

Liferay Portal 二次开发指南

作者：柯自聪 zcke0728@hotmail.com

So many open source projects.
Why not open your documents and share your experiences?

文档说明

参与人员：

作者	网名	联络
柯自聪	eamoi educhina	eamoi@163.com (技术) zcke0728@hotmail.com (版权)

发布记录：

版本	日期	作者	说明
1.0	2005-10-20	柯自聪	创建，第一版

链接：

类别	网址
Blog	http://www.blogjava.net/eamoi/
MSN-Space	http://spaces.msn.com/members/eamoi/

OpenDoc 版权说明：

本文档版权归原作者所有。

在免费、且无任何附加条件的前提下，可在**网络媒体**中自由传播。

如需部分或者全文引用，请事先征求作者意见。

如果本文对您有些许帮助，表达谢意的最好方式，是将您发现的问题和文档改进意见及时反馈给作者。当然，倘若有时间和能力，能为技术群体无偿贡献自己的所学为最好的回馈。

序	5
第一部分 Liferay Portal 架构解析	6
第一章 Liferay Portal	6
第一节 Portal 规范	6
1.1.1 JSR168	6
1.1.2 WSRP	6
第二节 什么是 Portal	7
1.2.1 Portal 服务器	7
1.2.2 Portlet 容器	7
第三节 什么是 Portlet	8
1.3.1 Portlet	8
1.3.2 Portlet 与 Servlet 的关系	8
1.3.3 Portlet 的生命周期	9
第四节 Liferay Portal 工作原理	9
1.4.1 Portlet 样式以及窗口状态	10
1.4.2 Portal 页面	11
第二章 Liferay Portal 的使用	13
第一节 Liferay Portal 安装	13
第二节 Liferay Portal 的用户策略	14
2.2.1 定义用户	14
2.2.2 添加用户	15
2.2.3 修改用户	15
2.2.4 定义用户组	18
2.2.5 新增、重命名用户组	19
2.2.6 修改用户组	19
2.2.7 定义角色	20
2.2.8 新增、重命名角色	21
2.2.9 修改用户组角色	21
2.2.10 定义 Portlet 的角色	22
第三节 Liferay Portal 内容和布局	24
2.3.1 什么是布局	24
2.3.2 什么是内容	26
2.3.3 内容布局与 Portlet 的关系	27
2.3.4 选择内容和布局	28
第四节 Liferay Portal 的桌面	28
2.4.1 什么是桌面	28
2.4.2 定义个性化的桌面	29
第五节 Liferay Portal 的品质	29
2.5.1 什么是品质	30
2.5.2 品质和 Portlet、Portal 的关系	30
2.5.3 定义个性化的品质	30
第六节 Liferay Portal 的部署描述文件	31
2.6.1 web.xml	31

2.6.2	portlet.xml	32
2.6.3	liferay-Portlet.xml	33
2.6.4	liferay-display.xml	34
2.6.5	liferay-layout-templates.xml	35
2.6.7	liferay-look-and-feel.xml	35
第二部分	Liferay Portal 二次开发	36
第三章	开发自己的 Portlet	36
第一节	重要的基类：GenericPortlet.....	36
第二节	Portlet 标签	37
3.2.1	defineObjects 标签.....	37
3.2.2	renderURL 标签.....	37
3.2.3	actionURL 标签	38
3.2.4	param 标签.....	38
3.2.5	namespace 标签	38
第三节	Portal 的对象	38
3.3.1	Request 对象.....	39
3.3.2	Response 对象	40
3.3.3	PortletConfig 对象.....	41
3.3.4	Session 对象	41
3.3.5	Preference 对象.....	43
第四节	编写自己的 Portlet 类	44
3.4.1	开发环境.....	44
3.4.2	准备工作.....	44
3.4.3	HelloWorldPortlet	45
3.4.4	HelloJSPPortlet	46
第五节	修改 Web 部署描述文件.....	48
第六节	创建 Liferay Portal 部署描述文件.....	49
第三部分	Liferay Portal 部署	53
第四章	部署自己的 Portlet	53
第一节	手动部署.....	53
第二节	Ant 自动部署.....	54
第三节	加入 Liferay Portal 自有列表.....	54
第四节	普通 Java Web 应用转化为 Portlet 应用	55
第四部分	附录.....	57
第五章	相关资源.....	57
第一节	资源网站.....	57
第二节	示例.....	57
第六章	参考资料.....	58
后序	58

序

随着信息化建设的深入，Portal 门户已经成为新型办公环境的一个重要组成部分。Portal 所提供的单点登录、权限控制、个性化定制、内容集成、文件管理等独特的功能，已经大大占据公众的眼球，并在信息集成和消除信息孤岛方面发挥了重要的左右。

随着 Portal 技术的成熟，以 MyNestcape、MyYahoo、MSN-Space 等为代表大型网站也较多的采用 Portal 架构来实现个性化的内容聚合和定制，以实现灵活的扩展的服务策略。

Liferay Portal 作为一个开源的 Portal 项目，利用 Hibernate、Struts、Spring 等开源框架，实现了 JCP JSR168 规范中提出的 Portal 功能，在开源 Portal 系统中有比较典型的代表性。

本文从 Liferay Portal 的架构入手，详细讲解 Portal 的用户策略、内容布局、桌面和品质的要素，引导读者完成 Liferay Portal 初步的二次开发，在 Liferay Portal 上定制自己的 Portlet。Liferay Portal 程序框架和源码分析不在本文的讨论范围。

第一部分 Liferay Portal 架构解析

本部分主要内容

Portal 服务器 Portal 容器 Portlet

第一章 Liferay Portal

作为一个开源 Portal 产品，Liferay Portal 提供对多个独立系统的内容集成，帮助多个组织实现更有效的合作。与其他商业的 Portal 产品相比，Liferay Portal 有着一系列的优良特性，而且不需要付费。

第一节 Portal 规范

随着 Portal 的兴起，越来越多的公司开始涉足 Portal 产品开发，并组建各自的 Portal 组件和基于其的产品，比如 IBM、BEA、MicroSoft、SAP、Apache 等。各个厂商的接口互不兼容，给软件开发商以及开发人员带来诸多不便。

1.1.1 JSR168

为此，JCP 组织发布了 JSR168(Java Specification Request)，Portlet Specification V1.0，用来提供不同的 Portal 和 Portlet 之间的互通性。只要开发的 Portlet 遵循 JSR168，则可以在所有遵循 JSR168 的 Portal 上部署运行。

JSR168 中定义了 Portal 的实现规范和接口，并对理想的 Portlet 进行了详细的规划和描述。

1.1.2 WSRP

WSRP 是 OASIS Web Service for Remote Portlet 的缩写。WSRP 是 Web Service 的一种新的商业应用，一种新的标准，主要用来简化 Portal 对于各种资源或者程序整合的复杂度，可以避免编程带来的整合麻烦和问题。而且 Portal 管理员可以从海量的 WSRP 服务中选择需要的功能用以整合到目前所用的 Portal 中。它有三种角色：

- 、生产者 → 提供 Portlet
- 、消费者 → 使用 Portlet
- 、终端用户 → 最终用户

它的特点在于生产者将消费者所需要的信息通过 WSRP 返回给消费者，这些信息是相对标记片断，例如 HTML、XHTML 等，可以直接嵌入用户的页面中，而不用像 Web Service 一样开发用户端接口。

实现这个规范，Portal 可以跟各式各样的数据源打交道，彻底终结信息孤岛的窘境。

第二节 什么是 Portal

Portal 是基于 Web 的，以“应用整合”和“消除信息孤岛”为最终目的，提供单点登录、内容聚合、个性化门户定制等功能的综合信息系统。

完整的 Portal 通常由 Portal 服务器、Portlet 容器、Portlet 构成。

1.2.1 Portal 服务器

Portal 服务器是容纳 Portlet 容器，支持 Portlet 呈现的普通或者特殊 Web 服务器。Portal 服务器通常会提供个性化设置、单点登录、内容聚合、信息发布、权限管理等功能，支持各种信息数据来源，并将这些数据信息放在网页中组合而成，提供个性化的内容定制，不同权限的浏览者能够浏览不同的信息内容。通常，Portal 提供以下功能：

单点登录：Portal 通常采用 ACL、SSL、LDAP 等业界标准的安全技术，提供对所有现有应用系统的安全集成，只需在 Portal 的唯一入口上登录一次，就可以访问所有应用系统和数据。对于安全性要求较高的应用系统，如电子商务平台、交易系统等，通过扩展接口传递用户身份信息，如数字证书信息、数字签名信息等，进行二次身份认证，保证单点登陆的安全性。

权限控制：系统采用 LDAP 对用户资源进行统一的管理，同时提供二次开发接口，可以与其他应用系统的用户管理模块对接，并能随相关业务系统实时更新访问权限。通过完善的授权机制及存取控制，用户访问权限控制到字段级别，确保用户只能访问具有权限的应用系统及相关信息。

内容管理：实现应用系统之间实时交换信息。采用多种缓存机制，保证内容交换的性能和准确性。采用基于 XML 的 Rich Site Summary (RSS) 标准，迅速在各应用系统之间传播最新变化。

信息发布：实现信息门户内容的动态维护。动态网站系统可与 OA 协同办公系统、知识管理系统等集成，网站信息须经 OA 系统的审批流程流转通过后或知识管理平台设置具有外部共享权限后才可正式发布，真正实现内外信息发布的同步。

文件管理：系统实现无缝集成多种数据源，包括：数据库、文档（Office 文档、PDF、AutoCAD、甚至 ZIP 文档）、Web 网页、FTP 站点等，并对数据按业务要求和职务特点加以分析整理，通过统一 Web 界面主动推送(Push)至用户的门户桌面，帮助用户做出及时、正确的决策。

1.2.2 Portlet 容器

Portlet 容器提供 Portlet 执行的环境，包含很多 Portlet 并管理它们的生命周期，保存 Portlet 的定制信息。

一个 Portal 容器接收到来自 Portal 的请求后，接着将这个请求传递给存在 Portal 容器的 Portlet 执行。Portlet 容器没有义务去组合 Portlet 产生的信息内容，这个工作必须由 Portal 来处理。Portal 和 Portal 容器可以放在一起视为同一个系统的组件，或者分开成为两个独立的组件。

Portlet 容器是普通 Web Servlet 容器的扩展，所以一个 Portlet 容器可以构建于一个已经存在的 Servlet 容器或者可能实现全部 Web Servlet 容器的全部功能。无论 Portlet 容器怎么实现，它的运行环境总是假定它支持 Servlet2.3 规范。

通常，Portlet 容器扩展自普通的 Servlet 容器。

第三节 什么是 Portlet

Portlet 是 Portal 中最重要的组件，负责在 Portal 中呈现信息内容，有相应的生命周期。通过自定义 Portlet，用户很容易定义个性化的 Portal 页面。Portlet 由 Portlet 容器负责管理、处理请求并返回动态页面，可以作为 Portal 的可即插即用的界面组件。

1.3.1 Portlet

一个Portlet是以Java技术为技术的Web组件，由Portlet容器所管理，专门处理客户的信息请求以及产生各种动态的信息内容。Portlet 为可插式的客户界面组件，提供呈现层成为一个信息系统。

这些由Portlet产生的内容也被称为片段，而片段是具有一些规则的标记(HTML、XHTML、WML)，而且可以和其他的片段组合而成一个复杂的文件。一个或多个 Portlet 的内容聚合而成为一个 Portal 网页。而 Portlet 的生命周期是被 Portlet 容器所管理控制的。

客户端和Portlet的互动是由Portal通过典型的请求/响应方式实现，正常来说，客户会和Portlet所产生的内容互动，举例来说，根据下一步的连接或者是确认送出的表单，结果Portal将会接收到Portlet的动作，将这个处理状况转向到目标Portlet。这些Portlet 内容的产生可能会因为不同的使用者而有不同的变化，完全是根据客户对于这个Portlet的设置。

1.3.2 Portlet 与 Servlet 的关系

Portlet 被定义成为一个新的组件，具有新的明确的界面与行为。为了尽可能与现有的 Servlet 结合达到重复使用的目的，Portlet 的规范利用了 Servlet 的规范，许多观念都很相似的，结合 Portlet、Servlet 及 Jsp 在同一个网站系统中，我们称为 Portlet 应用。在同一个 Portlet 应用中，他们将分享同一个类加载器(ClassLoader)，上下文(Context)及 Session。

、Portlet 和 Servlet 的相似之处

- @ Portlet 也是 Java 技术的 web 组件
- @ Portlet 也是有特定的 container 在管理
- @ Portlet 可以动态产生各种内容
- @ Portlet 的生命周期由 container 所管理
- @ Portlet 和客户端的互动是通过 request/response 的机制

、Portlet 和 Servlet 也有一些不同

- @ Portlet 只产生 markup 信息片段，不是完整的网页文件。而 Portal 会将所有的 Portlet markup 信息片段放到一个完整的 Portal 网页。
- @ Portlet 不会和 URL 有直接的关系
- @ 客户端必须通过 portal 系统才能和 Portlet 互动
- @ Portlet 有一些定义好的 request 处理，action request 以及 render request。
- @ Portlet 默认定义 Portlet modes 及窗口状态可以指出在网页中该 Portlet 的哪个功能正在执行及现在的状态。
- @ Portlet 可以在同一个 portal 网页之中存在多个。

、Portlet 有一些附加的功能是 Servlet 所没有的

- @ Portlet 能够存取及储存永久配置文件及定制资料。
- @ Portlet 可以存取使用者数据
- @ Portlet 具有 URL 的重写功能在文件中去动态建立连结，允许 portal server 不用去知道如何在网页的片 段之中建立连结及动作。
- @ Portlet 可以储存临时性的数据在 Portlet session 之中，拥有两个不同的范围：application-wide scope 及 Portlet private scope。

、Portlet 不具有一些功能，但是 Servlet 却有提供

- @ Servlet 具有设置输出的文字编码(character set encoding)方式
- @ Servlet 可以设置 HTTP 输出的 header
- @ Servlet 才能够接收客户对于 portal 发出的 URL 请求

1.3.3 Portlet 的生命周期

一个Portlet有着良好的生命周期管理，定义了怎样装载，实例化和初始化，怎样响应来自客户端的请求及怎样送出服务。这个Portlet生命周期由Portlet接口的init ,processAction ,render和destroy方法来表达。

载入和实例化:Portlet 容器负责载入和实例化 Portlet。当 Portlet 容器运行 Portlet 应用或者延迟到 Portlet 需要服务使用者的请求时，Portlet 就会被载入并实例化。载入 Portlet 类后，Portlet 类随即被实例化。

初始化:Portlet 类实例化后，Portlet 容器还需要初始化 Portlet。以调用 Portlet 去响应客户端的请求。Portlet 容器呼叫 Portlet 接口中的 init 方法初始化 Portlet。扩展自 PortletConfig 的类可以取出定义在部署描述文件中的初始化参数，以及 Resource Bundle。

初始化异常:在 Portlet 初始化期间，Portlet 可能会丢出 UnavailableException 或 PortletException 异常。此时，Portlet 容器不能把 Portlet 置入已启动的服务，并且 Portlet 容器必需释放这个 Portlet。destory 方法不能被呼叫，因为初始化被认为执行失败。发生失败后，Portlet 容器会尝试着重重新实例化及初始化 Portlet。这个异常处理的规则是：由一个 UnavailableException 指定一个不能执行的最小时间，当此异常发生时，Portlet 容器必需等到指定时间过去后才产生并且初始化一个新的 Portlet。

