一种基于MVC模式的Portlet开发框架的设计与实现

许 畅 杨 燕 王 帅 魏 峻
(中国科学院软件研究所软件工程技术中心 北京 100190)

摘 要 为了解决Portlet应用提供商、门户提供商之间互不协调的问题,JCP组织发布了Portlet规范来提供不同门户和Portlet之间的互操作性。JSR规范中定义的繁杂的编程接口和Portlet运行时上下文给Portlet开发带来了特殊性和复杂性。引入一种基于MVC思想的Portlet开发框架OPDS,其提供了简洁的编程接口和配置方式,利用此框架进行开发将大大简化开发的复杂性,同时可加强Portlet应用中业务逻辑资源的复用性。

关键词 门户,Portlet应用,模型-视图-控制器,开发框架
中图法分类号 TP301 **文献标识码** A

Design and Implementation of an MVC-based Framework for Developing Portlet

XU Chang YANG Yan WANG Shuai WEI Jun
(Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

Abstract In order to solve the incompatibility between different portlet application providers and portal providers, JCP issues the Java Portlet Specification to specify how portlet should be developed to be integrated into portals. The complex programming interface and the Portlet Context defined in JSR make it complicated to develop a portlet application. We proposed an MVC-based framework OPDS with simpler API and easier deployment to make the process more efficient and enhance the reusability of resources in portlet applications.

Keywords Portal, Portlet application, MVC, Framework

1 引言

Portlet组件是一个基于Java技术的、可插拔的Web组件,它由Portlet容器管理,执行容器转来的Portlet请求并生成一段页面片段。通常,在一张门户页面上会包含很多个Portlet组件,所有这些组件生成的页面片段组装在一起就形成了用户所看到的门户页面。如何快速高效地完成Portlet应用的开发构建是企业门户应用系统开发者关注的问题。

JCP将Portlet定义为一个新的组件,Portlet规范[1]定义了整合不同信息数据来源、底层应用的Portlet API和上下文环境,描述了Portlet与Portal之间的相互作用、Portlet的生命周期及一些安全性、用户定制、Portlet显示管理等方面的信息。开发人员利用这些新的接口进行开发时,需要投入大量的学习时间,而得到的系统中由于业务处理逻辑与对应视图渲染逻辑以及基于Portlet API的流程控制的相互混杂,使得系统难以维护与更新。

为解决上述问题,本文提出的解决方案是设计一种基于MVC思想的开发框架OPDS(Once Portlet Development Studio),其将与Portlet接口定义和Portlet运行时上下文相关的流程控制部分交给框架中的控制器组件来完成,使这部分代码达到"写一次、到处可运行",且对开发人员透明,大大降低了系统的冗余性和开发复杂性;模型层和视图层提供简洁易用的API和标签库,使作为模型层的业务逻辑资源和视图层的渲染逻辑完全被剥离开来,以便其方便地被移植和更新。

本文第2节分析了OPDS框架的设计原则和需要解决的问题;第3节提出了系统的整体设计;第4节就实现过程中的关键技术作了深入的研究;第5节验证了系统的实现效果;第6节将OPDS与相关研究进行了比较;最后得出结论并指出进一步的工作方向。

2 设计原则和问题分析

在传统的开发方法中,Portlet开发者必须根据组件定义的固定接口进行编程,为实现接口定义的方法将开发大量的、冗余的流程控制代码;同时,由于业务逻辑部分需要与Portlet组件的Request等对象交互来获取参数信息,导致这部分代码与Portlet API紧密耦合无法被复用,因此我们需要一种新的开发方法。我们的设计原则如下:

1) 流程控制对开发者透明,开发者只需专注于业务逻辑的开发以及如何布局一个好的用户界面,而不需要知道Portlet请求是如何被处理和转发的。

到稿日期:2011-09-23 返修日期:2012-02-18 本文受国家973重点基础研究发展规划项目(2009CB320704)和国家科技支撑计划(2011BAH15B03)资助。

许 畅(1988-),男,主要研究方向为网络分布计算、软件工程,E-mail:xuchang09@otcaix.iscas.ac.cn;杨 燕(1976-),女,助理研究员,主要研究方向为门户中间件;王 帅(1982-),男,助理研究员,主要研究方向为网络分布式计算、软件工程;魏 峻(1970-),男,博士,研究员,博士生导师,CCF高级会员,主要研究方向为网络分布计算、软件工程。

