogs, inline popups, tabs, tags, and other elements.

Some rules were added (that may need to be overwritten for your theme) and some rules were removed (and may need to be re-added). If however, your theme is already built, and you edited the files directly, for the most part, you won't be affected. Overall, most changes are minor, but there are a couple that could cause confusion. These are covered below.

THE PARENT ELEMENT OF THE DOCK MAY CHANGE POSITIONING WHEN UPGRADING

In 5.1 a change was made to the Dock JavaScript which is a fix, but older themes may rely on the “buggy” behavior. Essentially, the CSS positioning of the Dock's parent was always set to *relative*; however, there are often times when a theme requires that it use a different positioning mechanism. The script now looks to see what the theme developer has specified and uses that. There is, however, one caveat. Sometimes, you absolutely must not have a positioned parent at all. The script, however, needs a positioned parent in most cases, and will apply one whenever the CSS position is set to *position: static*, since that is what the browser will return if no positioning is applied. So the problem in that situation is that if you must have *position: static*, the script will always overwrite it. In order to account for this, checking has been added so that if your CSS selector has these two rules:

```
position: static;
top: 0;
```

then it will use static positioning. The thinking is that because setting *top: 0* on a statically positioned item has no effect visually, we use it as a trigger to say that you *really* want to use static positioning. However, it now will allow you to define other positioning (absolute, relative, or fixed) without worrying about your defined styling being overwritten.

THE CLASS NAMES FOR DIFFERENT UI COMPONENTS HAVE CHANGED

One of the only changes that will have a real impact is that the style *.popup* for the inline popups has been changed to *.ui-dialog*. Other class name changes have been made, but have been left on the elements for now so that breakages will be kept to a minimum. However, default styling for those class names have been removed from the Classic theme.

The class names that have been changed are as follows:

- *.portlet-section-header* is now *.results-header*
- *.portlet-section-body* is now *.results-row*
- *.portlet-section-body-hover* is now *.results-row.hover*
- *.portlet-section-alternate* is now *.results-row.alt*
- *.portlet-section-alternate-hover* is now *.results-row.alt.hover*
- *.popup* is now *.ui-dialog*
- *.tabs* is now *.ui-tabs*
- *.tag* is now *.ui-tag*
- *.autocomplete-box* is now *.ui-autocomplete-results*
- *.drag-indicator* is now *.ui-proxy*

CHANGE IN THEME CSS FAST LOAD

In 5.0.1, the default setting for *theme.css.fast.load* was *false* which means that when developers edited the deployed CSS files, the changes would take effect immediately on the next page load. Now, for the sake of better *out of the box* performance, the default in 5.1 has been changed to *true*. Of course this means that when developers edit

the deployed CSS files directly, they will *not* see the changes take effect immediately on the next page load. Instead, developers should make the change in the pre-deployed theme and run the deploy process on the theme for the changes to be bundled and packed into the *everything_packed.css* file.

This of course might make it harder for theme developers to do real-time testing. So, the solution to get the old behavior is simply to revert *theme.css.fast.load* to *false* and restart the portal.

CHANGE IN JAVASCRIPT FAST LOAD

For the same reasons as described above, a change was made to *javascript.fast.load* from *false* to *true*.

Developers may want to revert it to *false* for the same reason.

UPGRADING PHP PORTLETS

Some changes were made to the handling of PHP portlets, including the addition of new init-param values (see: <http://support.liferay.com/browse/LEP-6465>). Also, major changes to *util-java* classes (such as the elimination of *StringMaker*, for example) requires that some of the JARs in *WEB-INF/lib* be updated.

To upgrade an old PHP portlet, developers should first make a back-up of the existing portlet. Then the PHP portions (including HTML, images, CSS, javascript), should be zipped and re-deployed through the Plug-In Installer. Liferay, as usual, will inject the needed files, including everything in the *WEB-INF* directory, the *Quercus* libraries, and so forth.