在初始化过程中所丢出的 Runtime Exception 异常，被当作 PortletException 来处理。

第四节 Liferay Portal 工作原理

Portal 系统根据需要由一个或者多个 Portal 页面组成，每个 Portal 页面包含零个或者多个的 Portlet。每个 Portlet 呈现自己的信息内容，以此实现内容聚合。通过定义每个 Portlet 的可用权限，实现个性化的桌面信息定制。

1.4.1 Portlet 样式以及窗口状态

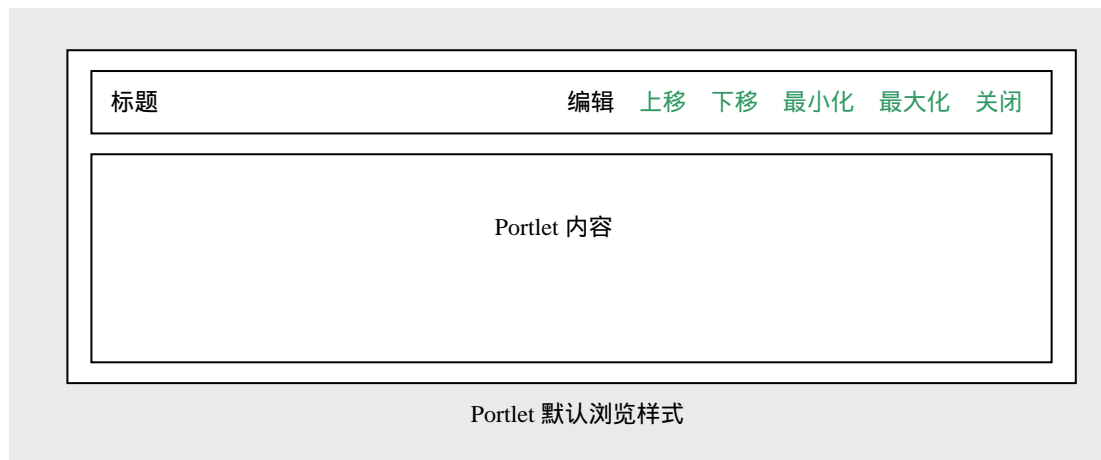


图 1.4.1-1

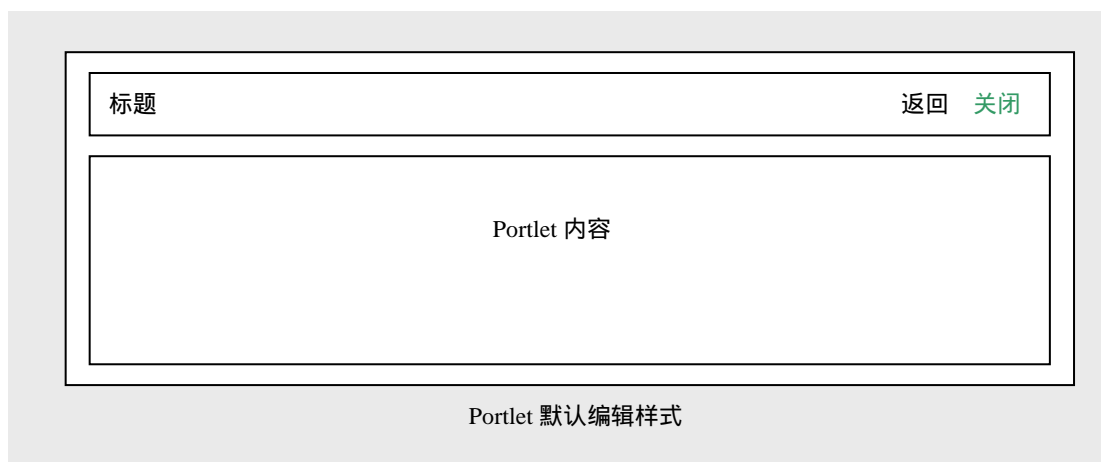


图 1.4.1-2

JCP 组织提出的 JSR168 规范定义了 Portlet 的实现标准。每个 Portlet 对外表现为一个小窗口，有自己的默认样式和窗口状态。如上图，Portlet 有自己的标题，浏览状态下支持编辑、关闭、上移、下移、最大化、最小化功能，编辑状态下支持返回和关闭功能。从各种数据来源提取的信息以 Portlet 内容的形式呈现在 Portal 中。

Portlet 样式指出 Portlet 正处于什么模式，Portlet 通常会根据所处的模式而执行不同的工作并产生不同的内容。

Portlet 模式让 Portlet 决定它该显示什么内容和执行什么动作。调用一个 Portlet 的时候，Portlet 容器会提供一个 Portlet 模式给那个 Portlet。当在处理一个请求动作时，Portlet 的模式是可以用程序来改变的。

JSR168 规范定义了三个 Portlet 模式：浏览、编辑和帮助，Liferay Portal 支持其中的全部三个模式。同时 Portal 是可以根据使用者的角色，来决定是要提供(显示)哪几个 Portlet 模式给使用者操作。

例如，匿名使用者可以操作浏览和帮助等 Portlet 模式的内容，而只有授权过的使用者可以操作编辑这个 Portlet 模式所提供的内容或动作。

在浏览这个 Portlet 模式里，所被期望要提供的功能是产生标记语言来表现此时 Portlet

的状态。举例来说，Portlet 的 浏览 模式可以包含一个或多个画面让使用者可以浏览与互动，或是一些不需要与使用者互动的静态内容。

在编辑这个 Portlet 模式里，Portlet 需要提供内容和逻辑来让使用者定制 Portlet 的行为。典型的说，编辑模式的 Portlet 会设定或更新 Portlet 的参数设定值。

在帮助这个模式里，Portlet 应该提供有关这个 Portlet 的帮助信息。这个帮助信息可以是有关这个 Portlet 的简单且条理清楚的视窗说明或是详细的说明整个来龙去脉。所有的 Portlet 并不需要都提供帮助这个模式。

一个 Portlet 可以根据窗口状态来决定在一个页面里该占多少空间。当调用一个 Portlet 时，Portlet 容器需要告诉该 Portlet 目前的窗口状态。此时 Portlet 可以根据窗口状态来决定它该对多少信息作处理。在处理请求的过程中，Portlet 可以通过程序的方式来改变窗口状态。

1.4.2 Portal 页面

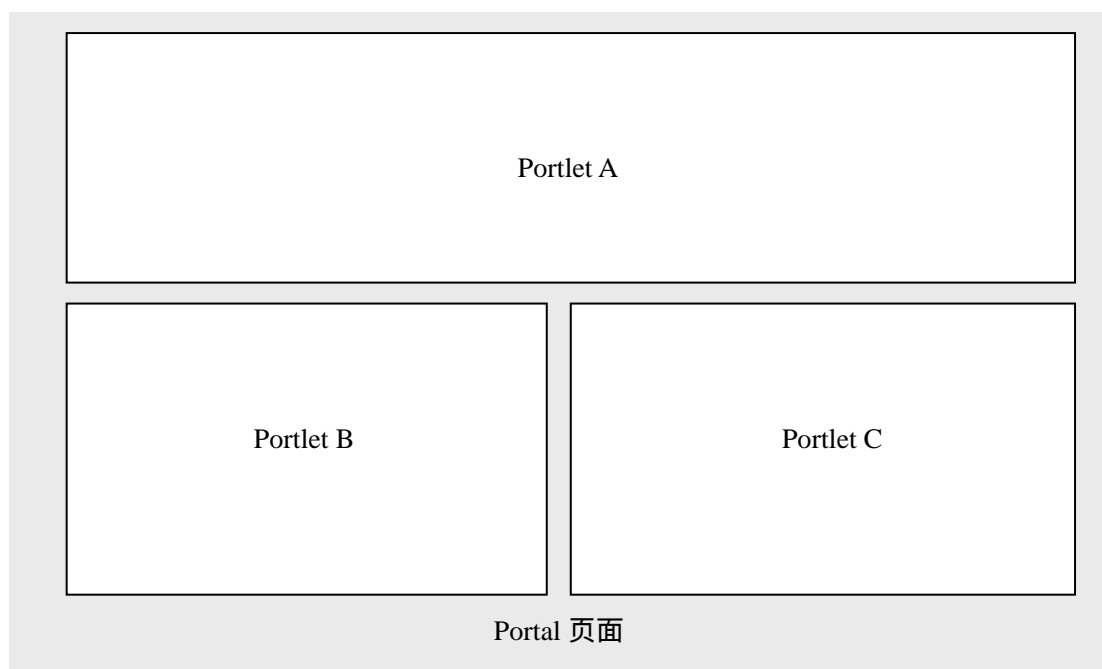


图 1.4.2-1

每个 Portal 页面包含零个或者多个 Portlet 小窗口，构成一个完整的信息呈现页面。Portal 在启动之后根据 Portlet 配置文件等信息，给 Portlet 的标题等属性赋值，赋予 Portlet 编辑、关闭等各种控制按钮，使 Portlet 成为一个标准的 Portlet 窗口。Portlet 合并这些 Portlet 窗口，组成一个完整的文档，即 Portal 页面。每个 Portlet 都处于相应的布局当中，呈现事先定义的内容，表现 Portal 公共的品质。而且 Portlet 可以在不同的布局之间切换。Portlet 响应客户端的请求，并将请求提交到相应的 URL 进行逻辑处理。

Portlet 开发完毕之后，部署到 Portal 服务器，由 Portal 服务器负责组织、权限控制和呈现。Portal 页面创建过程如下：

Portlet 在 Portlet 容器内执行，Portlet 容器接收 Portlet 产生的内容。通常 Portlet 容器将这些内容提交给 Portlet 服务器，Portlet 服务器依照这些内容建立 Portal 页面，然后将它传给客户端呈现。具体流程如下图：

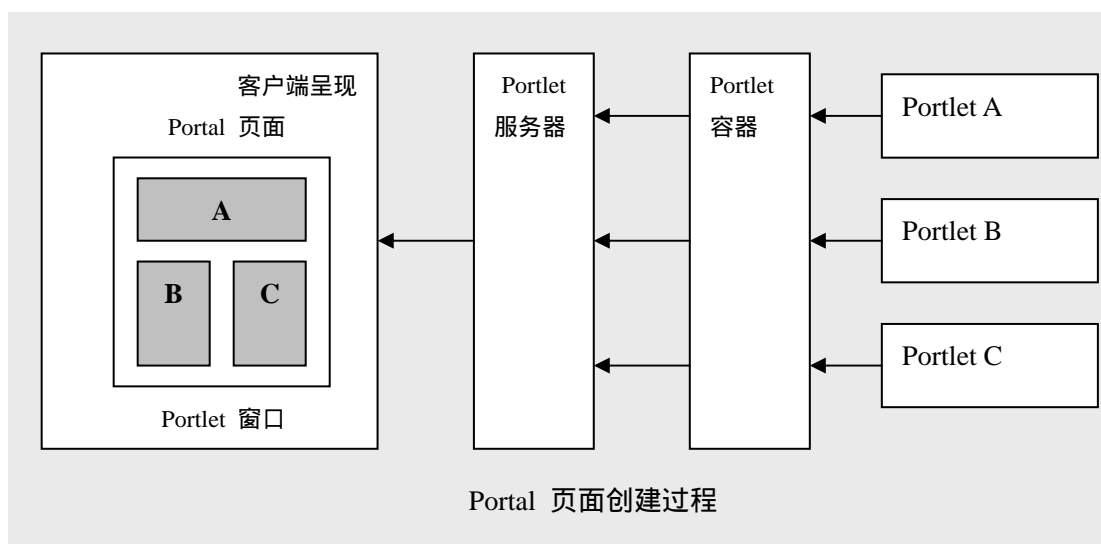


图 1.4.2-2

Portal 页面的请求过程如下:

使用者经由客户端设备 (例如浏览器) 存取 Portal , Portal 根据接收到的请求决定哪些 Portlet 需要被执行以满足需求。Portal 通过 Portlet 容器呼叫 Portlet , 然后由 Portlet 产生的片段建立 Portal 页面, 再传回客户端呈现给使用者。具体流程如下图:

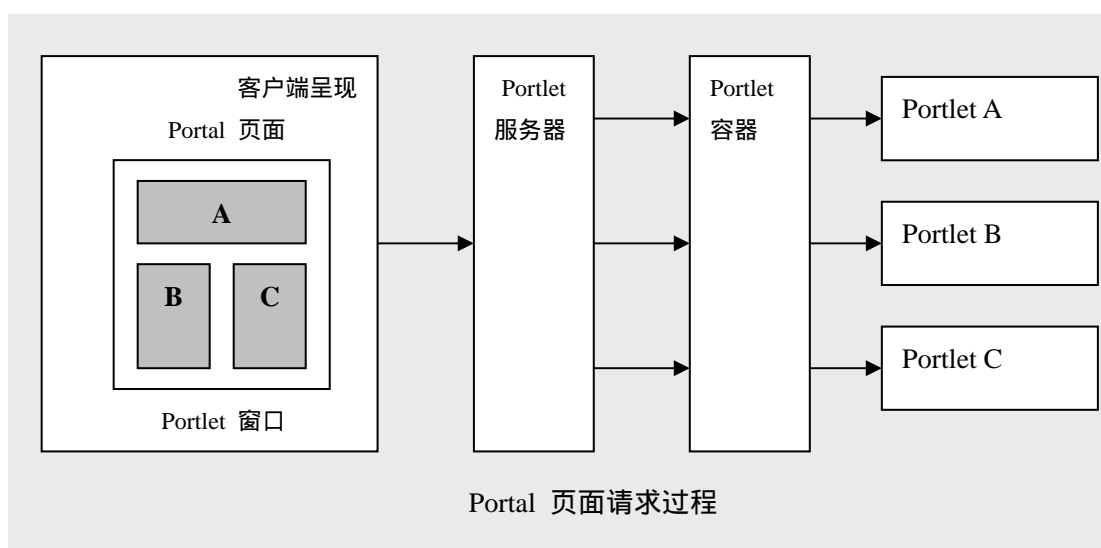


图 1.4.2-3

第二章 Liferay Portal 的使用

Liferay Portal 分为 Professional 和 Enterprise 两个版本。

Liferay Portal 支持多个应用服务器和 Servlet 容器。Liferay Portal Ent 版本需要一个健壮的 J2EE 服务器,而 Pro 版本只要一个普通的 Servlet 服务器就可以运行。如果需要运行 EJB,建议使用 Pro 版本。两个版本的源码和应用接口都是一样的。

默认的,Pro 版本分别集成 Tomcat / Jetty / Resin 作为 Web 服务器,采用 Struts 作为 Web 框架,实现轻量级的系统架构。Enterprise 集成 JBoss 作为 Web 服务器,采用 Spring 作为 Web 框架,兼顾 EJB。

Liferay Portal 默认集成 HSQL 数据库,来持久化保存用户自定义的数据。通过修改集成在 Liferay Portal 的 Tomcat 的部署描述文件,用户可以更改数据源。Liferay Portal 官方网站提供了数据库表的生成脚本。

下面以 Pro 版本(Tomcat 服务器)为例,讲述 Liferay Portal 的用户策略、内容布局、桌面和品质。

第一节 Liferay Portal 安装

由于 Liferay Portal Pro 版本集成了 Tomcat 服务器 V5,所以只要把应用包下载解压就可以直接运行。

1、从 http://www.liferay.com/web/guest/downloads/portal_pro 下载Pro版本zip包,解压到目录{PORTAL_HOME},目录结构相对普通的Tomcat增加了Liferay文件夹。Liferay是默认的Web应用。

