基于 Oracle AS Portal 下的 Java Portlet 开发

严墨洁，田 斌

（武汉理工大学 信息工程学院，湖北 武汉 430070）

摘 要：Oracle AS Portal 是 Oracle 应用服务器的一个组成部分，用于构建和部署企业级门户系统。PDK－Java 提供了一个在 Portal 下用 Java 开发 Portlet 的框架，从而简化了对 Java Portlets 的开发。文中介绍了基于 PDK－Java 框架下进行 Java Portlet 开发的技术步骤，并实现了多页面 Portlet 和支持 Struts 的 Portlet 两种不同的 Java Portlet 开发。

关键词：Oracle AS Portal；PDK－ Java API；Struts

中图分类号：TP311.52 **文献标识码**：A **文章编号**：1673－629X(2007)03－0128－04

Developing Java Portlet Based on Oracle AS Portal

YAN Mo-jie, TIAN Bin

(School of Information Engineering, Wuhan University of Technology, Wuhan 430070, China)

Abstract: Oracle AS Portal is one of the components of Oracle application server, which is used to build and deploy the enterprise portal system. PDK－Java provides a framework to develop the Portlet using Java, and this framework simplifies the development of Java Portlet. This paper introduces the technical development steps of Java Portlet under PDK－Java framework in details. Finally the paper implements two kinds of Java Portles: multipages Portlet and Struts Portlet.

Key words: Oracle AS Portal; PDK－ Java API; Struts

0 引 言

Oracle AS Portal 是 Oracle 应用服务器的一个组成部分，用于构建和部署企业级门户系统。它提供了一个安全的、可管理的环境来访问企业软件服务和信息资源。Oracle AS Portal 集成了一个创建 Portal 的框架，利用此框架开发人员可以创建、发布、管理个性化的 Portal 内容。Portal 是由一组 Portlet 组成的可定制的页面，Portlet 不仅仅是一个可重用、可插拔的 Web 组件，还是一种应用[1]。在门户系统上发布 Portlet 时，需要对使用者进行授权，用户可以根据自己所属权限定制自己的门户。Oracle AS Portal 提供了两种集成应用程序的方法[2]：(1)使用预先构建的 Portlet 访问企业应用程序；(2)创建新的 Portlet 以集成企业应用程序。

基于预先构建的 Portlet 的集成并不能满足所有用户的需求，因此 Oracle AS Portal 提供了多种构建 Portlet 的方法，主要有：(1)说明式的开发工具(Omni-Portlet, WebClipping)；(2)Portlet Builder；(3)编程的创建方法(如：Oracle Portal Development Kit(PDK)Java, PL \ SQL 和 JSR168 等)。

由于应用的扩展以及需求的多样性，简单的基于浏览器的 Portlet 开发方法不能满足实际的需求。因此，利用 Oracle AS PDK 提供的 API，通过 Java 编程定制具有特定业务逻辑的 Portlet 是一个重要的技术手段。

1 Oracle AS Portal 体系结构

企业级 Portal 把不同来源的内容结合到一起，使得用户通过单一的 Portal 页面就可以访问不同终端的应用，这是通过 Portal 自带的控制器 PPE (the Parallel Page Engine)处理终端用户的请求来实现的。当用户发出请求时，PPE 负责重获页面的元数据，并并行传回每个 Portlet 的内容、组装 HTML 部分，最后向浏览器传回完整的页面。如图 1 所示。

从图 1 可见，Portlet 代表的应用程序或信息源都是通过一个叫做 Provider 的实体与门户进行通信。每个门户必须有一个、且只能有一个 Provider。一个 Provider 可以有一个或多个显示底层应用程序或信息源的 Portlet。当一个 Provider 注册到门户实例后，它的 Provider 就可以放在该门户实例对应的门户页面

收稿日期：2006－05－31

作者简介：严墨洁(1983－)，女，湖北宜昌人，硕士研究生，研究方向为网络信息应用；田 斌，副教授，研究方向为信息安全技术、内容服务和发布技术、网络工程管理。

上。Portlet 一般可通过以下三种类型的 Provider 注册到 Oracle AS Portal 上:

