J2EE 平台上 MVC 设计模式的研究与实现

陆荣幸，郁 洲，阮永良，王志强
（同济大学 计算机科学与工程系，上海 200331）

摘 要：简要介绍了 MVC 模式思想，讨论了 J2EE 技术，同时以 MVC 模式思想构筑了 J2EE 企业应用平台。

关键词：MVC；JSP；Servlet；EJB
中图法分类号：TP393.092 文献标识码：A 文章编号：1001-3695(2003)03-0144-03

Study and Implementation of MVC Design Pattern on J2EE Platform

LU Rong-xing, YU Zhou, RUAN Yong-liang, WANG Zhi-qiang
(*Dept. of Computer Science & Engineering, Tongji University, Shanghai* 200331, *China*)

Abstract: This paper introduces MVC design pattern briefly, and discusses J2EE technology, and construct J2EE Enterprise Platform based on MVC design pattern.

Key words: MVC; JSP; Servlet; EJB

在对 Web 应用系统进行实际的开发过程中，我们经常会遇到这样的问题：如何进行系统的架构？如何更有利于模块化的设计与编码，今后功能的扩展，以及系统的快速有效的维护？每次的系统开发，我们都需要根据实际应用寻找一个新的解决方案。为了提高开发的效率，考虑到应用系统的灵活性、安全性、实用性等，我们就需要有一种好的设计模式作为一种可行的方案来解决这一常见问题。MVC 设计模式（模块—视图—控制器模式）是一种“分治”的思想，在实现 Web 应用系统中具有得天独厚的优势。本文主要讨论 MVC 设计模式在 J2EE 企业级平台构筑中的研究与实现。

1 MVC 模型介绍

1.1 基本概念

MVC 开发模式是一种“分治”的思想，它将数据的访问和数据的表现进行了分离。通过这种模式，可以开发一个具有伸缩性、便于扩展、便于整个流程维护的平台。

MVC 主要由三个部分组成：模块（Model）、视图（View）和控制器（Controller）。模块，即相关的数据，它是对象的内在属性，是整个模型的核心，它表示的是解决方案空间的真正的逻辑。它采用面向对象的方法，将问题领域中的对象抽象为应用程序对象。在这些抽象的对象中封装了对象的属性和这些对象所隐含的逻辑。视图是模型的外在表现，一个模型可以对应一个或者多个视图。视图具有与外界交互的功能，主管应用系统与外界的接口：一方面它为外界提供输入手段，并触发应用逻辑运行；另一方面，它又将逻辑运行的结果以某种形式显示给外界。控制器是模型与视图的联系纽带，控制器提取通过视图传输进来的外部信息，并将其转化成相应事件，对模型进行更新；同时，模型的更新与修改也将通过控制器来通知视图，从而保持视图与模型的一致性。三者之间的关系如图 1 所示。

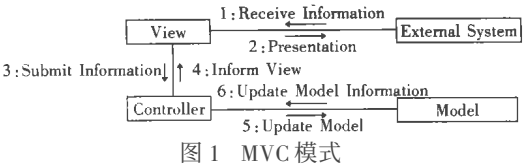

图 1 MVC 模式

1.2 MVC 设计模式的优势

MVC 设计模式具有设计清晰，易于扩展，运用可分布的特点，因此在构建 Web 应用中具有显著的优势。

(1)MVC 模式结构可适用于多用户的、可扩展的、可维护的、具有很高交互性的系统，如电子商务平台、CRM 系统和 ERP 系统等。

(2)MVC 可以很好的表达用户与系统的交互模式，以及整个系统的程序架构模式。

(3)MVC 模式可以很方便的用多个视图来显示多套数据，从而可以使系统能方便的支持其它新的客户端类型。除了运行桌面型的浏览器外，还可以运行在 PDA，WAP 浏览器上。

(4)对于开发人员来讲，由于 MVC 分离了模式中的数据的控制和数据表现，从而可以分清开发者的责任，后台开发人员可以专注业务的处理，前台开发人员专注于用户交互的界面，从而加快产品开发以及推向市场的时间。

