第6章　MVC设计模式

主要内容

1. 设计模式
2. JSP的两种开发模型
3. MVC实例

6.1 设计模式

6.1.1 模式简介

- 模式，即pattern，是解决某一类问题的方法论。通过使用模式，可以多次使用已有的解决方案，无需重复相同的工作。
- 设计模式（Design pattern）是一套被反复使用、经过分类编写的代码设计经验的总结。使用设计模式是为了可重用代码、让代码更容易被他人理解、保证代码的可靠性。

6.1 设计模式

- 常用的设计模式有以下几种：层(layers)、管道(pipes and filters)、过滤(blackboard)、代理(broker)、表示-抽象-控制 (Presentation-Abstraction-Control，简称PAC)、微核(microkernel)、映像（reflection）、模型-视图-控制器（Model-View-Controller，简称MVC）。

6.1.2 MVC模式

- MVC是Model-View-Controller的缩写，即模型-视图-控制器，是一种目前广泛流行的软件设计模式。
- MVC把一个应用的输入、处理、输出流程按照模型、视图、控制器的方式进行分离，从而将一个应用程序分成三个核心模块：模型、视图和控制器，它们各自完成不同的任务。简单地说，模型是应用对象，视图是它在屏幕上的表示，控制器定义用户界面对用户输入的响应方式。

6.1 设计模式

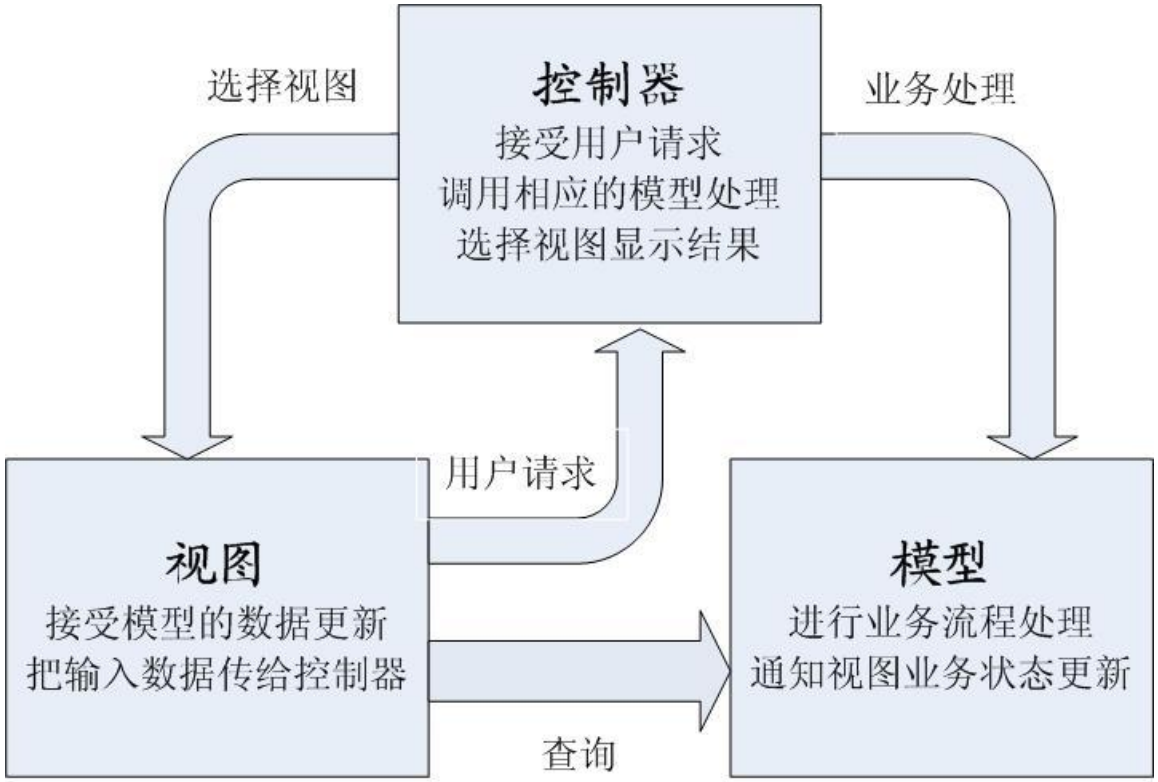


图6-1 MVC模式

1. 视图（View）

视图代表用户交互界面，对于Web应用来说，可以是JSP界面，也可以是HTML、XML和Applet。随着应用的复杂性和规模性的增加，一个应用可能有很多不同的视图，同一个Web应用程序会提供多种用户界面。例如用户希望既能通过浏览器来收发电子邮件，还能通过手机来访问电子邮箱，这就要求Web网站需要同时提供Internet界面和WAP界面。

