

摘要: MVC (Model, View, Controller)

是 Web 应用开发中的一项重要技术, 可使系统结构清晰, 易于管理和扩展。本文中作者针对具体情况, 运用 MVC 进行合理布局, 设计并实现了一个能同时向手机用户和 PC 机用户提供服务的花卉销售站点。

关键词: MVC JSP Java Servlet
WML JDBC



1 引言

21 世纪将是信息和通信的时代, Internet 网和移动电话网取得突破性的进展, 信息领域中 PC 机和手机迅速普及, 成为极为引人注目的两大增长点。它们正在逐渐地改变着我们原有的商务模式, 通过 Internet 网进行电子商务, 通过移动电话网进行移动电子商务。

目前的主要方案是: 手机上运行 WML 文件, 遵循 WAP 协议, 与 WAP 网关进行交互, 而 WAP 网关将 WAP 协议转换成 HTTP 协议再与 Internet 网进行交互, 然后再将 HTTP 协议转换成 WAP 协议, 把从 Internet 网上获得的信息传回给对应的手机。这里 WAP 网关的主要功能是: 协议转换, 网关对于手机用户是透明的, 通过 WAP 网关, 在手机上输入 Internet 网地址就可以访问 Internet 网上的 WML 文件。我们可以在同一网站开发用于 PC 机显示的 HTML 文件和用于手机显示的 WML 文件。

随着动态网页技术的发展, JSP 与其多方面优势正逐渐成为 Internet 网上的主流开发工具。JSP 文件能动态地生成 HTML 文件和 WML 文件, 来实现这双重应用。但是由于以下几点: 第一,

两种用户不同的显示要求, 不同的文件格式。第二, 电子商务的交易和安全等技术涉及到一系列复杂的商务逻辑。第三, 电子商务网站的信息刷新率要求高, 访问的人数多。这些原因, 使得页面间的导向凌乱, 显示逻辑和商务逻辑混合, 系统效率低, 难以管理和维护。因此, 合理采用 MVC 三层结构来进行开发已成为必要。

2 模型、视图、控制器模式

Model 是整个商务应用的模型, 实际应用中所涉及到的有关类、事务、数据结构以及商务逻辑等。定义 Model 要对软件流程中的对象进行合理抽象, 封装对象的属性和对象的方法。

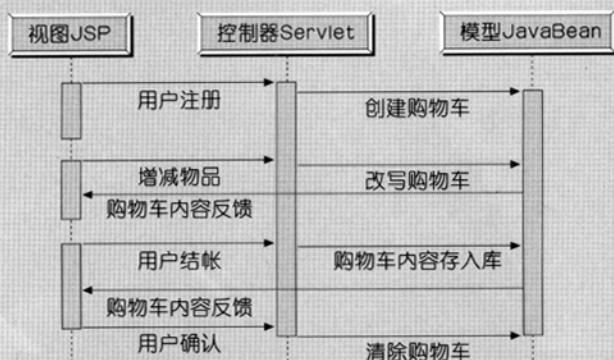
View 是用户视图, 它访问 Model, 将相应 Model 中的信息、商务数据等显示给用户, 同时接收用户的请求和信息的输入, 然后触发 Controller 运行。

Controller 是流程控制器, 从 View 获得用户的请求信息, 转换为相应商务逻辑的动作并执行。根据用户交互信息调用 Model 的方法, 改写 Model 的属性, 使程序流程定向到新的 View 并展现给用户。

采用 MVC 设计模式, 将应用的数据、数据表示和用户动作分离开来, 形成数据层、表示层、事务逻辑层, 各层之间是松耦合的关系, 可以修改每一层的实现而不会影响其他的层, 各层的开发工作可以同步进行, 使开发人员分工明确, 提高开发效率。

对于模型、视

图1 一次购买活动的用况图



MVC Based on Two Types Browser

图. 控制器三层结构, 以一次购买活动的用况图来说明, 如图 1 所示。

系统是这样实现的, 用户在登录 JSP 页面注册, 注册信息由 JSP 传向 Servlet, 经 Servlet 核实后创建购物车模型 Java Bean, 然后把购买 JSP 页面展现给用户, 用户每次增减商品的操作都由 JSP 传给 Servlet, Servlet 再改写购物车 Java Bean 中的内容, 并把改写后的内容反馈给用户, 最后用户结账, 购物车内容写入数据库永久保存, 清除购物车。