2 J2EE 技术

SUN 公司的 J2EE（Java2 企业版）是一种利用 Java 语

收稿日期：2002-03-10

言的标准体系结构定义的,它提供中间层集成框架用来满足高可用性、高可靠性以及可扩展性的应用的需求。利用它,各公司可以更为方便地在中间层加速分布式部署。开发中利用这种体系结构,开发者可以不必担心运行关键商务应用所需的"管道工程",从而可以集中精力重视商业逻辑的设计和应用的表示。

J2EE 技术主要有:EJB, Servlets, JSP, JNDI 等。J2EE 平台的应用主要由构件构成,应用系统的开发就是设计这些构件并组装成整个企业应用。

2.1 JSP

JSP 技术与 ASP 技术类似,是一种在服务器端进行解析,动态生成网页传递给客户端 Web 技术;它是基于 Java 技术,将 Java 代码嵌入 HTML 页面中实现的。本质上,JSP 是一种高层的 Servlet。它与其它网页编写脚本有很大的相似性,但是在执行时有所不同。JSP 引擎将它和它所在的 HTML 文件一起合成 Servlet 的代码,然后它的执行就与 Servlet 的一样:先编译成.class 文件,最后由支持 Java 虚拟机的服务器来执行,最后输出结果。在使用 JSP 中如果结合使用 JavaBean 技术,那么对于处理就会更加方便灵活。

2.2 Servlet

Servlets 是 Java 2.0 中新增的一个全新功能,Servlets 是一种采用 Java 技术来实现 CGI 功能的一种技术。Servlet 和 CGI 一样都是运行在 Web 服务器上,用来生成 Web 页面。与传统的 CGI 或其它 CGI 类似替代技术来说,Java Servlets 效率更高,使用更方便,功能更强大,更小巧也更便宜。Java Servlet 提供了如下许多优势:

(1)Servlet 可以和其它资源(文件、数据库、Applet, Java 应用程序等)交互,以生成返回给客户端的响应内容。如果需要,还可以保存请求-响应过程中的信息。

(2)采用 Servlet,服务器可以完全授权对本地资源的访问(如数据库),并且 Servlet 自身将会控制外部用户的访问数量及访问性质。

(3)Servlet 可以是其它服务的客户端程序,例如它们可以用于分布式的应用系统中。

(4)可以从本地硬盘,或者通过网络从远端硬盘激活 Servlet。

(5)Servlet 可被链接(Chain)。一个 Servlet 可以调用另一个或一个系列 Servelt,即成为它的客户端。

(6)采用 Servlet Tag 技术,可以在 HTML 页面中动态调用 Servlet。

(7)Servlet API 与协议无关。它并不对传递它的协议有任何假设。

(8)像所有的 Java 程序一样,Servlet 拥有面向对象 Java 语言的所有优势。

2.3 EJB

EJB(企业 JavaBean 技术)非常类似于微软的 DCOM。它有一个自己要存活、要活动的容器,为了可以让客户进行透明调用,而不必关心位置;它同时又有一个本地和远程接口及一个相关的配置文件。EJB 技术同时支持暂时和持久对象。暂时对象称为会话组件(Session Bean),持久对象称为实体组件(Entity Bean)。EJB 技术一般应用于数据库操作。在开发过程中,一般采用的开发方式是在会话 Bean 内部调用实体 Bean,因为实体 Bean 没有状态但是对数据库亲和,而会话 Bean 中有我们为了控制程序而需要的上下文信息。这样,我们可以结合这两种 Bean 的所有优点,来比较轻松的进行开发。比如在会话 Bean 中用实体 Bean 进行数据库的访问,同时会话 Bean 用来保存客户的上下文信息。EJB 技术的特点有可移植、可重用,平台独立及可扩展。

3 J2EE 平台上 MVC 设计思想的实现

3.1 MVC 系统模型构建

J2EE 技术结合 MVC 设计模式在构建企业级 Web 应用的实现中,JSP 对应于视图,因为整个应用系统主要通过 JSP 来与外界进行交互;Servlet 对应于控制类,作为 JSP 与 EJB 之间的中间枢纽;EJB 和 JavaBean 对应于模块,主要进行数据业务的处理。MVC 设计模式构建的 Web 应用框架如图 2 所示。