If your developers are uneasy about this approach, you can also deploy an empty *index.php* file in a ZIP through the Plug-In Installer and then use the result to do a “DIFF” of your existing portlet with the injected portions of the new portlet. Then the parts that need to be changed can be manually updated in the existing portlet. This might useful, for example if you want to make sure the ID of your portlet does not change (especially if you have many instances scattered throughout your site), or if you've been using the *liferay-plugin-package.xml* file to keep track of versions and compatibility.

JAVASCRIPT CHANGES

The Javascript in Liferay has undergone a refactoring to use the jQuery UI engine, upgrading jQuery to the latest version (version 1.2.6 in Liferay 5.1), and removing the *Interface* library. We've also removed and changed many of our methods that were polluting the global name space and/or were not being used anymore.

CHANGED METHODS

addPortlet, addPortletHTML: These have now been changed to *Liferay.Portlet.add()* and *Liferay.Portlet.addHTML()*, respectively.

ShowLayoutTemplates: This is now *Liferay.Layout.showTemplates()*.

StarRating, ThumbnailRating, Tooltip, Tabs: These are now `Liferay.Portal.StarRating()`, `Liferay.Portal.ThumbnailRating()`, `Liferay.Portal.Tooltip`, and `Liferay.Portal.Tabs` respectively.

Element.disable: Please use `Liferay.Util.disableElements()`.

Element.remove: Please use `jQuery(selector).remove()`.

Viewport.frame, Viewport.scroll, Viewport.page: These are now `Liferay.Util.viewport.frame()`, `Liferay.Util.viewport.scroll()`, and `Liferay.Util.viewport.page()`, respectively.

REMOVED JAVASCRIPT METHODS

AjaxUtil, AjaxRequest, and loadPage: Please use `jQuery.ajax` or any of the other `jQuery` AJAX methods.

LinkedList, NavFlyout, DragLink, PhotoSlider, PortletHeaderBar: No longer needed.

Coordinates, Coordinate, MousePos: No longer needed. If you need mouse coordinates during an event (such as `onMousemove`, `onMouseover`), or if you're attaching events with `jQuery`, you can reliably use the `event.pageX` and `event.pageY` properties, like so:

```
jQuery(window).mousemove( function(event){ var x = event.pageX; var y = event.pageY; } );
```

Liferay.Util.toJSONString, Liferay.Util.toObject: No longer needed. You can use `jQuery.parseJSON(str)` and `jQuery.toJSON(obj)` respectively.

Liferay.Util.getSelectedIndex: No longer needed, as it was being used to select the “checked” item in a group of radio buttons. To do the same thing, you would do something like:

```
jQuery(Element|Selector).filter(':checked');
```

If you absolutely must have the actual numerical index of the selected item, you could do:

```
var radioGroup = jQuery(Element|Selector);  
var index = radioGroup.index(radioGroup.filter(':checked')[0]);
```

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 1.

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 rewritten as follows:

```
ServiceContext serviceContext = ServiceContextFactory.getIn-  
stance(BlogsEntry.class.getName(), portletRequest);  
  
BlogsEntryServiceUtil.addEntry(..., serviceContext);
```

8. 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 copy-rightable 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 jurisdic-

tions 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 administers 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 endorsement 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 compliance 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/>.

9. COLOPHON

The text and layout were accomplished using OpenOffice.org 3.0.1 running on Mandriva 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 that were taken by the author were taken using Ksnapshot, an excellent screen shot program that is part of the KDE desktop, version 4.2.2. Other screen shots were taken by the contributors using unknown software. Drawings inside the book (such as the Unbreakable Liferay diagram) were created in OpenOffice.org draw, using some stencils from Dia (<http://www.gnome.org/projects/dia>). Some drawings and screen shots were touched up using the GIMP.

The cover of the book was done professionally this time by Liferay's graphics design team.