2) 业务逻辑可复用，作为模型层的业务逻辑是独立于视图的[2]，所以开发者可以通过配置文件的方式把一个模型对应于不同的视图，或移植到其它应用中。

使用 OPDS 进行开发的系统总体结构如图 1 所示。

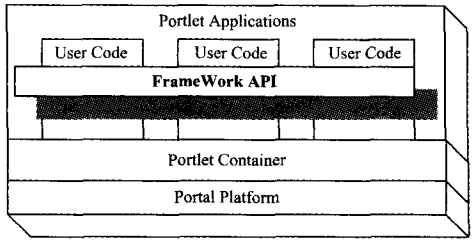

图 1　系统架构图

从图 1 可以看出，OPDS 框架组件作为用户代码与 Portlet 容器之间的桥梁，提供简洁易用可扩展的 API，且不依赖于任何第三方运行平台的支持，从而利用此框架开发出来的应用可以部署在所有支持 JSR 的门户系统中。

基于以上设计原则，我们重点需要解决以下几个问题：

1) 框架 API 与 Portlet 容器的通讯机制：Portlet 组件由 Portlet 容器管理，接收容器传递的请求并进行处理。由于开发者不必知道请求处理的过程，因此框架 API 必须拦截所有 Portlet URL 的请求，解析输入参数与控制参数，将它们映射为业务逻辑单元可执行的操作，并根据逻辑层的处理结果选择恰当的视图以更新，并将结果传回给 Portlet 容器。

2) 业务逻辑单元的管理：由于业务逻辑代码需要能够被复用和移植，因此不能直接让开发者在程序中用 new 主动获取和构造对象，而是让开发者通过配置文件的方式来描述创建对象的方式[3]。在开发者的代码中不直接构造对象的实例，但在配置文件中需描述对象实例化时需要的信息，并由框架中的组件管理对象的生命周期以及对象之间的相互关系。

3　系统设计

本系统基于 MVC 模式架构设计，分为三层实现：控制层、模型层、视图层，如图 2 所示。

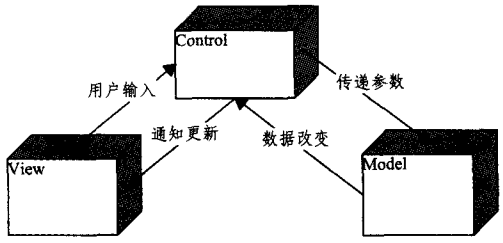

图 2　MVC 结构模型

控制层(Control)是整个框架的联系纽带。它拦截所有 Portlet URL 的请求，解析输入参数与控制参数，即将 Portlet 请求、Portlet 会话等对象中的参数值以键值对的形式存入当前线程的容器中。因此，这个容器包含了模型层和视图层所需的所有运行时数据，具有按名称检索的接口，每个线程对应一个该容器的副本，在线程存在的过程中，可以随意访问属于本线程的副本，这就保证了多线程访问下的安全性。另外，控制器将解析配置文件，对业务逻辑对象进行生成、维护，并处理对象之间的依赖关系，建立模型和视图层的映射规则，具体实现算法将在后续讨论。

模型层(Model)提供给用户一个简单的编程接口，开发者实现此接口并编写普通 Java 类(POJO)[4]来进行各种业务实体数据的操作。在这些 POJO 类中，所需处理的输入数据以类成员变量的形式存在，提供 get 和 set 方法，控制层在请求该对象的实例时会对数据进行赋值。模型层对数据进行改变后，告知控制器有数据发生了变化。最后，模型层的函数被要求返回一个字符串类型的值，控制器将以此字符串作为键值，根据映射规则，选出对应的视图予以更新。一个模型层对象的单个实例可以被多个视图所共享，同一对象的不同实例可以被不同 Portlet 应用所使用，从而实现了模型层对象的可重用性[5]。

视图层(View)提供自定义标签库，主要完成以下两个功能：

1) 处理表单请求，封装 Portlet URL，避免开发者直接与 Portlet URL 交互。开发者并不知道表单中的数据实际被提交到了什么地方，只需在表单中填写要请求的业务逻辑单元的 ID，视图层会包装所有信息发给控制层处理。