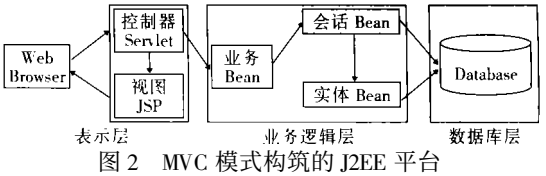

图 2　MVC 模式构筑的 J2EE 平台

MVC 系统模型明确的将数据的显示和数据业务的处理分开,从而使得逻辑结构更为清晰。如果数据的显示方式有所改变,只需更改 JSP 视图页面,而并不要求相应更改数据处理模块;反之,如果业务要求发生变化,也只需更改相应的处理数据模块。因而系统可以很容易加入新的业务,可以灵活适应各种需求的变化。

MVC 设计模式应用于 Web 应用程序,其整个流程如下:当 Web 客户端的 HTML 或 JSP 网页向服务器提交时,服务器端的控制器 Servlet 统一处理这些提交请求。这个控制器 Servlet 根据提交的业务不同,将请求传递给相应的业务 Bean 操作处理,然后将业务 Bean 的处理结果再传递给视图 JSP。视图 JSP 在服务器上处理之后以 HTML 的方式回显给客户端。这里的业务 Bean 是一系统处理业务逻辑的 Java Bean,每个 Java Bean 处理一种业务。会话 Bean 和实体 Bean 都是具体的与数据库的操作,在整个数据模块中供各种业务的 Bean 进行调用。

3.2 MVC 系统模型技术实现

下面就从技术实现的角度上分析 MVC 模型的实现,给出具体的实现类图,同时分析其中的关键技术细节。具体的实现类图如图 3 所示。

MVC 模式中最关键的部分在于控制器的实现。控制器必须能够很好的处理视图与模块的交互。在我们的实现中就是基于这样的一种思想。首先我们将所有的提交功能进行编号,每个编号对应一种功能,每种功能对应一个业务处理类,同时每种功能又对应于一个后继页面。将这种关系存入两个 XML 文件供控制器调用。XML 文件格式如下:

<url-mapping>

```
    <funtion>cmd1</funtion>//功能编号
    <url-pattern>cmd1.class</url-pattern>//功能实现类
</url-mapping>
<url-mapping>
    <funtion>cmd1</funtion>//功能编号
    <url-pattern>/nextpage.jsp</url-pattern>//功能处理的后继页面
</url-mapping>
```

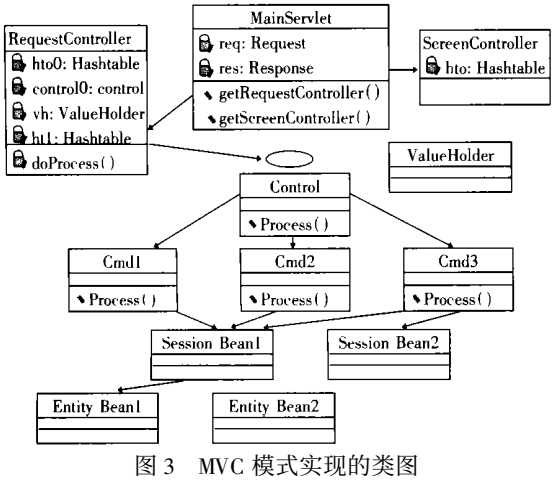

图 3 MVC 模式实现的类图

在控制器进行初始化时，首先读取这两个 XML 文件，然后以 Hashtable 对象的形式将对应关系存入内存中，供后面的各种提交请求进行调用。

控制器的实现主要由三个部分组成：MainServlet，RequestController 和 ScreenController。

MainServlet 主要接受各种业务请求，通过调用 RequestController 进行各种请求的处理，此时控制权属于 MainServlet 中。当 RequestController 处理请求，将结果存入 Session 之后，MainServlet 就将请求与响应交于 ScreenController，释放控制权。

RequestController 功能主要是具体处理各种请求，它在初始化时读取 XML 文件，将功能编号与功能实现类存入一张内存 HASH 表中。当 MainServlet 传来请示 Request 时，RequestController 将请求封装在另一个 Hashtable 对象中，同时从请求中提取功能编号，然后根据功能编号查找初始化时的那张 HASH 表，找出功能实现类。实例化后，将封装的 Hashtable 对象往下处理。主要操作代码如下：

```
control cmd0=null;//command 为业务处理的接口对象
String func=request.getParameter("function");
//从请求中得到功能编号
String processClass=(String)ht0.get(func);
//根据功能编号查找 HASH 表 ht0 得到实现类名称
cmd0=(control)Class.forName(processClass).newInstance();
//实例化一个业务类对象
ValueHolder vh=cmd0.Process(ht1);
//将封装请求的 HASH 表 ht1 作为参数进行请求处理
Session.setAttributes("valueholder",vh);
//将返回对象 vh 存入 session 中，供 JSP 调用
```