The picture on the front cover is a picture of Crepuscular Rays. These are rays of sunlight which appear to spring from a hidden source—usually from behind a cloud,

but can also appear diffused through trees. The rays are actually parallel, and it is perspective that makes them appear to converge at the Sun. Also known as God's rays, crepuscular rays can be spectacular and certainly seemed appropriate for the cover of a book about a product called *Liferay Portal*. This great shot was taken by none other than Liferay's CEO, Bryan Cheung, in Langen, Germany, near the offices of Liferay's European headquarters.

Rich Sezov, May 2009

A

Admin. 1, 2, 4, 5, 7, 9, 11, 12, 27, 28, 30, 42, 53, 59, 64, 67, 70, 77-82, 84-88, 90-102, 104, 111, 113, 114, 117, 119, 120, 134, 135, 137-140, 146, 152, 153, 158, 160, 165-170, 174, 175, 179, 190, 197, 198, 205, 206, 213, 223, 226, 230, 233, 235, 237, 244, 246, 247, 254, 260, 262, 267, 271, 283-285, 302

AJAX.....17, 209, 294

App.server.properties.....273

Authentication pipeline.....6, 179

Auto deploy.....6, 47, 150, 152, 233-235

Auto login.....6, 181, 182

B

Back up.....235, 283, 285

Blogs.....5, 8, 12, 93, 112, 119, 123-129, 137, 146, 180, 197, 208, 214, 217, 297

Bundle.....3, 11, 15, 19-25, 75, 132, 225, 232, 251, 262, 278, 293

C

Caching. .8, 46, 157, 202, 208, 211, 249, 256-

259

Calendar.....5, 8, 12, 17, 81, 91, 95, 96, 112, 119, 123, 129-131, 146, 180, 197, 214, 215

Captcha.....6, 46, 185

Cas....4, 6, 21, 23, 50, 93, 103, 105, 109-111, 133, 140, 148, 160, 161, 177, 179, 182, 202, 203, 207, 223, 229, 233, 236, 241, 244, 247, 254, 256, 258, 264, 266, 270, 278, 284, 285, 287, 292, 297, 300, 301, 303

Chat.....5, 12, 119, 131, 132, 194, 214

Clustering...8, 9, 11, 199, 202, 211, 249, 250, 256-259

Communities.....4, 8, 9, 81-84, 87, 89-92, 99, 100, 102, 111, 112, 116, 119, 120, 167, 198, 215, 220, 226, 237, 261, 278, 284

Community....4, 15, 16, 18, 78-84, 89-92, 98-100, 119, 120, 123, 124, 126, 128-130, 134, 135, 139, 144, 146, 147, 153, 166, 168, 169, 180, 193, 194, 216, 220, 223, 228, 237, 242, 245, 246, 262, 284, 290, 307

Company ID.....262

Control panel. 4, 7, 12, 20, 77, 79-81, 84, 86, 87, 89-91, 93, 98, 101, 103, 105, 106, 108, 110, 112-116, 132, 139, 144, 147, 198, 227, 230, 234-237, 244, 254, 260, 280, 284-286,

296
Css..8, 10, 154, 210, 211, 216, 221, 277, 291-293, 296

D

Database.....3, 6, 8, 9, 12, 15, 17, 20-29, 31, 32, 35, 37, 40, 42-45, 47-54, 56-61, 67, 69, 72, 73, 103, 104, 107, 108, 114, 116, 120, 147, 149, 150, 158, 160, 174, 179, 183-185, 196, 200, 202, 203, 242, 244, 246, 250-253, 255, 256, 258, 259, 261, 262, 267, 272, 275, 278-280, 285, 287-289, 295

Document Library...8, 16, 91, 115, 206, 215, 216, 245, 246, 251, 258, 259, 272, 285

E

Ehcache..6, 17, 157, 160, 202, 210, 249, 256-259, 285

Email.....4, 5, 13, 77-79, 84, 85, 88, 90, 102, 103, 105, 106, 112, 113, 116, 129-134, 136, 138, 142, 143, 145, 165-167, 170, 175, 176, 178, 179, 182, 190, 205, 206, 213-215, 217-220, 222, 224, 225, 262, 264-266