2) 分离控制逻辑与渲染逻辑：在传统开发中，需要在页面中插入 Java 脚本来完成条件判断控制(如 if-else)、回圈控制(如 for，while)等，导致代码难以阅读、维护成本高、美工人员难以参与开发。使用了自定义标签，这种标签具有 HTML 标签类似的语法，实际上由标签处理类提供支持，可以封装复杂的功能。

以上从宏观上给出了解决问题的方案和 OPDS 框架的分层设计，接下来将详细地讨论 OPDS 框架 API 与 Portlet 容器的通讯机制、业务逻辑单元的管理等系统实现中的关键技术问题。

4　系统实现关键技术研究

1) 框架 API 与 Portlet 容器的通讯机制

根据 JSR 的定义，一次 render 请求的全过程[6]为：客户端通过 PortletURL 访问门户，门户解析 URL 后向 Portlet 容器发出请求，Portlet 容器根据参数信息和请求属性请求对应的 Portlet 组件，Portlet 在 doView() 方法中进行实际的业务逻辑处理，并将响应内容写入 response 输出流中，响应内容被逐级返回，最终返回给客户端浏览器。

现在，这个通讯过程必须交由框架来处理，开发者并不知道 Portlet 容器和 PortletURL 的存在。在 OPDS 组件初始化阶段，控制器会获取 PortletURL 的一份副本，以及请求分发器并保存起来；视图层封装了用户表单中的所有数据后向控制器发出询问，获取到保存的 PortletURL 副本，将用户请求的业务逻辑单元标识符以及其它控制信息拼接到 PortletURL 中形成新的请求链接，并进行页面跳转。控制器以配置代理类的方式，拦截了所有的请求，这个代理类 DeleagatePortlet 是 Portlet 的实现类(见图 3)，实现了 doView()，processAction() 等 Portlet API 中的固定接口来处理 Portlet 容器的请求并做出响应。它将根据新拼接 URL 中的业务逻辑单元标识符选择业务逻辑对象进行调用，数据处理完后同样由控制器通知对应视图进行更新。最后，控制器调用初始化时保存的请求分发器将视图层更新结果返回给 Portlet 容器

作为响应内容。

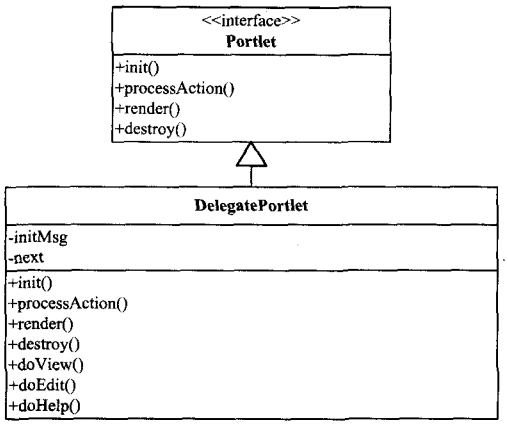

图 3　代理类 DelegatePortlet

整个通讯流程如图 4 所示，从图中可以看出，开发者完全不知道请求是如何被提交、如何被 Portlet 容器处理以及如何被返回的。从开发者的角度来看，数据就像是被直接提交给了所指定的业务逻辑单元对象，处理完后便形成了门户中的页面。因此开发者可以专注于业务逻辑和页面美化两部分的开发，其余部分交给 OPDS 框架中的各个组件来处理，这大大简化了开发过程，提高了开发效率。

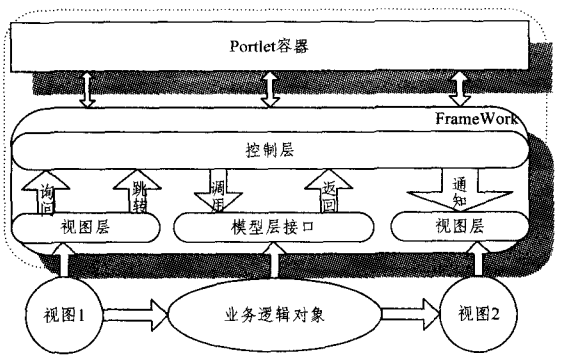

图 4　通讯过程的封装

2）业务逻辑单元的管理

之前已经说过，业务逻辑对象的成员变量可能为基本类型参数（int、String 等），但也有可能为容器中的其它对象，如一个获取天气预报的模型层对象需要通过一个访问数据库的业务逻辑对象来读或写数据的缓存，则必须将访问数据库的对象引入自身的类定义中作为成员变量。