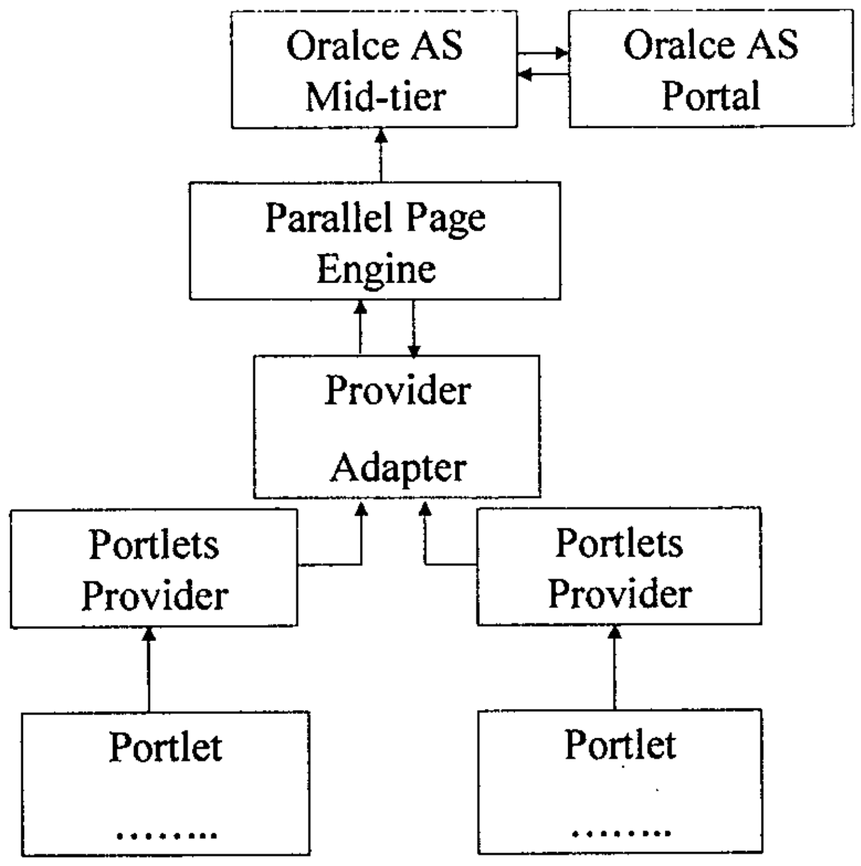

图 1　PPE 处理中断用户的请求

(1)Web Providers:可被部署到 J2EE 应用服务器上,然后通过 SOAP 和 Oracle AS Portal 通信,也可直接部署在 Oracle AS Portal 上。在开发 Portlet 时,将其以 Web Provider 形式展示,并具有远程部署 Portlet、利用已存在的 Web 应用代码来创建 Portlet 以及开发 Portlet 时可以使用标准的 Java 技术,如 Servlet,JSP 等优点。

(2)WSRP(Web Services for Remote Portlets)producers:作为一种标准,WSRP 可在基于标准的容器和 WSRP portal 之间实现合作性,其中,这些容器应是基于标准的语言,如:JSR168,.NET,Perl 等。

(3)Database Providers:通过 JDBC 实现和数据库的通信。

2　PDK－ Java 框架

PDK－Java 提供了一个在 Portal 下用 Java 开发 Portlet 的框架,从而简化了对 Java Portlets 的开发。同时它也支持现有的开发技巧和应用组件,如 JSP,Servlets 以及静态的 HTML 页面等,还屏蔽了 Oracle AS Portal 如何同 Providers 通信的问题。框架包含以下几个主要部分[2]:

(1)Provider Adapter:它隔离了 Oracle AS Portal 和 Web Providers 之间通信的 HTTP 语法,没有 adapter 就无法实现对 Portlets 的管理。

(2)Provider Interface:定义了执行 Java 语句所需要的 API。当 Provider Adapter 收到从 Portal 传来的 messages 时,通过调用 Provider Interface 解释这些 messages,将 Provider 的 response 转换为 Portal 能够理解的格式。同时它包含了一些 Java 类,为 Provider 的执行提供所需的方法。

(3)Provider Runtime:它包含了一系列缺省的类,这些类用来执行每个 Provider Interface,同时允许开发者利用 PDK－Java 提供的 render,personalization 和 security 框架。通过 Provider Runtime 可以让开发者把精力放在 Portlets 的开发上,而不是实现和 Portal 通信所需的基础构架搭建上。

(4)Provider Utilities:它提供了简化提交 Portlets 的方法,所包含的方法用于构建有效链接、提交 Portlet 容器(包含 head)、提交 HTML 表格以及支持缓存等。

3　Java Portlet 开发实例

3.1　开发 Java Portlet 的环境配置

Oracle 公司专门为其开发工具 JDeveloper 提供了 PDK 的 JDeveloper 插件:Oracle JDeveloper 10g Portal Extensions。该插件提供了开发 Portlet 必需的一些库文件,可以简化基于 Oracle PDK API 的 Portlets 的开发。插件安装步骤如下:

(1)下载 portal－addin.zip,解压,然后从命令行进入到该目录下;

(2)安装:java －jar portal－addin－install.jar <jdev home>;

(3)打开 JDeveloper,在 Web 层选择 Portlet,可看到有两个开发向导:"Java Portlet"和"PDK－Java Portlet",这就说明安装成功。