Extension environment...109, 227, 257, 272-274, 284, 286-290

G

Glassfish.....3, 24, 28-33, 151, 152, 232, 233, 273, 274

H

Hibernate.....6, 8, 17, 39, 46, 156-158, 160, 202, 257, 258, 261, 267, 275, 279, 281

Hot deploy. .6, 8, 47, 149, 151-154, 226, 227, 230-236, 251, 254, 256

HSQL.....20, 25, 26, 159, 200, 251, 261, 262

Html....31, 137, 144, 145, 158, 160, 161, 171, 193, 194, 197, 205, 208, 216, 219, 224, 272, 291, 293, 296, 307

I

Image Gallery.....8, 216

Invitation.....8, 197, 217

J

Jaas.....6, 35, 36, 38, 40, 51, 52, 174, 175

Jackrabbit....8, 9, 20, 203, 204, 234, 250-252, 255, 259, 272, 285, 295

Jar file.....48-50, 52, 109, 147, 148, 251, 253, 257

Javascript. 6, 10, 17, 161, 162, 164, 210, 244, 277, 278, 292-294

Jboss....3, 24, 36-42, 151, 152, 232, 260, 261, 265-267, 272, 273

JCR.....7, 149, 156, 203, 204, 215, 252, 279, 281, 286, 295

Jetty.....3, 24, 34-36, 45, 151, 273

Jonas.....24, 151, 273

Journal....8, 24, 111, 169, 181, 197, 202, 208, 217-219, 272, 303

Jsp.....153, 161, 186, 191-193, 197, 208, 224, 227, 291

JSR-168.....78, 225, 287

JSR-170....17, 20, 115, 203, 250-252, 259, 285

JSR-286.....15, 17, 78, 121, 225, 287

L

Language..3, 6, 17, 18, 78, 97, 113, 134, 135, 162, 170, 171, 190, 214, 217, 223, 247, 261, 264, 266

Language.properties. 190, 214, 217, 264, 266

LDAP.....4, 6, 77, 84, 102-110, 112, 116, 175-177, 179, 183

Liferay Portal. .9, 11-13, 15-17, 19-21, 24, 25, 27, 28, 34, 36-38, 40, 47-49, 58, 68, 69, 74, 75, 77, 89, 105, 108, 111, 115-117, 119, 124, 126, 148, 178, 225, 227, 230, 244, 249, 253, 259, 262, 272, 273, 283, 284, 308

Liferay-portlet.xml.....172, 173, 207, 278

Locations.....170, 196

Logging.....9, 35, 78, 123, 124, 127, 128, 133, 164, 173, 175, 187, 209, 246, 271, 283, 285, 286

Logs..5, 8, 12, 21, 79, 93, 103, 107, 112, 119, 123-129, 137, 146, 173, 180, 186, 197, 208, 209, 214, 217, 258, 267, 291, 297

Look and Feel...3, 6, 16, 119, 172, 221, 226, 230, 256, 296

Lucene...7, 8, 17, 20, 199-201, 234, 251, 253-255, 295

M

Mail. 4, 5, 7, 12, 13, 25-30, 32, 33, 35, 38-42, 45, 46, 48, 51, 53, 55, 57, 58, 63, 66, 68, 72-74, 77-79, 84, 85, 88, 90, 102, 103, 105, 106, 112, 113, 116, 119, 129-134, 136, 138, 142, 143, 145, 146, 156, 165-167, 170, 175, 176, 178, 179, 182, 190, 197, 204-206, 213-215, 217-220, 222, 224, 225, 262, 264-266, 279, 281, 295

Memory....9, 31, 36, 117, 158, 174, 186, 201, 211, 258, 259, 274, 275, 277

Message boards....5, 8, 12, 91, 92, 100, 113, 119-121, 133-142, 146, 219, 256, 258