2、正确安装 JDK1.4 或者 JDK1.5,并在环境变量里面正确配置 JAVA_HOME 变量。

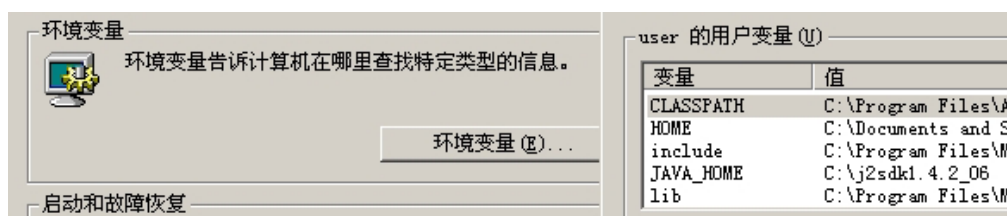


图 2.1-1

3、从命令行启动{PORTAL_HOME}/bin/startup.bat,启动 Liferay Portal。

4、在浏览器地址栏输入<http://localhost>,访问Portal首页。

5、用 Login 为 test@liferay.com 密码为 test 的用户登录 Portal 系统,得到的是一个 Demo 的首页。



图 2.1-2

如果启动呈现异常，请查看 Tomcat 控制台查找原因。

Liferay Portal 启动之后，HSQL 数据库自动启动。

登录系统后，点击右上角“ My Account ”链接，在“ Display ”选项卡中将 Language 改为“ Chinese(China)”，以便中文化 Portal 界面。

第二节 Liferay Portal 的用户策略

Liferay Portal 通过定义严谨的用户策略、灵活的可个性化定制的内容和布局以及丰富可定制的品质策略，实现灵活的可定制的产品理念。

Liferay Portal 采用用户 - 用户组 - 角色 - Portlet 的关联方式来实现用户权限的管理。用户隶属于用户组（也可以单独存在），该用户组具有某种（多种）角色，角色分配给用户组，也可以直接分配给用户。而操作某个 Portlet 需要具有其指定的角色。下面通过实例操作，来了解和体验一下 Liferay Portal 的用户管理策略。

2.2.1 定义用户

Liferay Portal 的用户管理在系统管理的 Portlet 中。缺省只有系统管理员才能使用。登录 Portal 后，可以在默认的桌面上找到“系统管理”Portlet。如果没有，从页面底部的选择框中选择“系统管理”添加上。也可以通过右上角“CMS”桌面的“内容和布局”页面找到管理入口。

从“系统管理”Portlet 中选择“用户”项，进入用户管理界面。

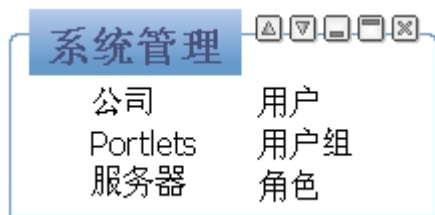


图 2.2.1-1



图 2.2.1-2

2.2.2 添加用户

图 2.2.1-2 所示页面右边为“新增用户”列，填入你所要增加的用户名称，姓氏，用户标识（可自动生成），邮件地址，密码（可自动生成）等。可以修改该用户所具有的用户组和角色信息（也可创建之后再修改）。用户标识必须是系统唯一的，所以请确保你所输入的用户标识与已有的不冲突。

点击“新增用户”，我们成功增加一位用户标示为“educhina”的用户，如图 2.2.1-2 所示。左侧列表中新增一项“educhina eamoi”。然后我们就可修改这位用户的用户组，角色，个人档案等信息了。

2.2.3 修改用户

选择用户列表中一项，然后点击底部的三个编辑按钮，就可以分别编辑该用户的用户组、角色、档案等信息了。

此处我们选择用户“educhina eamoi”，然后选择“编辑档案”，出现档案编辑页面。如图 2.2.3-3 所示。填写你想要修改的信息，点击对应的“更新”按钮即可完成修改。需要注意的是整个档案页面分成几个部分，需要分别修改更新。

选择用户“educhina eamoi”，然后选择“编辑角色”，进入角色编辑页面，如图 2.2.3-4 所示。左侧列表框为当前该用户所具有的角色，右侧列表为所有可用的角色。要赋给用户新角色，则从右侧选择一项或多项，通过中间的转移按钮，从右侧添加至左侧。要删减用户角色，则从左侧移至右侧。最后点击底部的“更新”即可。更新完页面会自动返回。

编辑用户组的操作同上。

新增用户

名字

educhina

中间名

姓氏

eamoi

用户标识

educhina

自动生成用户标识 ☐

邮件地址

eamoi@163.com

密码

自动生成密码 ☐

需要重置密码 ☐

生日

一月

1

1984

性别

男

图 2.2.3-1

您成功地增加一名用户。

编辑用户

输入名称或从列表中选择:

John Wayne

Test Mail

educhina eamoi

图 2.2.3-2

系统管理

公司 - Portlets - 服务器 | 用户 - 用户组 - 角色

主要档案

显示

名字: educhina

中间名:

姓氏: eamoi

昵称:

用户标识: educhina

邮件地址: eamoi@163.com

生日: 一月 1 1904

语言: 中文 (中国)

时区: (GMT) GMT

问候语: Welcome, educhina eamoi!

分辨率: 800 by 600 pixels

更新

更改

上次登录: Never

失败的登录尝试: 0

活跃: 是

更新

短消息信使标识

具有文本短消息功能的手机可以将手机号码映射到邮件地址。例如, 一个中国移动的手机号码139 0123 4567可以有一个邮件地址 sms13901234567@monetnet.com。

SMS:

更新

即时信使标识

AIM:

图 2.2.3-3

系统管理

公司 - Portlets - 服务器 | 用户 - 用户组 - 角色

编辑用户的角色: educhina eamoi

当前

Power User
User

可用

Administrator
Bookmarks Admin
Calendar Admin
Document Library Admin
Guest
Journal Admin
Journal Designer
Journal Editor
Journal Writer
Message Boards Admin

更新

图 2.2.3-4

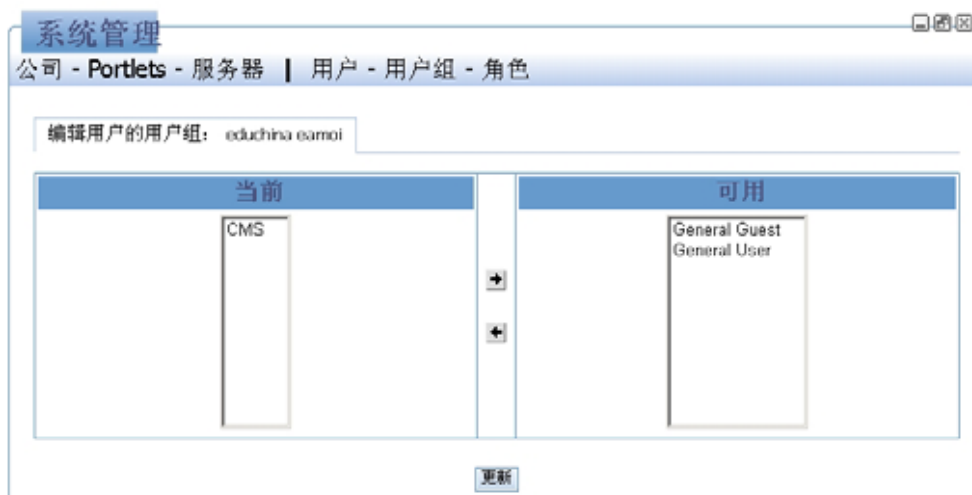


图 2.2.3-5

2.2.4 定义用户组

用户组是对具有相同角色的用户的聚合。只要把用户需要的角色赋给用户组，则该用户组内所有的用户就都具备了该角色，具有该角色所有的权限，这样子就简化了用户权限管理。

用户组还有一个共用首页面的概念，是该用户组所有用户共享的页面。这些用户可以看到一个内容布局一致的页面，用以在用户组中共享信息。

从系统管理中选择[用户组]项，进入用户组定义页面。如图 2.2.4-1 所示：



图 2.2.4-1

2.2.5 新增、重命名用户组

图中左侧为用户组列表，右侧为新增，和重命名。

在新增部分直接添入用户组名称，点击“新增用户组”即可。

选择列表中一个用户组，然后在右侧下方添入要修改的新组名，点击“更新”既可。

“友好的 URL”为该用户组的共用首页面设置 URL。

2.2.6 修改用户组

对于用户组我们可以修改它的角色，所包含的用户和该用户组的共用首页面。

选择用户组列表中一项，然后点击底部的“编辑角色”，进入用户组角色编辑页面，如图 2.2.6-1 所示。页面左侧为用户组当前已具备的角色，右侧为所有可用角色。添加删除操作同[2.2.3 修改用户]。更新完会自动返回用户组列表页面。

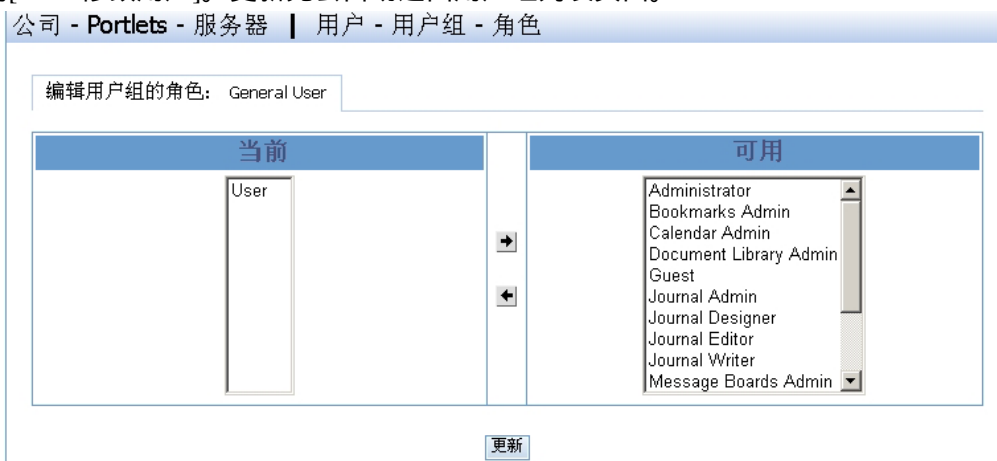


图 2.2.6-1

选择用户组列表中一项，然后点击底部的“编辑用户”，进入用户组用户编辑页面，如图 2.2.6-2 所示。页面左侧为用户组当前包含的用户，右侧为所有用户。添加删除操作同[2.2.3 修改用户]。更新完会自动返回用户组列表页面。

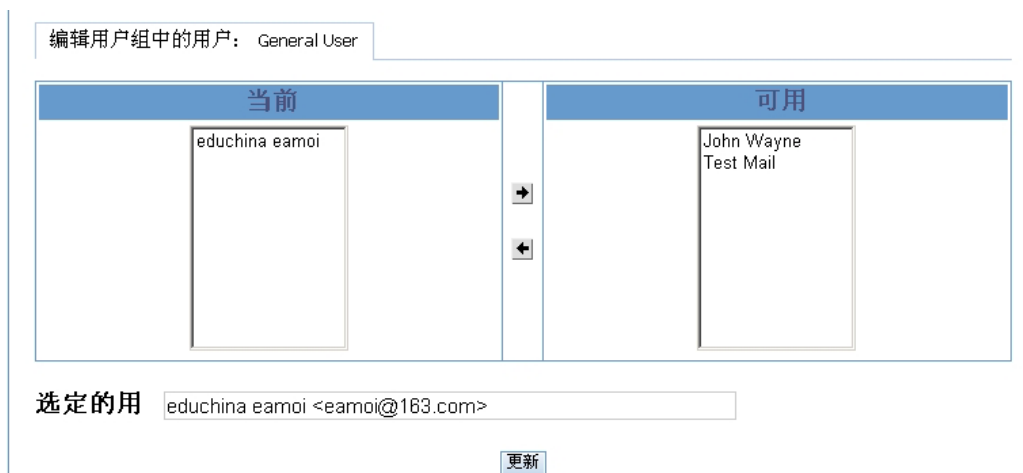


图 2.2.6-2

选择用户组列表中一项，然后点击底部的“编辑页面”，进入用户组首页编辑页面，如图 2.2.6-3 所示。

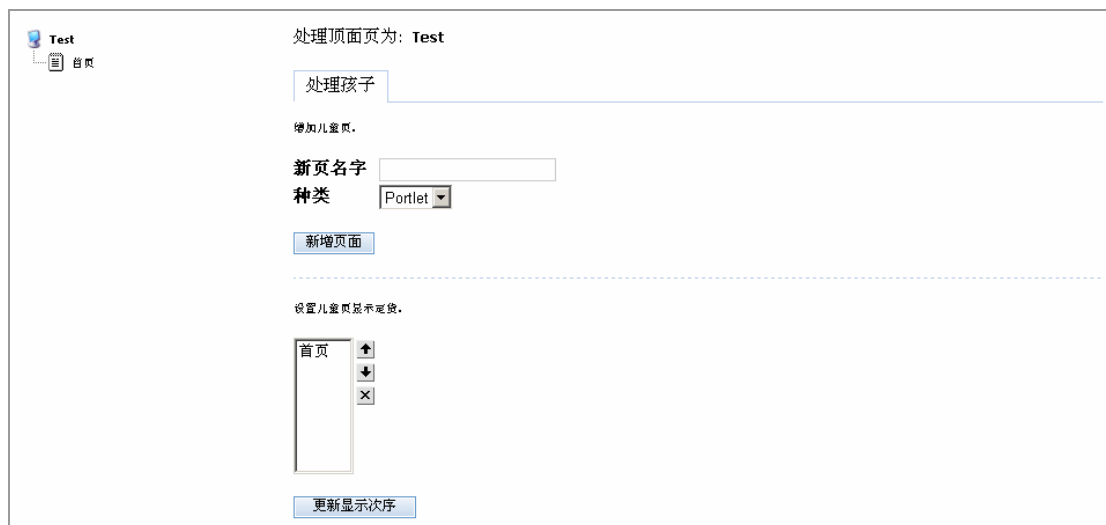


图 2.2.6-3

首先增加一个新页。在“处理子页”一栏，填入新页的名字，选择类型，点击“新增页面”，左侧树状列表中会增添一个以新页名字为标题的新项。

然后为新页设置布局。点击左侧列表中的新页，右侧出现布局编辑页面，如图 2.2.6-4 所示。详细设置布局的操作可参考所述。



图 2.2.6-4

2.2.7 定义角色

角色是对用户身份的一种定义。不同的角色具有不同的权限。被赋予这种角色的用户自然就获得了该角色的权限。

从系统管理中选择[角色]项，进入角色定义页面。如图 2.2.7-1 所示。

图 2.2.7-1

2.2.8 新增、重命名角色

图中左侧为角色列表，右侧为新增，和重命名。

在新增部分输入角色名称，点击“新增角色”即可新增一个角色。

选择列表中一个角色，然后在右侧下方对应栏位填入新角色名，点击“重命名角色”即可重命名该角色。

2.2.9 修改用户组角色

对于角色，我们可以修改它的用户组和用户。该操作可以通过修改用户组和用户的角色来完成。

选择角色列表中一项，然后点击底部的“编辑用户组”，进入角色的用户组编辑页面，如图 2.2.9-1 所示。页面左侧为已具备当前角色的用户组，右侧为所有用户组。添加删除操作同[2.2.3 修改用户]。更新后自动返回角色列表页面。

图 2.2.9-1

选择角色列表中一项，然后点击底部的“编辑用户”，进入角色的用户编辑页面，如图 2.2.9-2 所示。页面左侧为已具备当前角色的用户，右侧为所有用户。添加删除操作同[2.2.3 修改用户]。更新后自动返回角色列表页面。



图 2.2.9-2

2.2.10 定义 Portlet 的角色

通过为 Portlet 设置必需的角色，我们实现了用户与 Portlet 的关联。只有当用户或所属的用户组具有 Portlet 所必需的角色，他才能操作该 Portlet。