视图能接受用户的输入数据，但是它并不进行任何数据处理，而是将接受的数据交予模型（Model）处理。视图还能接受模型发出的数据更新事件，从而对用户界面进行同步更新，向用户显示相关的数据。

2. 模型（Model）

模型是应用程序的主体部分，就是业务流程的处理以及业务规则的制定。模型接受视图请求的数据，并返回最终的处理结果。一个模型能为多个视图提供数据，即同一个模型可以被多个视图重用，所以提高了应用的可重用性。

模型的功能一般由JavaBean来实现。

6.1 设计模式

❖JavaBean简介

JavaBean是一种可重复使用、跨平台的组件，它可分为两种：

- 一种是有用户界面（UI）的JavaBean组件
- 另一种是没有用户界面、主要负责处理业务（如数据运算、操纵数据库）的JavaBean。

在JSP中，通常访问的是后一种JavaBean。

6.1 设计模式

一个标准的JavaBean组件具有以下几个特性：

1) 它是一个公开的（pubic）类。

2) 它有一个默认的构造方法，也就是不带参数的构造方法。

3) 它提供setXXX()方法和getXXX()方法来设置和获取JavaBean的属性。

符合上述条件的类，都可以把它看作是JavaBean组件。除了定义setXXX()方法和getXXX()方法外，在JavaBean中也可以像普通Java类那样定义其它完成特定功能的方法。

JavaBean示例

```
package XXX;
import java.XXX;
public class UserBean {
        private String name;
        private int sex;
        private String education;
        public void setName(String name) {
                this.name = name;
        }
        public String getName() {
                return name;
        }

        //这里省略了其它的get和set方法
……
}
```

3. 控制器（Controller）

控制器接受用户的输入并调用模型和视图，将模型与视图匹配在一起，共同完成用户的请求。控制器的功能一般由Servlet实现。

控制器并不做任何的数据处理，而是控制着模型和视图之间的交互过程。例如，用户点击一个连接，控制器接受请求后，并不处理业务信息，它只把用户的信息传递给模型，告诉模型做什么，选择符合要求的视图返回给用户。因此，一个模型可能对应多个视图，一个视图可能对应多个模型。

4. MVC处理过程

控制器接收用户的请求，并决定调用哪个模型来进行处理；然后模型根据用户请求进行相应的业务逻辑处理，并返回数据；最后控制器调用相应的视图来格式化模型返回的数据，并通过视图呈现给用户。

5. MVC的优点

在最初的JSP网页中，像数据库查询语句这样的数据层代码和像HTML这样的表示层代码混在一起。要将数据从表示层分离出来，通常不太容易做到。MVC从根本上强制性地将它们分开，较好的解决了这个问题，但对程序员要求较高，且需要一些额外的工作。

使用MVC架构编写的程序，只需在以前的程序上稍作修改或增加新的类，即可轻松增加许多程序功能。以前开发的许多类可以重用，而程序结构不再需要改变，各类之间相互独立，便于团体开发，提高开发效率。

6. MVC的适用范围

将MVC运用到应用程序中，由于会带来额外的工作量，增加应用的复杂性，所以MVC不适合小型应用程序。对于一些非常小的项目，或者没有后期开发的项目，MVC的分模块设计会给开发带来额外的工作量，可考虑其它方法。

但对于开发存在大量用户界面，并且业务逻辑复杂的大型应用程序，MVC将会使软件在健壮性、代码重用和结构方面显示出优势。尽管在最初构建MVC框架时会花费一定的时间，但从长远角度看，它会大大提高后期软件开发的效率。