My Places.....8, 79, 80, 220

Mysql....26, 27, 29, 30, 32-35, 37, 39-41, 43, 45-51, 54, 56, 57, 59, 68, 71-73, 159, 160, 200, 252, 255, 259, 278-280, 285

N

Navigation....8, 113, 145, 164, 198, 220, 221, 296

Ntlm.....4, 6, 110, 177, 178, 182, 210, 278

O

Ocaj.....3, 42, 44-46

OpenId.....4, 6, 111, 178, 182

OpenSSO.....4, 6, 111, 112, 178, 182

Organizations. 4, 6, 16, 19, 23, 77, 81-84, 87-90, 116, 120, 167, 169, 170, 184, 198, 206, 220, 226, 237, 246, 272, 283, 284, 295

P

Passwords....6, 101, 108, 175, 179, 183, 184, 246

Performance Tuning.....9, 274

Permissions...4-6, 15, 16, 23, 24, 82, 83, 85-87, 92, 98-102, 114, 120, 121, 126, 130, 135, 139, 141, 142, 145, 184, 185, 194, 240, 245, 259, 270, 287, 290

Plugin....3, 4, 6, 8, 17, 20, 23, 24, 27, 47, 78, 81, 101, 113, 116, 117, 132, 147, 152, 153, 199, 225-240, 242-244, 251, 253-256, 260, 265, 284, 286, 287, 291, 293, 295, 296

Portal-ext.properties..3, 5, 23, 25-28, 30, 33, 39, 41, 45, 46, 54, 57, 67, 73, 74, 93, 103, 107, 108, 112, 131, 144, 147, 148, 231-235,

245, 247, 251, 252, 255, 257, 258, 260, 277, 278, 280, 281, 285, 288, 289, 295

Portal.properties.....46, 108, 148, 245, 251, 252, 255, 257, 258, 288

Portlet.5-12, 15-17, 22, 47, 48, 58, 59, 77-81, 83, 84, 87, 89-93, 95-101, 112-116, 119-122, 124-135, 137-143, 145-147, 149-156, 162, 165, 172-175, 179, 180, 185, 188-191, 193-197, 202-209, 211, 213-246, 256, 258-261, 263-265, 267, 271, 272, 278, 279, 281, 284, 285, 287, 289-295, 297

Portlet.xml.....151, 153, 172, 173, 195, 207, 278, 291

R

Release.properties.....273

Request...6, 7, 55, 84, 90, 114, 166, 171-173, 181, 182, 185, 186, 194-196, 205, 210, 211, 215, 216, 218, 219, 221, 222, 225, 253-255, 260, 265, 271, 272, 275, 278, 291, 294, 297, 302, 303, 305

Resin.....3, 24, 47-50, 151, 152, 273

Roles....4, 6, 9, 38, 40, 77, 81-88, 90, 92, 98-100, 102, 112-114, 116, 119, 121, 126, 130, 135, 142, 167-169, 184, 185, 202, 223, 228, 259, 261, 265, 276

S

Schema.....6, 149, 153, 254, 259, 261

Service Builder.....6, 155, 244, 288, 289

Services Oriented Architecture.....8, 244

Servlet.....7, 9, 15, 24, 36, 37, 78, 148, 149, 151, 166, 173, 174, 177, 182, 186, 194, 208, 210-213, 216, 218, 221, 222, 225, 226, 245, 278, 291, 296, 297

Session....6, 25-30, 32, 33, 35, 38-42, 45, 46, 48, 51, 53, 55, 57, 58, 63, 66, 68, 73, 74, 113, 156, 157, 164, 165, 172-174, 181, 182, 185, 186, 195, 204-206, 209-211, 296