3 一个面向两种浏览器的 MVC 实现

下面就以一个花卉电子商务网站来实现可用于 PC 机和手机两种浏览器的 MVC。根据 MVC 的设计原理, 各层之间是松耦合的关系, 每一层的改变不会影响其他层设计。现在要设计出两种不同的界面 (视图) 来针对不同的浏览器, 所以, 需要有对应于手机浏览器的基于 WML 的 JSP 页面和对应于 PC 机浏览器的基于 HTML 的 JSP 页面。而商务逻辑 (控制器) 和数据模型 (模型) 是共同的, 如图 2 所示。两种 View 使用相同的 Controller 和 Model, 当相同的帐号同时从两种浏览器登录时系统也能够识别。这将能保持事务处理的唯一性, 从而维护整个系统的安全。

3.1 模型 Model 映射数据结构和事物

模型应能合理的映射出数据库中的表、视图及商务逻辑中所涉及到的事物。模型一般由一些 JavaBean 构成, 例如, 对应于用户帐户表 userid (userno 序号, username 用户名, password 密码, useraccount 帐号, money 资金数) 的模型为 userid 类, 其代码如下:

```
import java.io.*;
import java.util.*;
import java.sql.*;

public class userid { // 下面定义属性

    public int usern;
```

```
public String username, useraccount;

    public float money;

    public String password;

    // 下面定义操作属性的方法

    public float getMoney(){return money;} // 获得用户资金数

    public void setMoney(float pri){money=pri;} // 设置用户资金数

    public String getUsername(){return username;} // 获得用户名

    ..... }
```

其中每一属性映射数据库表中的一个字段, 后面我们将会看到带参数的构造函数能方便地利用结果集进行初始化。同理, 对应于花卉信息表的模型为 Score 类。而对应于用户的购物车模型, 则应考虑到每个购物车都有唯一的用户帐号, 同时它又要含有多个花卉信息记录。所以, 购物车模型类 buygood 应继承于 userid 类, 又包含多个 Score 类, 代码如下:

```
.....

public class buygood extends userid{

    public Vector vs; // 增加一个矢量属性

    .....

    public buygood(){ // 构造函数

        vs=new Vector(); // 增加操作矢量属性的方法

    public void delinVs(int index){vs.removeElementAt(index);} // 用于向购物车添加物品

    public void addinVs(Score score1){vs.addElement(score1);} // 用于删除购物车中的物品

    public Vector getVs(){return vs;} // 获得购物车中的所有物品

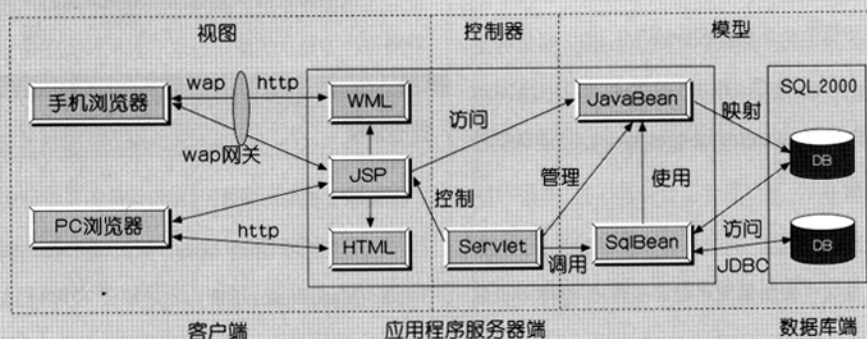
    ..... }
```

可以看出, 类 buygood 除了拥有类 userid 的所有属性和方法外还增加了一个矢量属性 vs 用于装载多个 Score 类的实例。同时添加了操作购物车的一系列方法, 实现对购物车这一事物的抽象。这里通过类的继承体现了软件构件代码重用的思想。