从系统管理中选择[Portlet]项，进入 Portlet 定制页面。如图 2.2.10-1 所示。页面中显示了目前系统中可用的 Portlet 列表，列表中显示了 Portlet 目前的状态和必需的角色。

系统管理				
公司 - Portlets - 服务器 用户 - 用户组 - 角色				
Portlets				
Portlet名称	编辑	状态	被索引	必需的角色
CVS	编辑	活跃	否	Power User, User
Google	编辑	活跃	否	Power User, User
Hello Velocity	编辑	活跃	否	Power User, User
Hello World	编辑	活跃	否	Power User, User
IFrame	编辑	活跃	否	Guest, Power User, User
Portlet Aggregator	编辑	活跃	否	Guest, Power User, User
Portlet Aggregator	编辑	活跃	否	Guest, Power User, User
RSS	编辑	活跃	否	Power User, User
Reports	编辑	活跃	否	Power User, User
Reverend Fun	编辑	活跃	否	Power User, User
Stand to Reason	编辑	活跃	否	Power User, User
WSRP 代理人	编辑	活跃	否	
Wiki	编辑	活跃	是	Power User, User
Wiki 显示	编辑	活跃	否	Guest, Power User, User

图 2.2.10-1

选择一个 Portlet，点击“编辑”进入 Portlet 定制页面，如图 2.2.10-2 所示。页面左侧为 Portlet 必需的角色，右侧为所有角色。添加删除操作同[2.2.3 修改用户]。

系统管理	
公司 - Portlets - 服务器 用户 - 用户组 - 角色	
书签	
宽度 <input type="button" value="宽"/>	状态 <input type="button" value="活跃"/>
角色	
<div>当前</div> <div>Power User User</div>	<div>可用</div> <div> Administrator Bookmarks Admin Calendar Admin Document Library Admin Guest Journal Admin Journal Designer Journal Editor Journal Writer Message Boards Admin </div>
<input type="button" value="更新"/> <input type="button" value="取消"/>	

图 2.2.10 - 2

第三节 Liferay Portal 内容和布局

Portlet 容器采用布局来对包含的 Portlet 进行管理并呈现,不同的布局决定了不同的 Portlet 呈现效果。每个加入到 Portal 服务器的 Portlet 必须属于某个布局,才能够被使用者所看到。内容则是 Portlet 对外呈现的信息片断,是 Portlet 的核心。两者都是 Portal 的重要组成部分。Liferay Portal 采用开源框架 Struts 的 Tiles 来管理内容和布局。

2.3.1 什么是布局

布局,即 Layout,也可以称为布局管理器,是 Portlet 容器管理 Portlet 的一个重要工具。一个布局,在生成的 Portal 页面中,呈现出单行多列或者多行多列的效果。而 Portlet 就内嵌在某一列中。

在 Liferay Portal 中,将列分为宽栏和窄栏。通常,宽栏占据页面 2/3 的宽度,窄栏占据页面 1/3 的宽度。每个 Portlet 在部署的时候都必须在部署描述符文件中指定 Portlet 是被部署在宽栏或者窄栏当中,默认是部署在宽栏中。



图 2.3.1-1

Liferay Portal 采用 tpl 文件来定义布局,这些 tpl 文件存储在 {PORTAL_HOME} /liferay/html/layouttpl 文件夹中。在 tpl 文件中,规定每个列的宽度。当 Portlet 加入到列中时,取得当前列的宽度,然后根据这个宽度确定 Portlet 窗口的显示宽度。tpl 文件采用标准的 HTML 代码和 Liferay Portal 自定义的标签来定义布局。如下图:

```
<table border="0" cellpadding="0" cellspacing="0" width="100%">
<tr>
  <td colspan="3" valign="top">
    <runtime-Portlet-column id="column_1" />
  </td>
</tr>
<tr>
  <td colspan="3" valign="top">
    <runtime-Portlet-column id="column_2" />
  </td>
</tr>
</table>
```

图 2.3.1-2

只要把定义的 tpl 文件路径加入到部署描述文件中,Liferay Portal 在启动的时候可以自动载入,供系统调用。如下图:


```

<layout-templates>
  <layout-template id="1_column" name="1 Column">
    <template-path>/html/layouttpl/1_column.tpl</template-path>
  </layout-template>
  <layout-template id="2_columns_i" name="2 Columns (50/50)">
    <template-path>/html/layouttpl/2_columns_i.tpl</template-path>
  </layout-template>
  <layout-template id="2_columns_ii" name="2 Columns (30/70)">
    <template-path>/html/layouttpl/2_columns_ii.tpl</template-path>
  </layout-template>
  <layout-template id="1_2_1_columns" name="1-2-1 Columns">
    <template-path>/html/layouttpl/1_2_1_columns.tpl</template-path>
  </layout-template>
</layout-templates>

```

图 2.3.1-3

Liferay Portal 默认的布局允许有一列、二列、三列的布局。二次开发的时候可以定义自己的布局文件。

在每个列的底部,有一个下拉列表框,列出本列可用的所有 Portlet。列表框旁边的“添加”按钮,则可以将选中的按钮添加到列中显示。

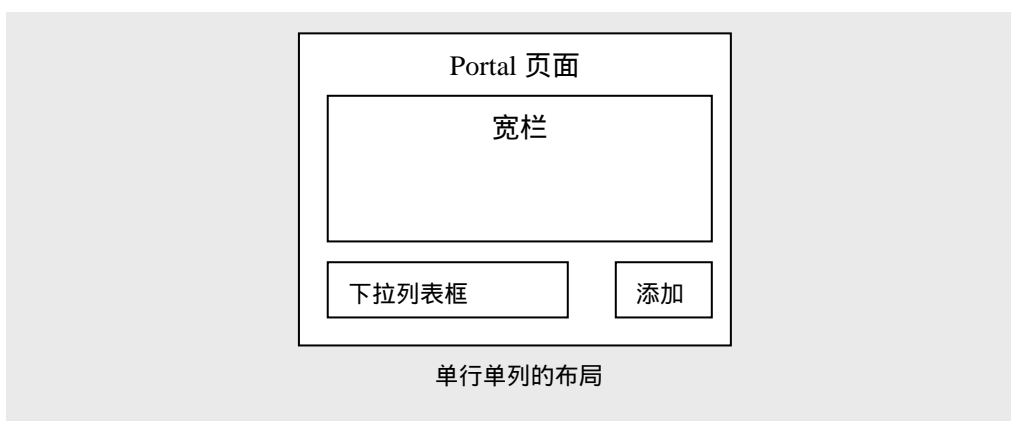


图 2.3.1-4

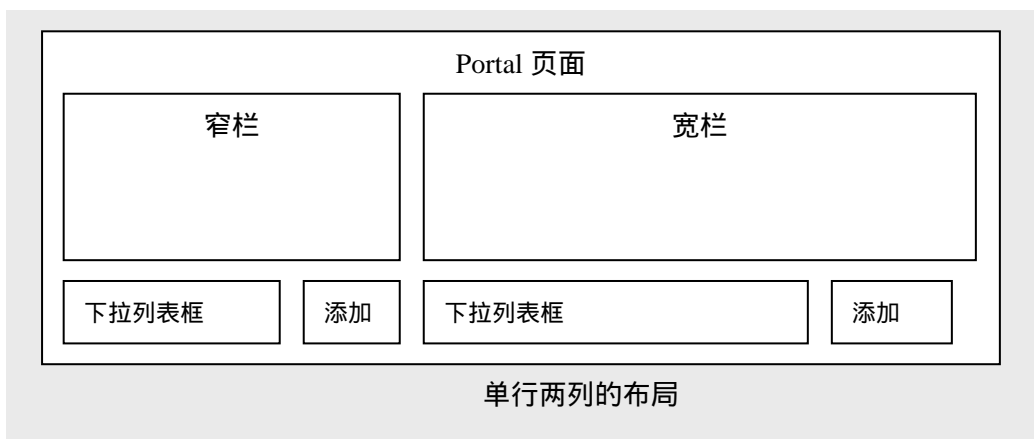


图 2.3.1-5

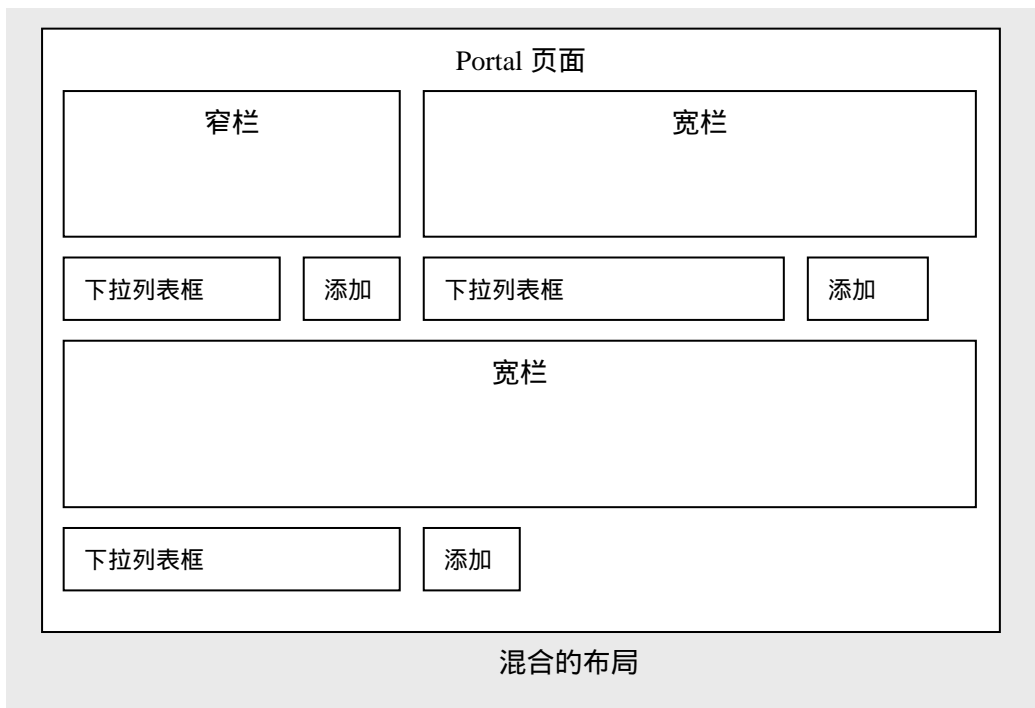


图 2.3.1-6

2.3.2 什么是内容

内容具体指 Portlet 显示出来的标记片断，称为 Portlet 内容。通常，当 Portlet 窗口处于浏览或者编辑状态的时候，就会表现相应的 Portlet 内容。内容在开发 Portlet 的时候确定。

Portlet 对各种来源的数据进行加工和逻辑处理，最后输出为一些规则的标记(HTML、XHTML、WML)，最后在 Portlet 容器中形成 Portlet 窗口，供 Portal 组合成为 Portal 页面。

内容是 Portlet 的信息主体，它形成的表单、链接等同时接受使用者的信息请求或者数据提交，并将系统对使用者请求的响应呈现在客户端。下图为以日历为内容的 Portlet。

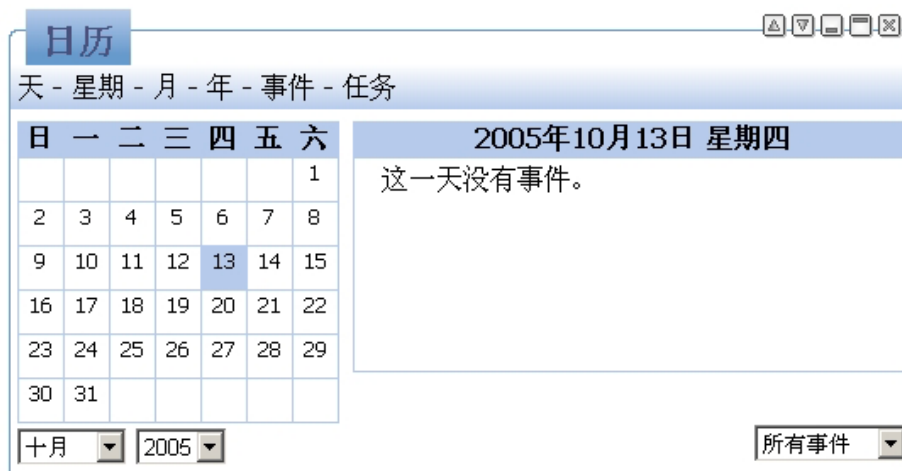


图 2.3.2-1

2.3.3 内容布局与 Portlet 的关系

通过定义布局，对 Portlet 进行有效管理，是 Liferay Portal 容器组织 Portlet 的有效方式。在相同的列中，Portlet 可以很容易的调整位置。当列中的 Portlet 数量超过一个的时候，通过 Portlet 右上角的“上移”和“下移”按钮，可以调整相邻 Portlet 的上下位置。当 Portlet 的内容较长的时候，可以把 Portlet 部署到宽栏中，占据更大的屏幕空间，以有效的显示数据。相应的，如果 Portlet 内容较少时，可以把 Portlet 部署到窄栏中。



图 2.3.3-1

每个 Portlet 在定义的时候，可以在部署描述文件中定义 Portlet 所属的类 (Category)，每个类可用的布局，这些定义也可以启动 Portal 之后在“内容与布局”选项卡中修改。

在“修改布局”子选项卡中可以修改的包括桌面的标识，如果是单行两列的布局，还可以调整宽栏和窄栏的位置。如下图：利用 Liferay Portal 提供的工具，可以很方便的修改布局内容和它被显示在 Portal 页面的什么地方。



图 2.3.2-2



图 2.3.3-3

在“处理孩子”子选项卡中，可以定义每个 Portal 页面的子页面，形成页面树。根节点子页面会平行地出现在桌面上。如下图：

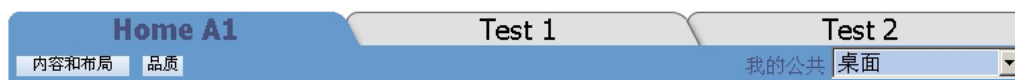


图 2.3.3-4

Portal 是大量信息和系统的集成。Portlet 内容往往来源与集成的各个系统。Portlet 面向的用户通常也是复杂的。除了用户策略中合理定义 Portlet 的用户策略外，也可以对 Portlet 内容进行过滤，针对相应的用户显示适当的信息。当然，这种方法没有定义用户策略那样来得直观。

2.3.4 选择内容和布局

Liferay Portal 内置了数个 Portlet 应用，包括系统管理、日历、书签等等。目前，Liferay Portal 支持单行单列、单行两列、单行三列的布局显示，可以在相应桌面的“内容和布局”中选择。

每个默认的 Portlet 则来自于各个数据源的既有数据，或者对该数据的重新加工处理。通过定义 Portlet 所属类别和相应的用户策略，成功实现 Portlet 的合理显示。

Liferay Portal 提供了基于 Web 的工具，可以很方便的在几种默认的布局之间切换。

- 、登录系统后，选择桌面当中的“内容和布局”，进入布局管理页面。
- 、选中桌面的第一级节点，然后在“列数”中选择需要的列数。
- 、点击底部的“更新页”按钮，提交选择。布局修改生效。返回桌面。

可以看到，单行单列的布局默认是一个宽栏；单行两列的布局默认是一个宽栏和一个窄栏；单行三列的布局默认是三个窄栏。

第四节 Liferay Portal 的桌面

2.4.1 什么是桌面

定义个性化的桌面是 Portal 的标准功能之一。用户可以把任何允许的 Portlet 添加到桌面上，构建符合自己需求的信息集合。

桌面是用户定义的 Portlet 的集合，也是 Portlet 内容的最终呈现媒介之一，可以是一个 Portal 页面，或者是一个 Portal 页面集合，里面包含一个或者多个的 Portlet。每个桌面通常用一个或者多个布局来管理桌面上的 Portlet。