Shopping.....8, 133, 181, 197, 221, 222

Single sign-on. 4, 84, 102, 105, 109-112, 116, 227, 278

SiteMinder.....4, 6, 112, 178, 179, 182, 187

SMTP....26, 27, 30, 33, 35, 38, 39, 41, 42, 45, 48, 51, 55, 57, 58, 68, 73, 85, 133, 136, 204-206

Social Bookmarks.....7, 208

Software catalog.....8, 222, 236-240, 242-244
Solr.....114, 253-256
Spring. .6, 7, 17, 155-158, 160, 212, 226, 245, 254, 279, 281, 307
SQL.....6, 20, 24-27, 29, 30, 32-35, 37, 39-41, 43-51, 54, 56, 57, 59, 68, 71-73, 158-160, 164, 165, 200, 251, 252, 255, 259, 261, 262, 278-280, 285
SSO with MAC.....6, 182
Swimlane.....261-263, 266, 270

T

Tags....8, 39, 74, 90, 144, 146, 164, 181, 194, 222, 223, 226, 239, 243, 291
Theme.....6, 8, 10, 16, 22, 78, 113, 116, 144, 147, 150, 152-155, 161, 172, 188-190, 198, 208, 216, 225, 226, 229-231, 234, 236, 239, 256, 277, 283, 284, 287, 291-293, 295, 296
Time zone.....6, 113, 130, 170, 171
Tomcat....3, 19-21, 24, 50-52, 109, 110, 150-152, 174, 186, 231, 232, 235, 236, 247, 253, 257, 273, 290, 295

U

Upgrade. .6, 9-11, 23, 24, 149, 150, 236, 237, 241, 254, 287-290, 293-296
User groups...4, 77, 81-84, 87, 92, 93, 95-98, 102, 112, 116
Users.....4-7, 9, 11-13, 15, 16, 21, 25, 38, 40, 50, 74, 77, 79-93, 95-113, 115-117, 119, 122-128, 130-142, 144, 146, 161, 164-177, 179-181, 184, 185, 187-190, 194, 195, 197, 199, 202, 204, 205, 210, 216, 220, 226, 229, 236-238, 240, 241, 243-246, 251, 255, 256, 258, 259, 261, 262, 267, 271, 272, 278-280, 284, 289

V

Velocity.....7, 17, 208, 209, 211, 217-219

Virtual Host.....7, 66, 102, 116, 209, 211, 278

W

War file. 20, 27, 28, 30, 33-36, 38, 40, 41, 44, 54, 58, 69, 73, 74, 225, 230, 232, 234, 235, 238, 243, 253, 256, 288
WEB-INF....35, 41, 46, 50, 109, 148, 231, 233, 254, 257, 260, 261, 265, 268, 272, 293, 295, 296
Web.xml.....36, 151, 155, 172, 177, 178, 194, 234, 278, 296
WebDAV.....7, 17, 211-213, 216, 245
WebLogic.....3, 20, 52-58, 74, 151, 158, 232
Websphere..3, 20, 58, 59, 67-69, 73, 74, 151, 233-235
Wiki. 5, 8, 9, 12, 13, 59, 81, 91, 119, 141-146, 181, 182, 197, 203, 204, 223-225, 288, 289, 307
Workflow....3, 9, 16, 155, 203, 227, 249, 250, 259-263, 265-268, 271, 272
WSDL.....8, 247

X

Xml....31, 34-41, 48, 50-52, 74, 151-158, 160, 172, 173, 177, 178, 181, 194-196, 204, 207, 210, 216, 218, 234, 236, 237, 242-244, 252, 254, 257-266, 268, 271, 272, 278, 279, 281, 289-291, 293, 295, 296

Z

Zip. .21, 22, 24, 25, 38, 40-42, 48-50, 52, 210, 216, 229, 293, 296

.13, 29, 32, 37, 40, 48, 51, 63, 149-153, 156-158, 165-169, 172-174, 176-181, 183-189, 192, 194, 195, 197, 200-203, 208, 211-213, 215-219, 221-225, 233, 247, 252, 256-258, 262, 265, 266, 270, 272, 273, 289-291, 293, 294, 300, 308