3.2 视图 View 分别针对两种浏览器界面

可使用 FrontPage、WAPtor 等工具先设计出 PC 机的 HTML 文件和手机的 WML 文件, 然后分别在这两种文件中都同样的加入 Java 代码以获取和显示 Model 中的内容, 若需要也可以加入一些控

图 2 MVC 三层结构图



制分页的代码,以查询花卉信息为例代码如下:

```
.....
Vector v=(Vector)session.getAttribute("myv");
// 通过 session 获得矢量
Score score1=(Score)v.elementAt(0);
// 从矢量中取出 Score 类
myname=score1.getName();
// 用 Score 类的方法获取属性值
price=score1.getPrice();
.....
...<%=myname%>...
```

//用<%=...%>形式将属性值内容插入页面

...单价: <%=price%>元...

.....

再把文件的后缀名改为.jsp如: ShowHTML.

jsp, ShowWML.jsp 等。

3.3 控制器 实现商务逻辑、控制程序流程

在本系统中控制器主要解决两个问题。

首先,要对不同浏览器发出的请求进行判断然后决定用 ShowWML.jsp (手机) 还是用 ShowHTML.jsp (PC机) 展现给用户。判断和定向的语句如下:

.....

String destPage="/ShowHTML.jsp";

// 默认目标页面是 PC 机的 ShowHTML.jsp

String accept=request.getHeader("Accept");

// 获取 accept

if(accept.indexOf("text/vnd.wap.wml")>=0) {

// 判断 accept 中的 mime 类型是否为 text/vnd.wap.wml

destPage="/ShowWML.jsp"; } //若是, 则

目标页面改为手机的 ShowWML.jsp

getServletContext().getRequestDispatcher
(destPage).forward(request,response); //跳转到目
标页面

.....

其次,要根据用户的请求调用 SqlBean (数

据库操作类) 完成对数据库的操作,并把结果发给相应用户。控制器 Servlet 主要操作代码如下:

.....

Vector v=SqlBean.executeQuery("SELECT * FROM goodsinfo"); //用矢量 v 来装 SqlBean 类的查
询结果

HttpSession session=request.getSession(); //通过 request 获得 session

session.setAttribute("myv",v); //把矢量 v 放进 session, 准备转向 JSP 页面

.....

要强调一点,这里的结果不是通常所用的结果集,在 SqlBean 类中已经将查询的结果集读出,并封装到相应 Model 类的矢量中,这样做的好处是 SqlBean 在执行完查询后可以及时关闭与数据库的连接,释放资源,不必等到 JSP 页面取出数据后才关闭,在 SqlBean 中执行查询并把结果集读出封装在矢量中的有关代码如下:

.....

conn = connMgr.getConnection("idb"); //通过管理类获得连接池 idb 中的一个连接

Statement stmt = conn.createStatement(); //由连接创建语句对象

ResultSet rs = stmt.executeQuery(sql);

//由语句执行 sql 定义的数据库查询操作,并把结果给 Resultset 对象 rs

userid userid1=new userid(rs); //使用带结果集参数的构造函数来构造用户信息类userid 的一个实例

v.addElement(userid1); //再把这个实例放入矢量中

stmt.close(); //查询语句关闭

connMgr.freeConnection("idb", conn); //连接池管理类及时收回连接池 idb 中的这个连接

return v; //最后将装有查询结果的矢量返回

.....

所以 Servlet 和 JSP 中操作的不是结果集而是矢量,在 3.2 视图部分可以看到 JSP 是通过 session 来获得这个矢量的,然后使用模型中的 Score 类来解析它。

4 总结

实践证明, MVC 技术适用于商务、事务逻辑比较复杂的大中型 Web 系统的开发,能充分满足目前不断发展变化中的电子商务的需要,具有广阔的前景。 ■

参考文献

- 1 邵维忠、杨芙清编著,《面向对象的系统分析》,清华大学出版社,1998.12。
- 2 (美) David M.Geary 编著,贺民译,《advanced JavaServerPages》,科学出版社,2002.9。
- 3 (美) Mark Wutka 编著,程显华译,《JSP 和 Servlet 程序设计使用专辑》,机械工业出版社,2002.3。