定义 1（依赖对象）　在组件容器中，如果对象 B 的一个实例是对象 A 的成员变量，且对象 A 的实例化必须依赖于对象 B 的实例化，则称对象 B 是对象 A 的依赖对象，或 A 依赖 B。

基于 IOC（Inversion of Control）的思想，对业务逻辑对象进行管理时，对象的生命周期管理权归于组件容器而不是使用这些对象的用户代码；不是直接让开发者在程序中用 new 主动获取和构造对象，而是让开发者通过配置文件的方式描述创建对象的方式，从而降低代码的紧耦合性，增强业务逻辑单元的可重用性。因此在处理依赖对象时，开发者的代码中不直接让对象与它们的依赖对象连接，但在配置文件中描述哪一个组件需要哪些依赖的组件，由框架中的组件容器负责将这些组件联系在一起。

定义 2（依赖深度）　如果对象 A 没有任何依赖对象，则对象 A 的依赖深度 $h(A)=0$；否则，设对象 B 为 A 的所有依赖对象中依赖深度最大的对象，$h(A)=h(B)+1$。

由定义看出，相互依赖的对象将构成树形结构（见图 5），越靠近根结点的对象，依赖深度越大。对于依赖深度为 0 的对象，成员变量全部为基本类型参数，我们只需从容器中查找参数的值，并通过 set 方法将其注入即可。而对于依赖深度不为 0 的对象，必须等到其子树上所有对象实例化后它本身才能开始实例化。因此我们设计如下递归算法 GetBean。

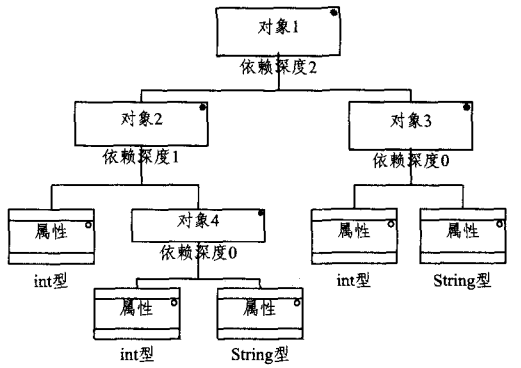

图 5　依赖对象组成的树形结构

算法　GetBean(A)

输入：要创建 Bean 的名称 A

输出：对应于名称 A 的 Bean 对象

```
1. m:= the number of member-variables of A
2. T[1,m]:=the array of variable definition
3.   Initialization:Call the constructor to generate a new Object:bean,
     with all properties of value null
4.   for i:=1 to m do
5.   {
6.     if(isBasicProperty(T[i]. type)) //basic properties
7.     {
8.       invokeSetMethod(bean,T[i]. name,value).
9.     }  //end if
10.    else if(isObject(T[i]. type)) //an object
11.    {
12.      obj= GetBean(T[i]. Name) //Call recursively
13.      invokeSetMethod(bean,T[i]. name,obj).
14.  } //end else
15. } // end loop
16. return bean;
```

5　系统验证

本文以一个标准的表单提交处理应用为例，说明使用 OPDS 框架开发 Portlet 应用的过程，并在中国科学院软件研究所软件工程技术中心门户中间件 OncePortal 3.0[7] 上验证其应用效果。

1）修改 portlet. xml，指定 portlet-class 为控制器中的代理类：

```
〈portlet-class〉
    com. onceportal. controller. DelegatePortlet
```

```
</portlet-class>
```

2）编写视图层登录界面，引入自定义标签进行提交表单：

```
<xc:form name="myform" action="HelloAction" >
  <input id="username" type="text" />
    <input id="password" " type=" password " />
</xc:form>
```

3）编写业务逻辑组件用来验证用户名密码：

```
if(ValidateSuccess(username,password))
{
  ActionContext act= ActionContext. getContext();
  act. setValue("username",username);
  return "success";
}
```

4）编写处理成功后的显示界面 success. jsp，引用自定义标签显示前面输入的用户名和欢迎信息：

```
<div>
  <xc:property value="username" />
</div>
```