Portlet 在部署之前，会在部署描述文件中定义该 Portlet 可用的用户组和角色。在定义了用户所属的用户组和角色之后，就可以在桌面下方的添加列表中看到该用户可用的所有 Portlet。用户可以把任何符合该用户角色权限的 Portlet 添加到相应的布局中。这些 Portlet 和桌面的定制信息会被 Portal 服务器持久化保存。



图 2.4.1-1

Portal 启动之后 根据定制的 Portlet 和桌面信息 搜索并实例化 Portlet 构建 Portal 页面，把 Portlet 内容显示在用户定制的桌面上。

2.4.2 定义个性化的桌面

在完成用户策略、Portlet 定义之后，登录 Liferay Portal，就可以进行个性化桌面的定制了。用户登录进入到相应的桌面后，在相应的布局列底部可以看到可用的全部 Portlet 列表。选中某个 Portlet，点击“添加”按钮，将选中的 Portlet 添加到列中。对已经添加到列中的全部 Portlet，可以通过点击 Portlet 窗口右上角的“上移”、“下移”按钮，调整 Portlet 窗口的位置。也可以点击 Portlet 窗口右上角的“最大化”、“最小化”按钮，改变窗口的状态。定制完毕的桌面效果如下图：

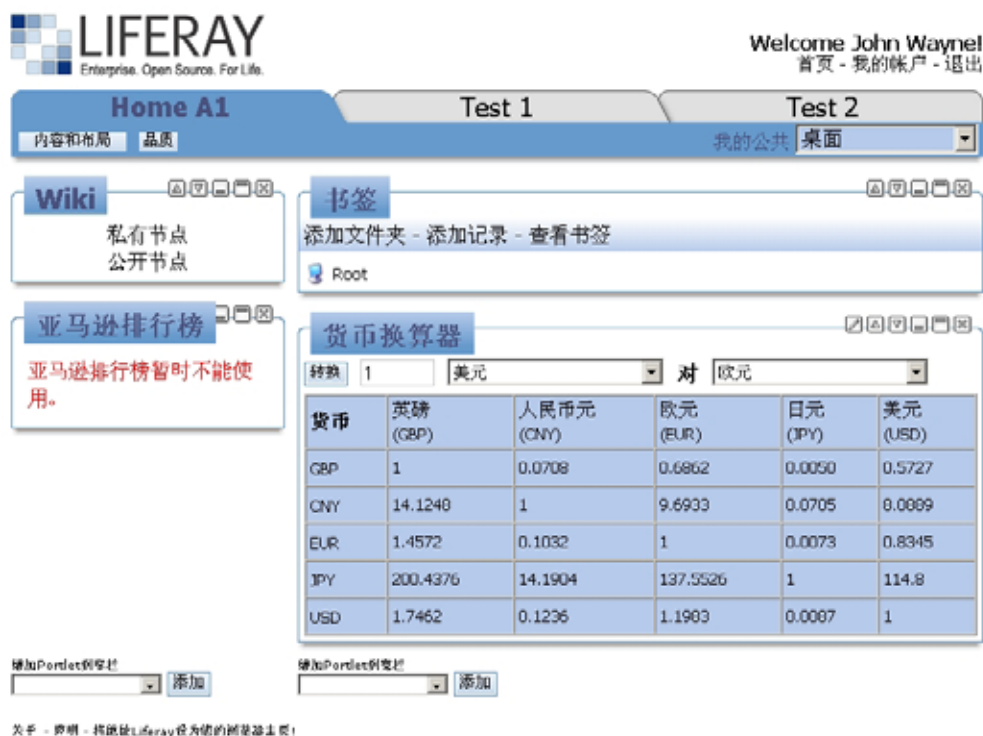


图 2.4.2-1

第五节 Liferay Portal 的品质

Liferay Portal 支持个性化的皮肤和外观设计，并将此作为品质单独管理。

2.5.1 什么是品质

品质是 Liferay Portal 的外观，包括题材和色彩设计两个部分。题材主要影响 Portlet 窗口的样式和 Portal 的整体效果，包括 Portlet 边框格式、功能按钮、Portal 页面效果等等。色彩设计主要影响 Portal 的 CSS 样式效果。

Liferay Portal 默认定义了多种题材效果和色彩设计效果。使用者可以在“品质”选项卡中很容易的选择自己满意的品质。

2.5.2 品质和 Portlet、Portal 的关系

品质跟 Portlet 和 Portal 的呈现效果有很大的关系。通常应该根据 Portlet 内容选择适当的品质即题材和色彩设计。

题材对 Portal 的影响主要体现在背景和整体风格上面，以及 Portlet 和其他功能菜单的布局位置。色彩设计主要影响 Portal 的字体大小以及颜色等效果。

题材主要控制 Portlet 生成的窗口的样式效果，包括边框效果、标题样式等等。色彩主要控制 Portlet 窗口的字体效果，包括字体大小、字体颜色等等。

选择合适的题材和色彩设计对于 Portal 页面的整体呈现效果有明显的影响。如下图：



图 2.5.2-1 采用不同的色彩设计得到的 Portal 页面



图 2.5.2-2 采用不同的题材效果得到的 Portal 页面

2.5.3 定义个性化的品质

用户登录 Portal 系统之后，点击功能菜单上的“品质”，进入品质定制页面。选择适当的题材和色彩设计，相应的品质效果立即生效。

返回桌面查看品质效果。

使用者可以在二次开发的时候定义自己的品质，只要按照规范，在部署描述文件中定义可用的品质，Liferay Portal 就可以自动调用。如下图：

```
<look-and-feel>
  <compatibility>
    <version>3.6.0</version>
  </compatibility>
  <theme id="brochure" name="Brochure">
    <root-path>/html/themes/brochure</root-path>
    <templates-path>/html/themes/brochure/templates</templates-path>
    <images-path>/html/themes/brochure/images</images-path>
    <template-extension>jsp</template-extension>
    <color-scheme id="01" name="Default">
      <![CDATA[
        body-bg=#FFFFFF

        layout-bg=#FFFFFF
        layout-text=#000000

        layout-tab-bg=#E0E0E0
        layout-tab-text=#000000

        layout-tab-selected-bg=#6699CC
        layout-tab-selected-text=#4A517D

        portlet-title-bg=#6699CC
        portlet-title-text=#4A517D

        portlet-menu-bg=#B6CBEB
        portlet-menu-text=#000000

        portlet-bg=#FFFFFF

        portlet-font=#000000
        portlet-font-dim=#C4C4C4

        portlet-msg-status=#000000
        portlet-msg-info=#000000
      ]]>
    </color-scheme>
  </theme>
</look-and-feel>
```

图 2.5.3-1

第六节 Liferay Portal 的部署描述文件

跟所有的 Web 应用一样，Liferay Portal 采用多个 XML 部署描述文件，来初始化部署信息，规范操作模式，比如 Portlet 的初始化信息、可用的 Portlet 列表、Portlet 所属角色和用户组等等。通过这些部署描述文件，Liferay Portal 可以在启动的时候自动加载 Portlet，根据需要生成所需的 Portlet 页面。普通的 Web 应用，也可以很方便的转换成可部署的 Portlet。这种实现也是 JSR168 所规定的。

2.6.1 web.xml

web.xml 是所有 Java Web 应用的部署描述文件。其正式的规范由 http://java.sun.com/dtd/web-app_2_3.dtd 定义。

与其他普通 Web 应用相比，Liferay Portal 的 Portlet 应用还需要在 web.xml 中增加如下内容：

a、监听器：

```
<listener>
  <listener-class>com.liferay.portal.servlet.PortletContextListener</listener-class>
</listener>
```

这个要求 web 服务器监听所有跟 Portlet 有关的请求信息，并将监听到的内容交给 Liferay Portal 的 Portlet 容器处理。

b、Portlet Servlet 映射：


```

<servlet>
  <servlet-name>yourPortlet</servlet-name>
  <servlet-class>com.liferay.portal.servlet.PortletServlet</servlet-class>
  <init-param>
    <param-name>portlet-class</param-name>
    <param-value>full.name.of.yourPortlet</param-value>
  </init-param>
  <load-on-startup>0</load-on-startup>
</servlet>

.....
<servlet-mapping>
  <servlet-name>yourPortlet</servlet-name>
  <url-pattern>/yourPortlet/*</url-pattern>
</servlet-mapping>

```

其中，servlet-name 为部署的 servlet 名称；init-param 中定义自己的 Portlet 类，这个 param-name 要跟 portlet.xml、liferay-portlet.xml、liferay-display.xml 中的 portlet-name 节点值一致。

c、标签库映射：

```

<taglib>
  <taglib-uri>http://java.sun.com/Portlet</taglib-uri>
  <taglib-location>/WEB-INF/tld/liferay-portlet.tld</taglib-location>
</taglib>

```

定义了这个标签库映射，在 JSP 文件中才可以使用诸如<portlet:defineObjects />在内的一些特定的 Portlet 标签。

如果在应用中用到其他的元素，可以按照 web.xml 规范加入到相应的位置当中。

Liferay Portal 默认的 liferay 应用，由于使用了 Struts、Hibernate、Spring 在内的多个开源框架，所以{PORTAL_HOME}/liferay/WEB-INF/web.xml 文件会相对复杂些。

在自定义的 Portlet，可以使用 getPortletConfig().getInitParameter(“”)和 getPortletConfig().getParameterNames(“”)两个方法来取得在 web.xml 中定义的参数。

2.6.2 portlet.xml

portlet.xml 用来定义 Portlet 的诸如部署名称、初始化参数、支持模式、resource bundle 等普通的初始化信息，包括 portlet-name、display-name、portlet-class、init-param、expiration-cache、supports、portlet-info、security-role-ref 等等。其正式的规范请参考：http://java.sun.com/xml/ns/Portlet/Portlet-app_1_0.xsd。根目录为 portlet-webapp。

portlet-name：Portlet 的规范名称，在 Portlet 应用中必须唯一，主要用在 Portlet 部署和映射中。

display-name：供部署工具调用的 Portlet 简称，在 Portlet 应用中必须唯一。

portlet-class：Portlet 对应的类，这个类必须直接或者间接的继承

javax.Portlet.GenericPortlet。

init-param：初始化参数，有成对的<name>和<value>子元素。通常定义 Portlet 相应模式下可用的 JSP 页面。

expiration-cache：定义 Portlet 加载允许最长的过期时间，以秒为单位。-1 代表用不过期。

supports：定义 Portlet 支持的模式。所有的 Portlet 都必须支持浏览模式。

其他的元素含义请参照：http://java.sun.com/xml/ns/Portlet/Portlet-app_1_0.xsd

当 Web 应用中有多多个的 Portlet 时，可以统一的在 Portlet.xml 中定义一组的<portlet>元素。

```
<portlet>
  <portlet-name>TestPortlet</Portlet-name>
  <display-name>TestPortlet</display-name>
  <portlet-class>com.educhina.portal.FirstPortlet</Portlet-class>
  <init-param>
    <name>view-jsp</name>
    <value>/view.jsp</value>
  </init-param>
  <init-param>
    <name>edit-jsp</name>
    <value>/edit.jsp</value>
  </init-param>
  <expiration-cache>0</expiration-cache>
  <supports>
    <mime-type>text/html</mime-type>
  </supports>
  <supports>
    <mime-type>text/html</mime-type>
    <Portlet-mode>edit</Portlet-mode>
  </supports>
  <portlet-info>
    <title>educhina Test Portlet</title>
    <short-title> educhina Test Portlet </short-title>
    <keywords> educhina Test Portlet </keywords>
  </portlet-info>
  <security-role-ref>
    <role-name>guest</role-name>
  </security-role-ref>
</portlet>
```

2.6.3 liferay-Portlet.xml

定义 Portlet 默认可用的用户组、默认模板、是否支持多个实例等，规范由http://www.liferay.com/dtd/liferay-Portlet-app_3_5_0.dtd 定义。

liferay-portlet.xml 主要包含单独或者成组的<portlet>、<role-mapper>。其中，<portlet>下包含<portlet-name>、<struts-path>、<use-default-template>、<instanceable>等子元素，<portlet-name>在应用中必须唯一，且要跟 portlet.xml 相同；<role-mapper>下包含成对的<role-name>、<role-link>子元素。具体的元素含义请查看上述 dtd 定义。

```
<liferay-portlet-app>
  <portlet>
    <portlet-name> TestPortlet </portlet-name>
    <struts-path> TestPortlet </struts-path>
    <use-default-template>true</use-default-template>
    <instanceable>true</instanceable>
  </portlet>
  <role-mapper>
    <role-name>administrator</role-name>
    <role-link>Administrator</role-link>
  </role-mapper>
  <role-mapper>
    <role-name>guest</role-name>
    <role-link>Guest</role-link>
  </role-mapper>
  <role-mapper>
    <role-name>power-user</role-name>
    <role-link>Power User</role-link>
  </role-mapper>
  <role-mapper>
    <role-name>user</role-name>
    <role-link>User</role-link>
  </role-mapper>
</liferay-portlet-app>
```

2.6.4 liferay-display.xml

定义Portlet默认的所属类别。Liferay Portal对Portlet实行按类别管理和划分用户权限。正如我们在用户策略中提到的，可以制定某个类别可用的用户组、用户和角色，方便权限控制。Liferay-display.xml规范由http://www.liferay.com/dtd/liferay-display_3_5_0.dtd 定义。

Liferay-display.xml 中，<display>下成组的<category>描述了可用的类别，其中 portlet 元素的 id 必须与 liferay-portlet.xml 的 portlet-name 保持一致，且在应用中唯一。

```
<display>
  <category name="category.test">
    <portlet id="TestPortlet" />
  </category>
</display>
```

2.6.5 liferay-layout-templates.xml

定义Portal可用的布局。正如我们在布局与品质中提到的那样，Portal采用tpl文件来规划桌面的布局。liferay-layout-templates.xml采用成组的layout-template来构建一个可用的布局列表。此xml的规范由http://www.liferay.com/dtd/liferay-layout-templates_3_6_0.dtd 来定义。

本文采用 Liferay Portal 默认的布局，暂时不需要定义自己的布局，故不准备深入讨论。读者有兴趣可以自己查看相关资料。

```
<layout-templates>
  <layout-template id="1_column" name="1 Column">
    <template-path>/html/layouttpl/1_column。tpl</template-path>
  </layout-template>
  <layout-template id="2_columns_i" name="2 Columns (50/50)">
    <template-path>/html/layouttpl/2_columns_i。tpl</template-path>
  </layout-template>
  <layout-template id="3_columns" name="3 Columns">
    <template-path>/html/layouttpl/3_columns。tpl</template-path>
  </layout-template>
  <layout-template id="1_2_1_columns" name="1-2-1 Columns">
    <template-path>/html/layouttpl/1_2_1_columns。tpl</template-path>
  </layout-template>
</layout-templates>
```

2.6.7 liferay-look-and-feel.xml

定义Portal可用品质的模板、图片、样式表等等，定义完毕后，Portal可以通过“布局与品质”管理工具来进行品质的切换。Liferay-look-and-feel.xml 规范由http://www.liferay.com/dtd/liferay-look-and-feel_3_5_0.dtd 定义。

本文采用 Liferay Portal 默认的品质，不准备对品质的自定义深入探讨。有兴趣的读者可以查看相关资料。

第二部分 Liferay Portal 二次开发

本部分主要内容

GenericPortlet 自定义 Portlet 类 部署描述文件

第三章 开发自己的 Portlet

在了解了 Liferay Portal 的基础架构，初步体会 Liferay Portal 良好的个性化定制之后，本章将开始 Liferay Portal 二次开发之旅，讲述并扩展 Portlet 的超类 GenericPortlet，创建或者修改部署描述文件，构建属于自己的 Portlet。