其次,要利用 PDK－Java 框架开发 PDK－Java Portlet,还需安装一个 standalone OC4J 来运行 PDK－Java 框架。它用于实现 Web Provider 的部署和注册。

3.2　PDK－Java Portlet 配置

在 JDeveloper 中利用向导生成 Portlet 的框架时,生成的以下几个文件对于部署和运行 Portlet 是必需的:provider.xml,web.xml,index.jsp,_default.properties 和 portlet2provider.properties。其中 provider.xml 这个文件对于 Portlet 的创建、管理是非常重要的。因为有了它,就可实现对所开发的 Portlet 的管理。其语法结构如下:

```
<? xml version='1.0' encoding='UTF-8'? >
<? providerDefinition version="3.1"? >
< provider class = "oracle.portal.provider.v2.DefaultProviderDefinition" >
  ……
  <preferenceStore class="oracle.portal.provider.v2.preference.FilePreferenceStore" >
  ……
  </preferenceStore>
  <portlet class="oracle.portal.provider.v2.Default-
```

```
PortletDefinition">
    ……
    <renderer class="oracle.portal.provider.v2.render.RenderManager">
    ……
    </renderer>
    ……
  </portlet>
</provider>
```

以下几个类对于实现 Provider 同 Portal 的通信非常重要[3]：

(1) DefaultProviderDefinition：它继承了基本的 ProviderDefinition 元数据类，包含的方法有 setSession，addContainerRenderer，setPassAllUrlParams，addPortlet 等。

(2) PreferenceStore：该抽象类实现了 PreferenceDataObjects 中数据装载和保存的存储机制，其具体实现类有 DBPreferenceStore 和 FilePreferenceStore。

(3) DefaultPortletDefinition：它在对基本 PortletDefinition 元数据类继承的基础上定义了 Portlet 的元数据，实现了对每个 Portlet 具体的描述。

(4) RenderManager：该类实现了 ManagedRenderer 接口，它定义了展现的具体页面。

3.3 多页面 PDK－Java Portlet 实现

该 Portlet 名为“portlet2”，它通过 Intraportlet link 和 OracleAS Portal 提供的 PAGE_LINK URLs 实现了同一个 Portlet 内多页面跳转以及表单的提交。部分实现代码与分析如下：

(1)该 Portlet 的 provider.xml：

```
<showPage>/htdocs/portlet2/first.jsp</showPage>
<pageParameterName>next_page</pageParameterName>
```

在 Show Mode 中的显示的页面为 first.jsp，通过“next_page”页面参数实现页面的跳转。

(2)first.jsp：

```
<% PortletRenderRequest prr=(PortletRenderRequest)
      request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
    NameValue[] linkParams=new NameValue[1];
    linkParams[0]=new NameValue(HttpPortletRendererUtil.portletParameter(request,"next_page"),"/htdocs/portlet2/second.jsp");
%>
……
<%=UrlUtils.constructHTMLLink(prr, UrlUtils.REFRESH_LINK,"Add to the list","", linkParams, true, true)%>
……
```

利用 portletParameter()方法将 String 类型的参数“next_page”转换成当产生 request 时能被 Portal 识别的、唯一的名字。利用 constructHTMLLink()方法创建一个 HTML LINK，其格式为：(PortletRenderRequest 的对象，链接类型，显示的文本，其他的关于 HTML 的属性，NameValue[]类型的参数，是否实现参数加密，是否实现参数替换)。

(3)second.jsp：

```
<%
PortletRenderRequest pRequest=(PortletRenderRequest)
request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
String paramNameSubmit="submit";
String paramNameNAME="n";
String paramNameADDRESS="a";
String qualParamNameNAME=
HttpPortletRendererUtil.portletParameter(pRequest, paramNameNAME);
String qualParamNameADDRESS=
HttpPortletRendererUtil.portletParameter(pRequest, paramNameADDRESS);
String qualParamNameSubmit=
HttpPortletRendererUtil.portletParameter(pRequest, paramNameSubmit);
String formName=UrlUtils.htmlFormName(pRequest,"submit_form");
%>
……
<form name="<%=formName%>" method="POST"
  action="<%=UrlUtils.htmlFormActionLink(pRequest,UrlUtils.PAGE_LINK)%>">
  <%=UrlUtils.htmlFormHiddenFields(pRequest, UrlUtils.PAGE_LINK, formName)%>
  <%=UrlUtils.emitHiddenField(HttpPortletRendererUtil.portletParameter(request,"next_page"),
  "htdocs/portlet2/third.jsp")%>
  <table>
  ……
  <td><input type="TEXT"name="<%=qualParamNameNAME%>"value=""/></td>
  ……
  <td><input type="text" name="<%=qualParamNameADDRESS%>"value=""/></td>
  ……
  </table>
  <input type="submit" name="<%=qualParamNameSubmit%>"value="Submit"/>
</form>
```