5）完善配置信息，将视图层业务组件与模型层关联起来：

```
<bean id="HelloAction" >
  <forward>
    <result name="success" value="success. jsp">
      </result>
  </forward>
</bean>
```

6）将 Web 应用打包部署到所在的 Web 服务器下，此应用在 OncePortal 中的运行效果如图 6 所示。

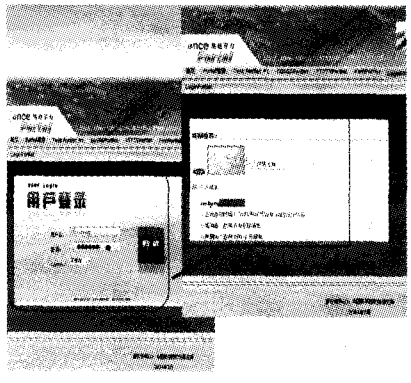

图 6　在 OncePortal 上基于 OPDS 的表单应用

6　相关工作

目前已经有一些基于 MVC 模式的 Portlet 开发框架，如 IBM WebSphere Studio Application Developer 中的 Faces Portlet[8] 框架等。本文所设计的 OPDS 开发框架与上述系统的最大区别是，其不需要依赖任何第三方工具库或软件包，接口更加简单易用，可以方便地在任何支持 JSR 的门户系统中使用。

另外，各大厂商都提供了帮助用户获得“一站式”开发体验[9]的开发工具，如 JBoss Portlet Plug-in，Sun Java Studio Creator 等，通过编辑和定制帮助用户生成 doView()等函数代码[10]。与上述系统相比，OPDS 不仅仅在开发过程、页面布局上提供支持，而且通过屏蔽底层 API 提供给开发者更加简单易用的接口，更大程度上减少了开发者所需投入的学习时间，同时使得到的系统更易维护（见表 1）。

表 1　OPDS 与 IBM Faces Portlet 的对比

	Faces Portlet	OPDS
支持行业标准 Portlet API	是	是
依赖的工具或软件	IBM Portal Toolkit IBM WSAD	无
视图层实现方式	基于 JSF 控件	自定义标签库
支持的部署方式	WAR 包	WAR 包或直接拷贝目录结构

结束语　JSR 中定义的繁杂的接口和 Portlet 运行时上下文给 Portlet 开发带来了特殊性和复杂性。本文引入一种基于 MVC 思想的 Portlet 开发框架 OPDS，其提供了简洁的编程接口和配置方式。全文细致阐述了系统的功能、设计和关键技术，最后通过实例验证了该系统的实用性。下一步工作的重点是提供更丰富的用户界面控件、便捷的事件处理方式，从而进一步降低开发 Portlet 应用的门槛，提高开发效率。

参 考 文 献

[1] Java Community Process. JSR168:Portlet Specification1. 0 [S]. 2003

[2] Sauter P, et al. Extending the MVC design pattern towards a task-oriented development approach for pervasive computing applications[C]// Organic and Pervasive Computing-Arcs 2004. vol. 2981,2004:309-321

[3] Barbaric I. Design patterns in object-oriented ABAP(2nd ed) [M]. Boston,MA:Galileo Press,2010

[4] Hohpe G, Woolf B. Enterprise integration patterns: designing, building,and deploying messaging solutions[M]. Boston: Addison-Wesley,2004:132-135

[5] Gamma E. Design patterns:elements of reusable object-oriented software[M]. Reading,Mass:Addison-Wesley,1995

[6] Rodriguez J R. International Business Machines Corporation. IBM WebSphere Portal V5:a guide for portlet application development[M]. IBM, International Technical Support Organization,2004

[7] 中科院软件所. 网驰平台门户中间件(OncePortal) [EB/OL]. http://www. once. org. cn,2007

[8] 陈隽伟. 基于 Faces Portlet 开发框架构建 Portlet 应用[EB/OL]. http://www. ibm. com/developerworks/cn/websphere/library/techarticles/chenjunwei/fportlet/fportlet2. html? ca = dwcn-newsletter-wsdd

[9] Bellas F,et al. An automatic approach to displaying Web applications as portlets[C]// Distributed Computing and Internet Technology,Proceedings,vol. 4317. 2006:264-277

[10] Diaz O,Paz M. Turning web applications into portlets:Raising the issues[C]//2005 Symposium on Applications and the Internet. Washington,DC:IEEE Computer Society,2005:31-37