第一节 重要的基类：GenericPortlet

像 Servlet 一样，编写的 Portlet 也必须直接或者间接的扩展基类 GenericPortlet，这个是由 JCP 针对 Portal 提出的 JSR168 规范定义的。只要扩展自规范的 GenericPortlet，所有的 Portlet 都可以在支持 JSR168 规范的 Portal 服务器上运行。

GenericPortlet 统一定义了可供 Portal 容器识别和调用的方法，包括：

public Init()：初始化；

public Init(PortletConfig)：初始化；

public getInitParameter(String)：取得在 Portlet.xml 中定义的初始化参数；

public getInitParameterNames()：取得在 Portlet.xml 中定义的全部初始化参数；

public getPortletConfig()：取得包含初始化参数的配置对象 PortletConfig 实例；

public getPortletContext()：取得 Portlet 上下文；

public getPortletName()：取得在 Portlet.xml 中定义的 Portlet 名称。

public getResourceBundle(Locale)：取得 Portlet 国际化的 Resource Bundle；

protected getTitle(RenderRequest)：取得 Portlet 的标题；

protected doView(RenderRequest, RenderResponse)：Portlet 浏览模式的处理方法；

protected doEdit(RenderRequest, RenderResponse)：Portlet 编辑模式的处理方法；

protected doHelp(RenderRequest, RenderResponse)：Portlet 帮助模式的处理方法；

protected doDispatch(RenderRequest, RenderResponse)：Portlet 行为分发；

protected processAction(RenderRequest, RenderResponse)：Portlet 处理 Action Request 的方法；

protected render(RenderRequest, RenderResponse)：Portal 处理 Render Request 的方法；

public destroy()：Portlet 销毁，终止其生命周期。

在 Portlet Portal 运行的时候，doView、doEdit、doHelp 三个方法分别被调用，用以生成 Portlet 标记。同样也可以调用 Servlet 生成 Portlet 标记，或者不调用 JSP 或者 Servlet，直接在方法中得到 PrintWriter 然后用最简单的 pw.println()打印出内容。这个过程类似 Servlet，如下：

```
PrintWriter pw = renderResponse.getWriter();
```

```
pw.println("Hello, world!");
```

与 Servlet 类似，可以使用 `getInitParameter(String s)` 得到配置文件中 Portlet 的初始值，只不过 Servlet 在 `web.xml` 中，而 Portlet 在 `portlet.xml` 中。

```
portlet.xml :
<init-param>
  <name>jspView</name>
  <value>/jsp/view.jsp</value>
</init-param>
```

针对如上 `portlet.xml` 中的初始化信息，可以采用如下的调用方式：

```
SimplePortlet.java :
String jspName = getPortletConfig().getInitParameter("jspView");
```

第二节 Portlet 标签

跟 Servlet 一样，Portlet 也自定义了很多灵活的标签。通过这些标签，可以调用 Portlet 内部的参数比如 `renderResponse`、`renderRequest`、`PortletConfig` 等，在 JSP 中跟 Portlet 通信。当然，在使用之前，除了要在 `web.xml` 中声明标签库外，还要在 JSP 的头部声明标签库调用：

```
<%@ taglib uri="http://java.sun.com/portlet" prefix="portlet" %>
```

3.2.1 defineObjects 标签

在使用 Portlet 典型标签之前，要见声明 `<portlet:defineObjects/>`，这样才可以使用其他的标签。`defineObjects` 中间不允许定义任何属性和包含任何内容。

3.2.2 renderURL 标签

属性	值类型	对应值
WindowState	String	minimized normal maximized。
portletMode	String	view , edit , help
var	String	
secure	String	true false

```
<portlet:renderURL portletMode="view" windowState="maximized">
  <portlet:param name="number" value="1"/>
  <portlet:param name="page" value="2"/>
</portlet:renderURL>
```

创建一个当前 RenderURL，当访问它时将使 Portlet 窗口变为最大化状态，模式变为浏览。<portlet:param/>子元素会在生成的 RenderURL 中增加 number、page 两个参数和值。

3.2.3 actionURL 标签

属性	值类型	对应值
windowState	String	minimized normal maximized。
portletMode	String	view , edit , help
var	String	
secure	String	true false

```
<portlet:actionURL windowState="normal" PortletMode="edit">
  <portlet:param name="action" value="login"/>
</portlet:actionURL>
```

创建一个当前 ActionURL，当访问它时将使 Portlet 窗口变为正常状态，模式变为编辑。<Portlet:param/>子元素会在生成的 ActionURL 中增加 action 参数和值。

renderURL 和 actionURL 两个标签在诸如生成 form 表单的 action 等方面特别有用。

3.2.4 param 标签

属性	值类型
name	String

用在 renderURL 和 actionURL 标签内部，用来在生成的 URL 中增加参数和值。param 标签不运行 body 内容存在。

3.2.5 namespace 标签

为目前的 Portlet 产生一个唯一的 Value，防止与其他 Portlet 或者 Portal 上面的 Value 冲突。

上述标签的具体属性及其约束，请参阅 {PORTAL_HOME}/liferay/WEB-INF/tld/liferay-portlet.tld。

第三节 Portal 的对象

JSR168 给 Portal 定义了几个特别的对象，用来操作 Portal 特有的信息。这些对象跟 Servlet 的对象有点类似，又有点不同。这些对象都封装在 {PORTAL_HOME}/common/lib/ext/portlet.jar 包中，具体支持实现要视 Portal 服务器而定。

3.3.1 Request 对象

Portlet 中的 Request 与 Servlet 的 Request 一样接受客户端发送的请求，但是与 Servlet 不同，Portlet 的 Request 分为 Action Request 及 Render Request 两种类型，因此 Portlet 接口中定义了两种方法来处理不同的 Request。分别是 `processAction(ActionRequest request, ActionResponse response)` 和 `render(RenderRequest request, RenderResponse response)`，分别用以处理 Action Request 和 Render Request。某种意义上讲，`render` 方法类似 Servlet 中的 `service` 方法，`doView`，`doEdit`，`doHelp` 方法又类似 `doGet`，`doPost` 方法。

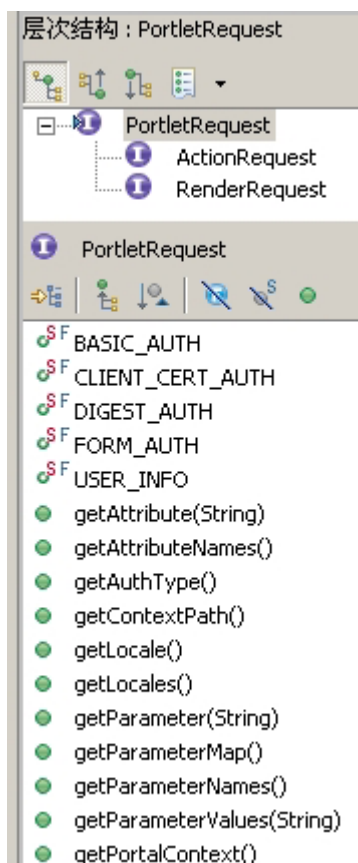


图 3.3.1-1

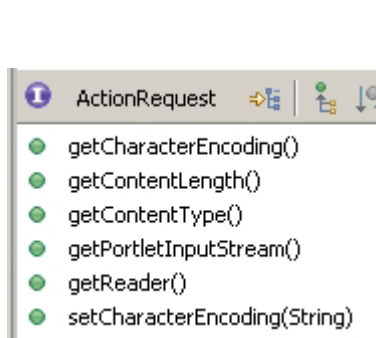


图 3.3.1-2

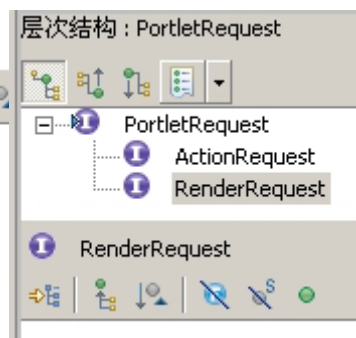


图 3.3.1-3

、RenderRequest 和 ActionRequest

PortletRequest 分为 RenderRequest 和 ActionRequest 两种，分别由 `renderURL` 和 `actionURL` 来触发。`renderURL` 是 `actionURL` 的一种优化。Portlet 的开发过程中尽量使用 `renderURL` 而避免 `actionURL`。`actionURL` 适用于有确实的 Action（行为）的情况下。比如说，表单 form 提交后 Persistent 状态的改变、session 的改变、perference 的修改等等。`renderURL` 通常用来处理 Portlet 的导航。举个例子：

使用 `actionURL`：

```
<%
PortletURL pu = renderResponse.createActionURL();
pu.setParameter("ACTION", "LOGIN");
<form name="usrform" method="post" action="<%=pu.toString()%>">
%>
```

说明：表单提交最好使用 Post 方法而不是 Get 方法，因为某些 Portal 服务器可能会将内部状态编码到 URL 的 Query 字符串中。

使用 renderURL：

```
<%
PortletURL pu=renderResponse.createRenderURL();
Pu.setParameter("PAGE", Number);
%>
<a href="<%=pu%>">下一页</a>
```

、renderURL 和 actionURL 的处理方式

当客户端请求是由一个 renderURL 触发的时候，Portal 服务器会调用该 Portal 页面所有 Portlet 的 render 方法。

而当客户端请求是由一个 actionURL 触发的时候，Portal 服务器会先按用该页面所有 Portlet 的 processAction 方法再调用 render 方法。所以，要明确自己到底使用那种 URL 来出发客户端请求。

、RenderRequest 和 ActionRequest 的 parameter 参数作用范围

当客户端请求由一个 actionRequest 触发时，所有 parameter 参数的取得都必须在 processAction 方法中进行。比如：

```
public void processAction(ActionRequest req, ActionResponse res){
    String str = req.getParameter("ACTION");
    //response.setRenderParameter("ACTION", action);
}

public void doView(ActionRequest req, ActionResponse res){
    String str = req.getParameter("ACTION");
}
```

如上 processAction 方法中，getParameter 方法将能成功得到表单中的参数 ACTION 所对应的值，因为我们知道，当目标 Portlet 的 processAction 方法运行完后，Portlet Container 将调用 Portal 页面中所有 Portlet 的 render 方法。但是实际上 doView 方法中使用 getParameter 不会得到任何值。但是如果把 processAction 方法中注释了的一行解除注释的话，你就可以在 doView 方法中的得到参数 ACTION 对应的值。这说明 action request 的参数，render 方法中不可以直接取到。必须使用了 setRenderParameter 方法，再次传递一次。

3.3.2 Response 对象

与 Request 对象一样，Response 对象也有两种：RenderResponse 和 ActionResponse，分别用来封装对应的 RenderRequest 和 ActionRequest 的返回信息，比如重定向、窗口状态、Portlet 模式等。他们两者的父类 PortletResponse 拥有 setProperty 和 getProperty 两个方法，用来传递信息给 Portal 容器。

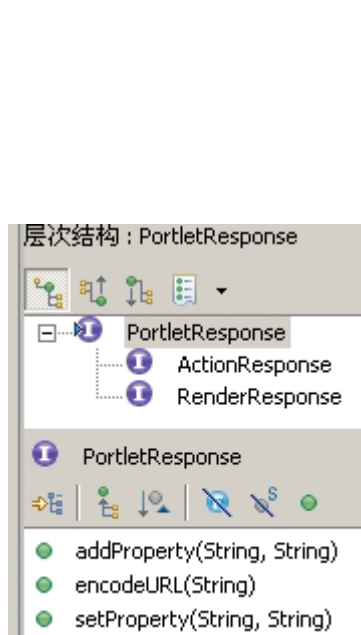


图 3.3.2-1

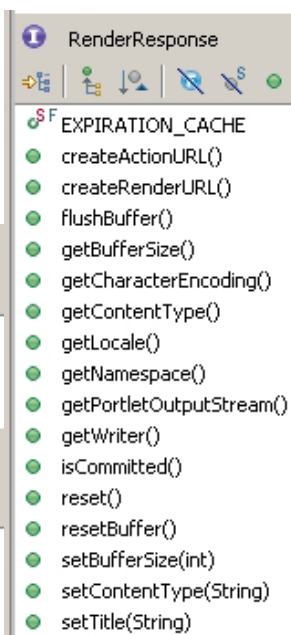


图 3.3.2-2

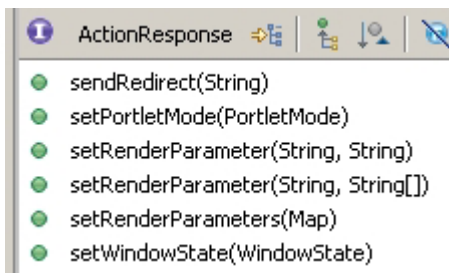


图 3.3.2-3

ActionResponse 主要用来处理以下功能：

- 重定向
- 改变窗口状态、Portlet 模式
- 传递 parameter 参数到 RenderRequest 中去

RenderResponse 主要用来提供以下功能：

- 设置 ContentType
- 得到 OutputStream 和 Writer 对象，用来输出页面内容
- Buffering 缓冲
- 设定 Portlet 的标题，但是必须在 Portlet 输出前调用，否则将被忽略

3.3.3 PortletConfig 对象

和 ServletConfig 对象类似，PortletConfig 对象提供对 Portlet 初始化信息以及 PortletContext 对象存取的方法。

和 ServletConfig 对象不同的是，PortletConfig 对象提供对 Portlet 的标题等资源的 I18N 支持，可以通过设定不同的 Resource Bundle 文件以提供多种语言支持。

3.3.4 Session 对象

由于容器不同，Portal 的 Session 对象与 Servlet 的 Session 对象略有不同。

由于 Portlet 处于 Portal 服务器的缘故，Portlet 的 Session 分为 Application Scope 和 Portlet Scope。两者的区别在于：

、Application Scope 范围的 Session 中保存的对象，对于同一个 Portlet 应用范围内的所有 Portlet 都是可用的。

、Portlet Scope 范围的 Session 中保存的对象，只对本 Portlet 可用，其他 Portlet 即使在一个应用中，也不可用。

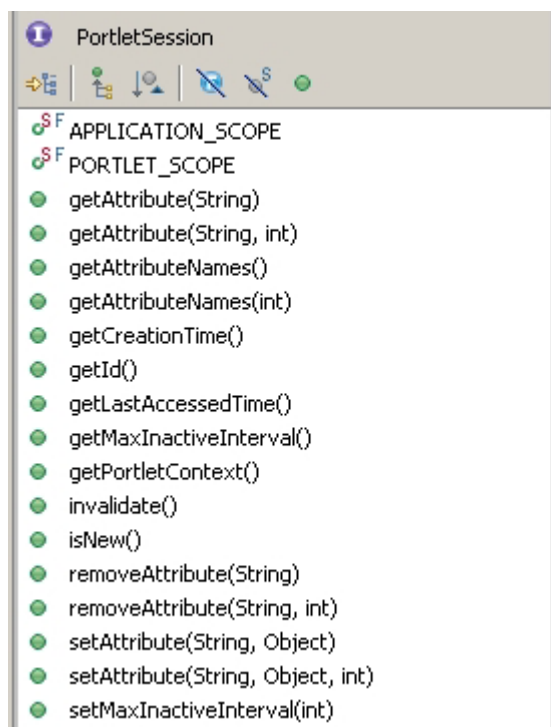


图 3.3.4-1

但是对于 Portlet 应用来说，可以通过 HttpSession 来访问。毕竟 Portlet 应用也是 Web 应用。在使用 Session 对象的时候，最好能明确指出使用的是那个 Scope 范围的 Session。比如：

```
<portlet:actionURL windowState="NORMAL" portletMode="view" var="pu1">
<portlet:param name="ACTION" value="ApplicationScope"/>
</portlet:actionURL>

<portlet:actionURL windowState="NORMAL" portletMode="view" var="pu2">
<portlet:param name="ACTION" value="PortletScope"/>
</portlet:actionURL>
```