通过 portletParameter()将表单中预提交的变量和通过提交表单向下一页面跳转的参数转换成能被 Por-

tal 识别的、唯一的的名字。通过 htmlFormActionLink()方法将 PAGE_LINK 链接转换为用于实现 form 跳转的动作标签。再由 emitHiddenField()方法实现提交 form 时,向下一个页面的跳转。

(4)third.jsp:

```
.....
<td><b>Name:</b></td>
<td><% = paramValueNAME %></td>
.....
<td><b>Address:</b></td>
<td><% = paramValueADDRESS %></td>
.....
<% = UrlUtils.constructHTMLLink(prr, UrlUtils.REFRESH_LINK,"Back to the main page","", linkParams, true, true)%>
```

通过和 second.jsp 中同名变量的定义,实现读取 second.jsp 页面表单提交的内容。再由 constructHTMLLink()向 first.jsp 页面跳转。

3.4　基于 Struts 的 Portlet 的开发

直接利用 PDK API 进行 Portlet 的开发存在开发效率低、业务逻辑不能复用等问题。PDK 从 9.0.4.0.2 版本开始就支持利用 Struts 来开发多页面的 Web provider portlets,因此在开发 Portlet 应用组件中引入 MVC 模式,可以给应用程序提供一个清晰的、有标准组件的视图,同时还允许开发者通过一系列简单、不同的组件来设计应用。

在 Struts 中,实现事件控制流有两个很重要的文件[4]:Web.xml 和 struts-config.xml。前者用于初始化 Web 应用程序的资源,如程序用到的 servlets、标签库 taglibs 等;后者用于初始化 Struts 自身的资源,如 ActionForms,ActionMappings,ActionForwards 等。

基于 Struts 应用的 Portlet 是通过 PPE 而不是浏览器来调用的。当产生对一个 portlet show page 的调用时,页面引擎给 Struts portlet renderer 发送一个 request,然后 Struts portlet renderer 将 request 传到 Struts 控制器 servlet。此功能是通过继承 PDK API 中 oracle.portal.provider.v2.render.http 包中的 StrutsRenderer 类来实现的。对于一个 Struts servlet 应用,StrutsRenderer[5]就相当于一个 adapter,StrutsRenderer 接受 PPE 传送来得 Portlet 的请求,然后再作为代理将请求转交到 Struts Action Servlet。

和单纯的 Struts 应用相比,新的 ActionMappings 中仅仅是路径属性被改动,FormBean action 负责对应用的业务逻辑维持不变。需要注意的就是新的 JSP 页面必须使用 pdk-struts-html 标签,这样才能保证用户在同一个 Portlet 容器中提交请求时包含了 Portal 页面的上下文。

如,<pdk-html:form action="update.do">、<pdk-html:text property="address"/>。

最后需要说明的是对配置文件 provider.xml 需要作以下修改:

(1)配置 struts renderer 在 Struts 上下文中存储 struts action:<actionInSession>true</actionInSession>,使得用户离开 Portal 页面后总能返回到同一个 Portlet 状态。

(2)当调用 Portlet 时,对第一个 action 要详细说明:

```
<showPage class="oracle.portal.provider.v2.render.http.StrutsRenderer">
<defaultAction>/htdocs/strutsportlet/main.jsp</defaultAction></showPage>
```

4　结　语

通过利用 Oracle AS Portal 提供的 API,可以定制满足特定业务逻辑的 Java Portlets。介绍了在门户下开发 Java Portlets 的两种方法:一个是利用最基本的 API,另一个是基于 Struts 的 Portlet。利用 API 实现 Java Portlets 还有许多其他的特性,如添加 JNDI 变量、设置安全特性以及实现单点登录(SSO)等,这些将在下一步的工作中继续探讨和研究。

参考文献:

[1] 范迪维尔,考克斯.Oracle9i Application Server Portal 手册[M].尹志军等译.北京:机械工业出版社,2002:356-357.

[2] Carter J,Grall T. Oracle Application Server Portal Developer's Guide 10g release2(10.1.2)[EB/OL].2005-07. http://huihoo.com/oracle/docs/B14099_19/portal.1012/b14134.pdf.

[3] PDK-Java XML Provider Definition Tag Reference[EB/OL].2003. http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html.

[4] 黄丹霞,杨　璐,崔永普.在 J2EE 项目中使用 Struts 对 MVC 模式的研究与实现[J].计算机工程与设计,2005,26(9):2488-2490.

[5] Oracle Application Server PDK for Java 9.0.4.0.2[EB/OL].2003. http://www.oracle.com/technology/products/ias/portal/html/javadoc/apidoc/index.html.