ScreenController 功能主要是处理响应，将请求结果与 JSP 页面结合回显给客户端。在初始化过程中，它与 RequestController 一样，读取 XML 文件，将功能编号与功能的后继页面存入一张 HASH 表中。当 MainServlet 将请求与响应传递过来时，它首先从请求中获取功能编号，随后根据功能编号查收 HASH 表，得出后继页面。随后将后继页面（主要是 JSP 页面）结合业务处理类存入 Session 中的结果进行处理，组织成 HTML 的格式回显给客户端。

MVC 模式中的模块部分主要由一系列的业务 JavaBean 和会话 Bean，实体 Bean 组成。为了便于控制器处理，我们对业务 Bean 的实现采取了接口技术。我们定义了接口 Control 及其方法 Process()。Process()的实现在下面具体的业务中实现，即一系列的具体的业务 Bean 实现 Control 接口，采用统一的 Process()来进行处理。

```
public interface control{//各种业务 Bean 的统一接口
    ValueHolder process(Hashtable hb);//统一的接口方法
}
public class cmd1 implements control{//具体的业务 Bean
    public ValueHolder process(Hashtable hb){//具体的业务实现方法
        …
        return vh;
    }
}
```

其中 ValueHolder 对象的作用是存放业务处理的结果，存入 Session 中，供 ScreenController 处理 JSP 时使用。

模块部分的会话的 Bean 与实体 Bean 主要是对数据的具体操作。在设计中根据业务的要求，我们采用的原则是对于数据集和非敏感数据操作采用会话 Bean 直接操作数据库。对于单个数据和敏感数据操作采用会话 Bean 调用实体 Bean，再与数据库进行操作的方式。

MVC 模式中视图部分主要是一系列 HTML 和 JSP 页面，主要功能是进行请求和将模块处理的数据表现给客户端。其中关键的一点就是针对每一个提交都要确定其功能。因此对于表单的提交需要增加一个隐含属性，确定其功能。

```
<input type="hidden"name="function"value="cmd1">
//确实功能是 cmd1
```

根据 MVC 模式，图 4 显示了一个业务的全过程。

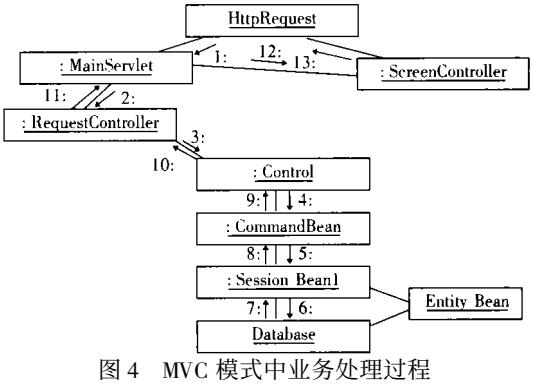

图 4 MVC 模式中业务处理过程

4 结论

MVC 作为一种设计模设在 Web 应用中具有独特的优势，本文主要从实现角度，将 MVC 的思想运用于 J2EE 平台的构筑。在该开发项目中，我们运用了这种设计模式，取得了明显的效果。

参考文献：

[1] McLaughlin B. Java and XML[Z]. O'Reilly&Assoiates, Inc., 2000.
[2] Wendy Boggs. UML with Rational Rose 从入门到精通[M]. 邱仲潘. 北京：电子工业出版社，2000.
[3] Roman E D. Mastering Enterprise JavaBean[Z]. 1999.
[4] Java 2 Platform, Enterprise Edition Specification Version 1.2 [EB/OL]. http://java.sun.com/J2ee/docus.htm.

作者简介：

陆荣幸（1978-），男，硕士生，主要研究方向为分布式系统、网络安全；郁洲，硕士生；阮永良，教授；王志强，学士。