这个 JSP 创建了两个 ActionURL，分别产生了两种 PortletSession 对象。

```
PortletSession ps = req.getPortletSession();
if(ps.getAttribute("PortletSession.AS", PortletSession.APPLICATION_SCOPE)!=null){
    app=ps.getAttribute("PortletSession.AS", PortletSession.APPLICATION_SCOPE).
toString();
}
if(ps.getAttribute("PortletSession.PS", PortletSession.PORTLET_SCOPE)!=null){
    Portlet=ps.getAttribute("PortletSession.PS", PortletSession.PORTLET_SCOPE).
toString();
}
```

以上代码根据需要取得不同 Scope 范围的 Session 对象值。

同一个应用下，可以直接通过 `ServletSession` 取得 `PortletSession.APPLICATION_SCOPE` 范围下的 Session 对象值。

```
HttpSession se = request.getSession();
if(se.getAttribute("PortletSession.AS")!=null){
    app=se.getAttribute("PortletSession.AS");
}
```

3.3.5 Preference 对象

Preference 对象被设计用来实现用户的个性化设置，可以帮助用户对 Portlet 进行符合用户需求的显示定制和行为定制，可以替代部分的数据库功能。需要指出的是，Preference 对象只是用来存取简单的配置信息，并不能完全替代数据库应用。

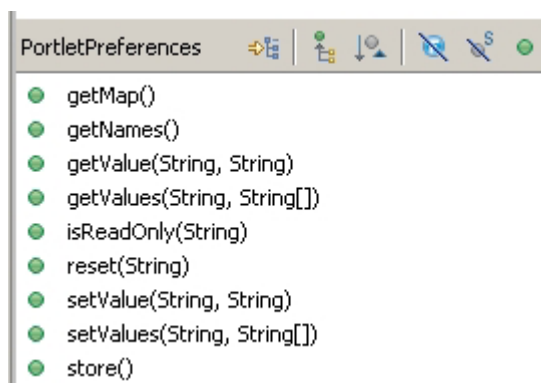


图 3.3.5-1

Preference 对象对于配置信息采用键-值的形式存取，用户可以将需要的信息暂时保存在 Preference 中。

```
PortletPreference p= req.getPortletPreferences();
p.setValue("educhina.username" , "educhina");
p.store();
```

Preference 对象用来存取用户的个性化信息，所以不同用户的 Preference 对象不能共享，这点跟 Session 不同。

可以在 Portlet.xml 中配置 Preference 信息，如下：

```
<portlet-preferences>
  <preference>
    <name>educhina.username</name>
    <value>educhina</value>
    <read-only>true</read-only>
  </preference>
</portlet-preferences>
```

另外，还可以配套使用 `PreferencesValidator` 对象，对 Portlet 的 Preference 在存储之前进

行验证，以确保 Preference 的正确性。

具体规范可以参照http://java.sun.com/xml/ns/Portlet/Portlet-app_1_0.xsd 的<complexType name="preferenceType">部分。

第四节 编写自己的 Portlet 类

Liferay Portal 内部集成了 78 个 Portlet，包括直接用 PrintWriter 输出的、调用 JSP 输出的、调用 Servlet 输出的，数据来源有直接从数据库取得的、通过 Web Service 取得的等等。这里，我们只讲述直接用 PrintWriter 输出的和调用 JSP 输出的，目的在于讲述如何编写自己的 Portlet 类。其他的与此类似，不赘述。

3.4.1 开发环境

IDE：Eclipse V3.0.1

JDK：V1.4.2_06

ANT：V1.6.2

Tomcat：V5.0（集成在 Liferay Portal 中）

Liferay Portal：liferay-portal-pro-3.6.0-tomcat

3.4.2 准备工作

- 、安装 JDK V1.4.2_06，在系统环境变量中增加变量 JAVA_HOME，指向 JDK 安装目录。
- 、安装 ANT，在系统环境变量中增加变量 ANT_HOME，指向 ANT 安装目录。
- 、下载并启动 Eclipse V3.0.1。
- 、下载并解压缩 liferay-portal-pro-3.6.0-tomcat.zip 到某一文件夹，该文件夹即为 {PORTAL_HOME}。
- 、在 Eclipse 中新建一个 Java 项目，命名为 TestPortal，路径为 D:\TestPortal，将 {PORTAL_HOME}\common\ext\portlet.jar 以外部 jar 的形式添加到库中。下文中，D:\TestPortal 将以 {APP_HOME} 代称。在 {APP_HOME} 下创建文件夹 webapp、deploy、bak。项目缺省输出文件夹为 {APP_HOME}\webapp\WEB-INF\classes。
- 、在 {APP_HOME}\webapp\WEB-INF 目录下创建 web.xml，内容如下：

```
<?xml version="1.0"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
2.3/EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <display-name>TestPortal</display-name>
</web-app>
```

- 、在 {APP_HOME}\webapp\WEB-INF 下创建 tld 文件夹，将 {PORTAL_HOME}\liferay\WEB-INF\tld\liferay-portlet.tld 拷贝到创建的 tld 文件夹下，备用。

、新建一个 Java 包 com.educhina.portal 。

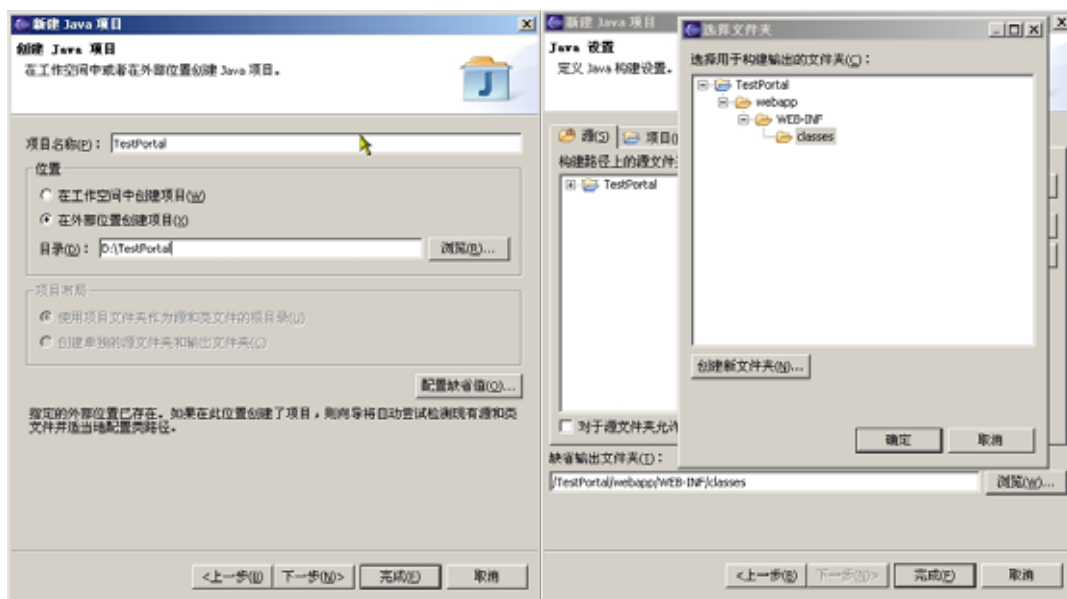


图 3.4.2-1

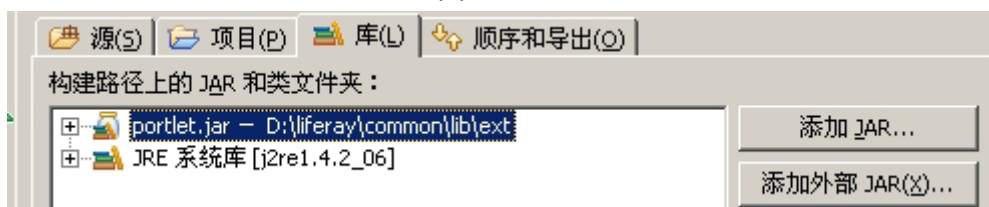


图 3.4.2-2

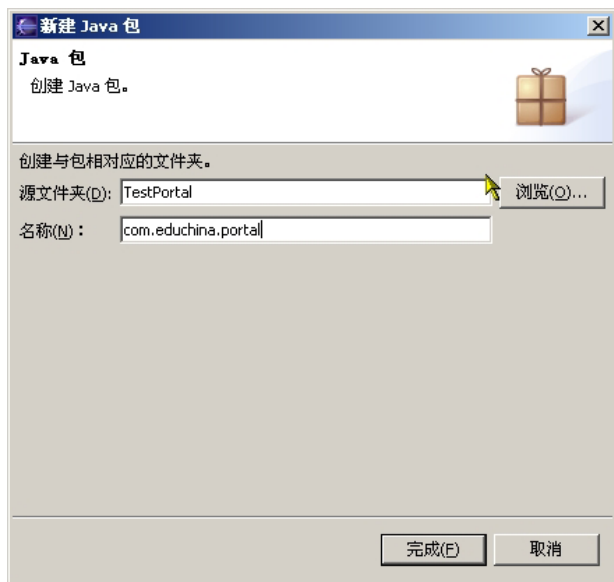


图 3.4.2-3

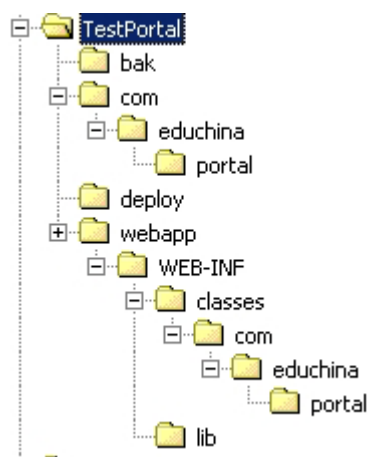


图 3.4.2-4

3.4.3 HelloWorldPortlet

HelloWorldPortlet 类计划用单纯的 PrintWriter 输出 Portlet 标记片断。在包 com.educhina.portal 下新建 Java 类 HelloWorldPortlet，这个类必须扩展自 javax.Portlet.GenericPortlet 类。设计让 HelloWorldPortlet 支持浏览和编辑两种模式，所以

HelloWorldPortlet 重写 doView 和 doEdit 方法。简单的代码如下：

```
package com.educhina.portal;
import java.io.IOException;
import javax.Portlet.GenericPortlet;
import javax.Portlet.PortletException;
import javax.Portlet.RenderRequest;
import javax.Portlet.RenderResponse;

public class HelloWorldPortlet extends GenericPortlet{
    public void doView(RenderRequest req, RenderResponse res)
        throws IOException, PortletException {
        res.setContentType("text/html");
        res.getWriter().println("HelloWorld!");
    }
    public void doEdit(RenderRequest req, RenderResponse res)
        throws IOException, PortletException {
        res.setContentType("text/html");
        res.getWriter().println("HelloWorld!");
    }
}
```

doView 和 doEdit 方法从 RenderRequest 取得 PrintWriter 对象，直接输出一个 String 字符 “HelloWorld !”。这个 String 字符将作为 HelloWorldPortlet 的片断内容。

3.4.4 HelloJSPPortlet

HelloJSPPortlet 类计划调用外部 JSP 输出。同样的，HelloJSPPortlet 也要扩展自 GenericPortlet 类。HelloJSPPortlet 调用 getPortletConfig().getInitParameter("..")方法，取得在 Portlet.xml 中配置的 view-jsp 和 edit-jsp 参数值，以此确定 JSP 页面的具体位置。然后调用 PortletRequestDispatcher 的 include 方法，将 JSP 页面加载到 RenderResponse。代码如下：

```

package com.educhina.portal;

import java.io.IOException;
import javax.Portlet.GenericPortlet;
import javax.Portlet.PortletException;
import javax.Portlet.PortletRequestDispatcher;
import javax.Portlet.RenderRequest;
import javax.Portlet.RenderResponse;

public class HelloJSPPortlet extends GenericPortlet{
    public void doView(RenderRequest req, RenderResponse res)
        throws IOException, PortletException {
        res.setContentType("text/html");
        String jspName = getPortletConfig().getInitParameter("view-jsp");
        PortletRequestDispatcher rd = getPortletContext().getRequestDispatcher(jspName);
        rd.include(req,res);
    }
    public void doEdit(RenderRequest req,RenderResponse res)
        throws IOException,PortletException {
        res.setContentType("text/html");
        String jspName = getPortletConfig().getInitParameter("edit-jsp");
        PortletRequestDispatcher rd = getPortletContext().getRequestDispatcher(jspName);
        rd.include(req,res);
    }
}

```

在{APP_HOME}/webapp 目录下创建 view.jsp 和 edit.jsp，view.jsp 代码如下，edit.jsp 类似：

```

<table cellpadding="8" cellspacing="0" width="100%">
<tr>
    <td>
        <font class="Portlet-font" style="font-size: x-small;">
            This is a <b>Sample JSP Portlet</b> used in viewing model。 Use this as a quick
            way to include JSPs。
        </font>
    </td>
</tr>
</table>

```

JSP 文件不能包含关于 HTML 的<head>、<body>、<html>的信息，只能包含原来位于<body></body>内部的 HTML 内容。那些<head>、<body>、<html>信息由 Portal 页面来提供。

只有在 JSP 页面中使用<% @ taglib uri="http://java.sun.com/portlet" prefix="portlet" %>和<portlet:defineObjects/>，JSP 页面才可以直接操作 Portlet 的一些变量，比如：renderResponse、

renderRequest、portletConfig。

第五节 修改 Web 部署描述文件

正如 2.6.1 所指出的那样,要保证 Portlet 能够在 Liferay Portal 成功部署,必须对 web.xml 进行必要的修改,添加 Portlet 监听器、Servlet 映射、Portlet 标签库。在先前 web.xml 的<display>节点下增加如下内容:

```
<listener>
<listener-class>com.liferay.portal.servlet.PortletContextListener</listener-class>
</listener>
<servlet>
  <servlet-name>HelloWorldPortlet</servlet-name>
  <servlet-class>com.liferay.portal.servlet.PortletServlet</servlet-class>
  <init-param>
    <param-name>portlet-class</param-name>
    <param-value>com.educhina.portal.HelloWorldPortlet</param-value>
  </init-param>
  <load-on-startup>0</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>HelloWorldPortlet</servlet-name>
  <url-pattern>/HelloWorldPortlet/*</url-pattern>
</servlet-mapping>
<servlet>
  <servlet-name>HelloJSPPortlet</servlet-name>
  <servlet-class>com.liferay.portal.servlet.PortletServlet</servlet-class>
  <init-param>
    <param-name>Portlet-class</param-name>
    <param-value>com.educhina.portal.HelloJSPPortlet</param-value>
  </init-param>
  <load-on-startup>0</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>HelloJSPPortlet</servlet-name>
  <url-pattern>/HelloJSPPortlet/*</url-pattern>
</servlet-mapping>
<taglib>
  <taglib-uri>http://java.sun.com/portlet</taglib-uri>
  <taglib-location>/WEB-INF/tld/liferay-portlet.tld</taglib-location>
</taglib>
```

其中,<listener>节点是增加一个监听器,以便 Liferay Portal 监听所有针对 Portlet 的操作。<servlet>以及<servlet-mapping>是将上述两个 Portlet 类加入 Servlet 容器中。Portlet 类实

质上也是 Servlet。<tablib>是将 Liferay Portal 标签库加入列表中，以便 JSP 调用。

第六节 创建 Liferay Portal 部署描述文件

修改完 web.xml 之后，还要创建 2.6 所说的三个 Portlet 部署描述文件：portlet.xml、liferay-portlet.xml、liferay-display.xml。

、portlet.xml

portlet.xml 定义 Portlet 的初始化信息。这里，我们在 portlet.xml 中增加两个 Portlet 节点，分别代表 HelloWorldPortlet 和 HelloJSPPortlet。其中，HelloWorldPortlet 支持 PrintWriter 输出，HelloJSPPortlet 支持 JSP 输出；两者都支持浏览和编辑两种模式。HelloJSPPortlet 需要定义两个 init 参数，告诉系统 JSP 文件的位置。

```
<portlet>
  <portlet-name>HelloWorldPortlet</portlet-name>
  <display-name>HelloWorldPortlet</display-name>
  <portlet-class>com.educhina.portal.HelloWorldPortlet</portlet-class>
  <expiration-cache>0</expiration-cache>
  <supports>
    <mime-type>text/html</mime-type>
  </supports>
  <supports>
    <mime-type>text/html</mime-type>
    <Portlet-mode>edit</Portlet-mode>
  </supports>
  <portlet-info>
    <title>HelloWorldPortlet</title>
    <short-title>HelloWorldPortlet</short-title>
    <keywords>HelloWorldPortlet</keywords>
  </portlet-info>
  <security-role-ref>
    <role-name>guest</role-name>
  </security-role-ref>
  <security-role-ref>
    <role-name>power-user</role-name>
  </security-role-ref>
  <security-role-ref>
    <role-name>user</role-name>
  </security-role-ref>
</portlet>
```

```
<portlet>
  <portlet-name>HelloJSPPortlet</portlet-name>
  <display-name>HelloJSPPortlet</display-name>
  <portlet-class>com.educhina.portal.HelloJSPPortlet</portlet-class>
  <init-param>
    <name>view-jsp</name>
    <value>/view.jsp</value>
  </init-param>
  <init-param>
    <name>edit-jsp</name>
    <value>/edit.jsp</value>
  </init-param>
  <expiration-cache>0</expiration-cache>
  <supports>
    <mime-type>text/html</mime-type>
  </supports>
  <supports>
    <mime-type>text/html</mime-type>
    <Portlet-mode>edit</Portlet-mode>
  </supports>
  <portlet-info>
    <title>HelloJSPPortlet</title>
    <short-title>HelloJSPPortlet</short-title>
    <keywords>HelloJSPPortlet</keywords>
  </portlet-info>
  <security-role-ref>
    <role-name>guest</role-name>
  </security-role-ref>
  <security-role-ref>
    <role-name>power-user</role-name>
  </security-role-ref>
  <security-role-ref>
    <role-name>user</role-name>
  </security-role-ref>
</portlet>
```

、liferay-Portlet.xml

Liferay-Portlet.xml 主要定义 Portlet 的模板、实例总数、是否允许重复定义等。同样的，我们增加了两个<Portlet>节点，代表 HelloWorldPortlet 和 HelloJSPPortlet。

```
<?xml version="1.0"?>
<!DOCTYPE liferay-Portlet-app PUBLIC "-//Liferay//DTD Portlet Application
3.5.0//EN" "http://www.liferay.com/dtd/liferay-Portlet-app_3_5_0.dtd">
<liferay-portlet-app>
  <portlet>
    <portlet-name>HelloWorldPortlet</portlet-name>
    <struts-path>HelloWorldPortlet</struts-path>
    <use-default-template>true</use-default-template>
    <instanceable>true</instanceable>
  </portlet>
  <portlet>
    <portlet-name>HelloJSPPortlet</portlet-name>
    <struts-path>HelloJSPPortlet</struts-path>
    <use-default-template>true</use-default-template>
    <instanceable>true</instanceable>
  </portlet>
  <role-mapper>
    <role-name>administrator</role-name>
    <role-link>Administrator</role-link>
  </role-mapper>
  <role-mapper>
    <role-name>guest</role-name>
    <role-link>Guest</role-link>
  </role-mapper>
  <role-mapper>
    <role-name>power-user</role-name>
    <role-link>Power User</role-link>
  </role-mapper>
  <role-mapper>
    <role-name>user</role-name>
    <role-link>User</role-link>
  </role-mapper>
</liferay-portlet-app>
```

、liferay-display.xml

liferay-display.xml 定义 Portlet 所属类别。Liferay Portal 默认定义了一个 category.test 类别，这里，我们将 HelloWorldPortlet 和 HelloJSPPortlet 归属到 category.test。

```
<?xml version="1.0"?>
<!DOCTYPE display PUBLIC "-//Liferay//DTD Display 3.5.0//EN"
"http://www.liferay.com/dtd/liferay-display_3_5_0.dtd">
<display>
  <category name="category.test">
    <portlet id="HelloWorldPortlet" />
    <portlet id="HelloJSPPortlet" />
  </category>
</display>
```

至此，一个简单的 Portlet 就开发完成了。接下来，我们把它部署到 Liferay Portal 上。

第三部分 Liferay Portal 部署

本部分主要内容

Portlet 部署 ANT 管理 Portlet

第四章 部署自己的 Portlet

Liferay Portal 跟 Tomcat5.0 集成在一起，从本质上讲，liferay-portal-pro-3.6.0-tomcat.zip 是一个 Tomcat 压缩包，只是其中将 liferay 作为默认应用，并将跟 Portlet 有关的操作都交给 liferay 应用处理而已。因此，Liferay Portal 支持所有针对 Tomcat5.0 的部署方式，包括：手动部署、Ant 部署，并且支持热部署。

第一节 手动部署

手动部署可以采用拷贝文件夹、war 部署、编写部署文件三种方式：

、拷贝文件夹：与单纯的 Tomcat 一样，我们可以将{APP_HOME}\webapp 目录拷贝到{PORTAL_HOME}\webapps\下，该 webapp 目录名为 TestPortal。启动 Liferay Portal（双击{PORTAL_HOME}\bin\startup.bat）即可。

、war 部署：或者将{APP_HOME}\webapp 打包成 TestPortal.war，拷贝 war 到{PORTAL_HOME}\webapps\下，启动 Liferay Portal，让 Tomcat 自动解压。在命令行模式下切换到{APP_HOME}\webapp 目录，执行 jar cvf TestPortal.war *。



TestPortal.war

图 4.1-1

、编写部署文件：

{PORTAL_HOME}\conf\Catalina\localhost 目录下，创建 TestPortal.xml 文件，内容如下：

```
<Context    path="/TestPortal"    docBase="D:\TestPortal\webapp"    debug="0"
reloadable="true" crossContext="true">
</Context>
```

部署成功后，登录 Liferay Portal，可以在桌面底部的下拉列表中看到 HelloWorldPortlet 和 HelloJSPPortlet 两个 Portlet。将它们添加到桌面中。



图 4.1-2



图 4.1-3

第二节 Ant 自动部署

确保之前已经安装 Apache Ant，并正确添加 ANT_HOME 到系统环境变量。

- 、拷贝之前打包的 TestPortlet.war 到 {APP_HOME}/deploy 目录；
- 、从 <http://prdownloads.sourceforge.net/lportal/Portlet-deployer-3.6.0.xml> 下载 Portlet-deployer-3.6.0.xml 到 {APP_HOME}\deploy，改名为 build.xml 以便 Ant 自动加载；
- 、确保 JDK 1.4.2 和 Ant 1.6 安装成功，并配置到系统环境变量；
- 、确保 Tomcat 或者其他服务器已经正确安装，或者 Liferay Portal 正常安装。

编辑 build.xml，使其只想你的应用服务器或者 Servlet 容器。比如，如果你安装 JBoss+Jetty 到 /opt/liferay 目录，那么编辑 build.xml，确保只有 JBoss+Jetty 部分没有被注释，修改 app.server 属性为 /opt/liferay。

Build.xml 默认是开启 JBoss+Jetty 部分，本文采用的是 Tomcat 集成包，所以将 JBoss+Jetty 部分注释掉，开始 Tomcat 部分。修改 app.server.dir 属性，指向 {PORTAL_HOME}。如下图：

```

64 - <!-- Tomcat
65 -->
66 - <!--
67 <property name="app.server.type" value="tomcat" />
68 <property name="app.server.dir" value="/C:\SR00T\liferay\ext\servers/${app.server.type}" />
69 <property name="app.server.deploy.dir" value="${app.server.dir}/webapps" />
70 <path id="project.classpath">
71 <pathelement location="${env.ANT_HOME}/lib/ant.jar" />
72 <fileset dir="${app.server.dir}/common/lib/ext" />
73 <pathelement location="${app.server.dir}/common/lib/servlet-api.jar" />
74 </path>
75
76 -->

```

图 4.2-1

- 、命令行切换到 {APP_HOME}/deploy 目录，执行 ant deploy，系统会自动将 TestPortal.war 解压，必要时修改 web.xml、portlet.xml 等部署文件，将解压后的 TestPortal 文件夹拷贝到 {PORTAL_HOME}\webapps 目录下。

启动 Liferay Portal 之前，建议先确认修改后的 web.xml、portlet.xml 等部署文件是否正确。

第三节 加入 Liferay Portal 自有列表

之前我们提到过，Liferay Portal 集成了 78 个默认的 Portlet 应用。这些应用都通过 {PORTAL_HOME}\liferay\WEB-INF\ 目录下的 portlet.xml、liferay-portlet.xml、liferay-display.xml 描述。我们只要更改这些描述文件，就可以将我们自己的应用加入到 Liferay Portal 的 Portlet 列表中了，效果跟手动部署和 Ant 自动部署一样。

- 、拷贝 {APP_HOME}\webapp 目录的内容到 {PORTAL_HOME}\liferay\html\Portlet 目

录下，更改文件夹名称为 TestPortal。

- 、将 TestPortal\WEB-INF\classes 文件夹剪切到 {PORTAL_HOME}\liferay\WEB-INF 目录下。

- 、将 TestPortal\WEB-INF\web.xml 中 <servlet>、<servlet-mapping> 的内容合并到 {PORTAL_HOME}\liferay\WEB-INF\web.xml 中。删除 TestPortal\WEB-INF\web.xml。

- 、将 TestPortal\WEB-INF\Portlet.xml 中关于 HelloWorldPortlet 和 HelloJSPPortlet 的 <portlet> 的内容合并到 {PORTAL_HOME}\liferay\WEB-INF\portlet.xml 中。删除 TestPortal\WEB-INF\portlet.xml。

- 、将 TestPortal\WEB-INF\liferay-portlet.xml 中关于 HelloWorldPortlet 和 HelloJSPPortlet 的 <portlet> 的内容合并到 {PORTAL_HOME}\liferay\WEB-INF\liferay-portlet.xml 中。删除 TestPortal\WEB-INF\liferay-portlet.xml。

- 、将 TestPortal\WEB-INF\liferay-display.xml 中关于 HelloWorldPortlet 和 HelloJSPPortlet 的 <portlet> 的内容合并到 {PORTAL_HOME}\liferay\WEB-INF\liferay-display.xml 中。删除 TestPortal\WEB-INF\liferay-display.xml。

这个方法比较复杂，而且不容易扩展和调试，通常不建议采用。

第四节 普通 Java Web 应用转化为 Portlet 应用

随着开发的深入，我们希望能够将原来的 Java Web 应用迁移到 Liferay Portal，构建真正的企业门户。Liferay Portal 灵活的二次开发机制，允许用户将各种各样的内容集成到 Portal 平台上来，消除信息孤岛。将一个 Java Web 应用转化为 Portlet 应用的步骤如下：

- 、撰写扩展自 GenericPortlet 的 Portlet 和 JSP 页面。这个 Portlet 可以使用 PrintWriter 输出或者调用 JSP 页面输出方式。通常，如果 Java Web 应用是采用 MVC 三层模式，那么只需要更改 View 层就可以了。

- 、修改 web.xml，增加 2.6.1 所述的 Portlet 监听器和 Portlet 标签库，增加针对上步骤所写的 servlet 和 servlet 映射。

```
<servlet>
  <servlet-name>yourPortlet</servlet-name>
  <servlet-class>com.liferay.portal.servlet.PortletServlet</servlet-class>
  <init-param>
    <param-name>portlet-class</param-name>
    <param-value>full.name.of.yourPortlet</param-value>
  </init-param>
  <load-on-startup>0</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>yourPortlet</servlet-name>
  <url-pattern>/yourPortlet/*</url-pattern>
</servlet-mapping>
```

- 、创建 portlet.xml，增加相应的 Portlet 定义信息，规范参考 2.6.2。
- 、创建 liferay-portlet.xml，增加相应的 Portlet 定义信息，规范参考 2.6.3。
- 、创建 liferay-display.xml，增加相应的 Portlet 类别定义信息，规范参考 2.6.4。

、拷贝 portlet.jar 和 liferay-Portlet.tld 到当前应用。其中，portlet.jar 是 Portlet API 包，作用类似 servlet-api.jar，位于 {PORTAL_HOME}\common\lib\ext\liferay-portlet.tld 是 Liferay Portal 提供的 Portlet 标签库。

、选择适当的部署方式，将修改后的 Java Web 应用部署到 Portlet 平台上。

第四部分 附录

本部分主要内容
资源网站 Portlet 范例 参考资料 后序

第五章 相关资源

作为一个开源的门户产品，Liferay Portal 已经比较成熟，有比较齐全的文档。随着应用的深入，开源免费的中文化文档也在陆续出现。

第一节 资源网站

Liferay Portal 官方网站：<http://www.liferay.com>

Liferay Portal 中文网站：<http://www.liferay.cn>

Liferay Portal 论坛：<http://forums.liferay.com>

Tracker：<http://support.liferay.com>

邮件列表：<http://sourceforge.net/mailarchive/forum.php?forum=lportal-development>

JavaLobby专题：<http://www.javalobby.org/articles/liferay/>

OSQS专题：<http://cstsolaris.cst.nait.ab.ca/ist410/gerry/liferay/index.jsp>

Leonardsoko1 专题：<http://www.leonardsokol.com/liferay/>

Developer专题：http://www.developer.com/java/web/article.php/10935_3372881_1

第二节 示例

Liferay Portal 随程序包提供了丰富的 documentation，其中的 Portlet Examples 对 Portal 内置的 Hello World、IFrame、Calendar、Message Boards、Mail 五个 Portlet 进行了比较详细的解说。启动 Liferay Portal 后，浏览这里：

<http://localhost/web/guest/documentation/development/Portlet>

另外，Liferay Portal 还在官方网站上提供了 Sample Layout Template、Sample Portlet、Sample Themes 供下载。其中，Sample Portlet 包括 Sample JSP Portlet、Sample Struts Portlet、Sample JSF SUN Portlet、Sample JSF MyFaces Portlet。浏览这里：

http://localhost/web/guest/downloads/sample_Portlet

第六章 参考资料

、文档

《JSR168 PORLET 标准手册汉化整理》

作者：Jini 等

《Portlet 应用开发(JSR168)》

作者：Terry Lee

《(原创翻译)Liferay-Portal 架构》

作者：eamoi

、网站

<http://www.liferay.com>

<http://www.liferay.cn>

后序

研究 Liferay Portal 属于半路出家。从开始到本文完成 ,俩月有余。作为一个开源的 Portal 产品 , Liferay 的确值得称许 , 虽然还有不少 bug。在本文截稿的时候 , Liferay Portal V3.6.1 已经发版 , 新版本在拖拉 Portlet、Spring 远程传输和布局热部署方面有比较大的提升。本文不会就此终结 , 暂称 V1.0 , 作为前段工作的总结。本文的用户策略部分参考了同事 Kevin 的文档 , 特此感谢。

PS : 写文章真的很费脑筋。