主要内容

1. 设计模式
2. JSP的两种开发模型
3. MVC实例

6.2.1 模型一 JSP+JavaBean

在模型一中，JSP页面负责响应用户请求并将处理结果返回用户，所有的数据通过JavaBean来处理。虽然该模型也实现了页面的显示和逻辑处理相分离，但是JSP既要负责业务流程控制，又要负责提供表示层数据，同时充当控制器和视图，未能实现这两个模块之间的独立和分离。所以当需要处理的业务逻辑很复杂时，常常会导致页面被嵌入大量的脚本语言或者Java代码。因此模型一适合小型应用的开发，不适合开发复杂的大型应用程序。

6.2 JSP的两种开发模型

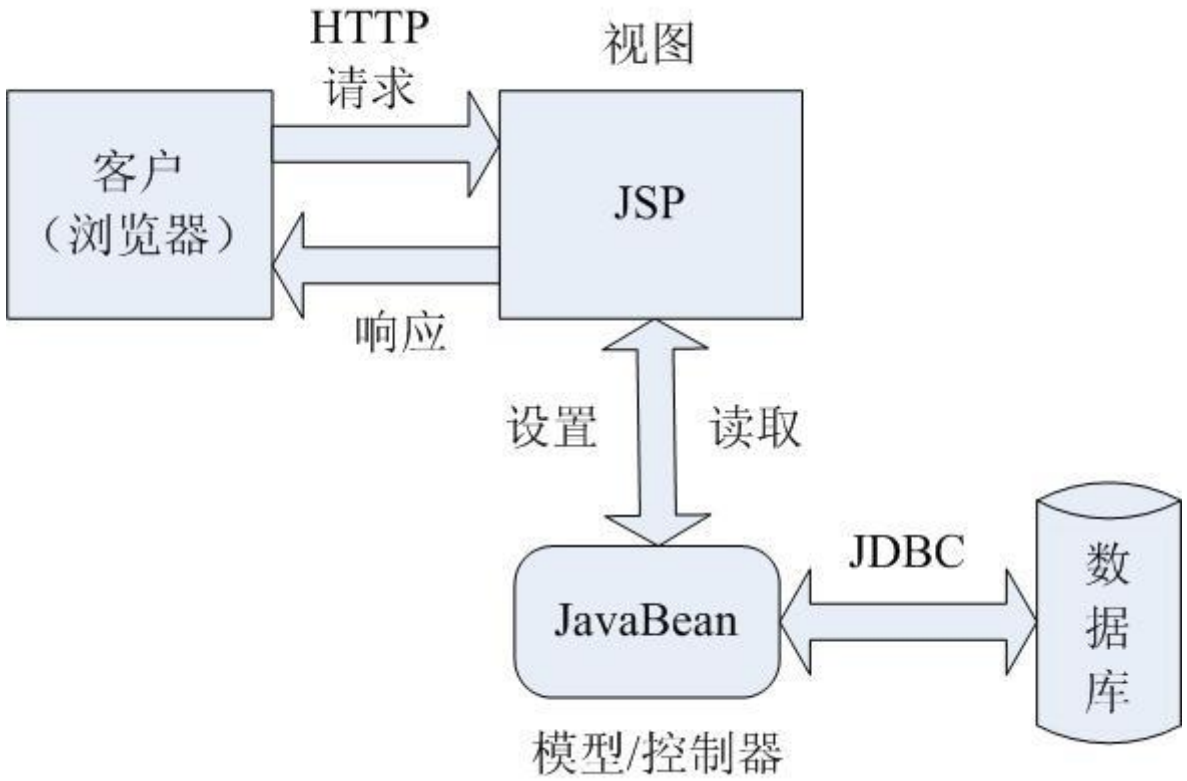


图6-2 JSP模型一体系结构

6.2.2 模型二 JSP+JavaBean+Servlet

模型二体系结构是一种联合使用JSP与Servlet来提供动态内容服务的方法。它的主要思想是使用一个或多个Servlet作为控制器，用JSP生成表示层的内容，让Servlet完成深层次的处理任务，JavaBean作为模型的角色，充当JSP和Servlet通信的工具。在该模型中， Servlet处理完后设置JavaBean的属性，JSP读取此JavaBean的属性，然后进行显示。在JSP页内没有处理逻辑，它只负责检索原先由Servlet创建的JavaBean对象，从Servlet中提取动态内容插入到静态模板。

6.2 JSP的两种开发模型

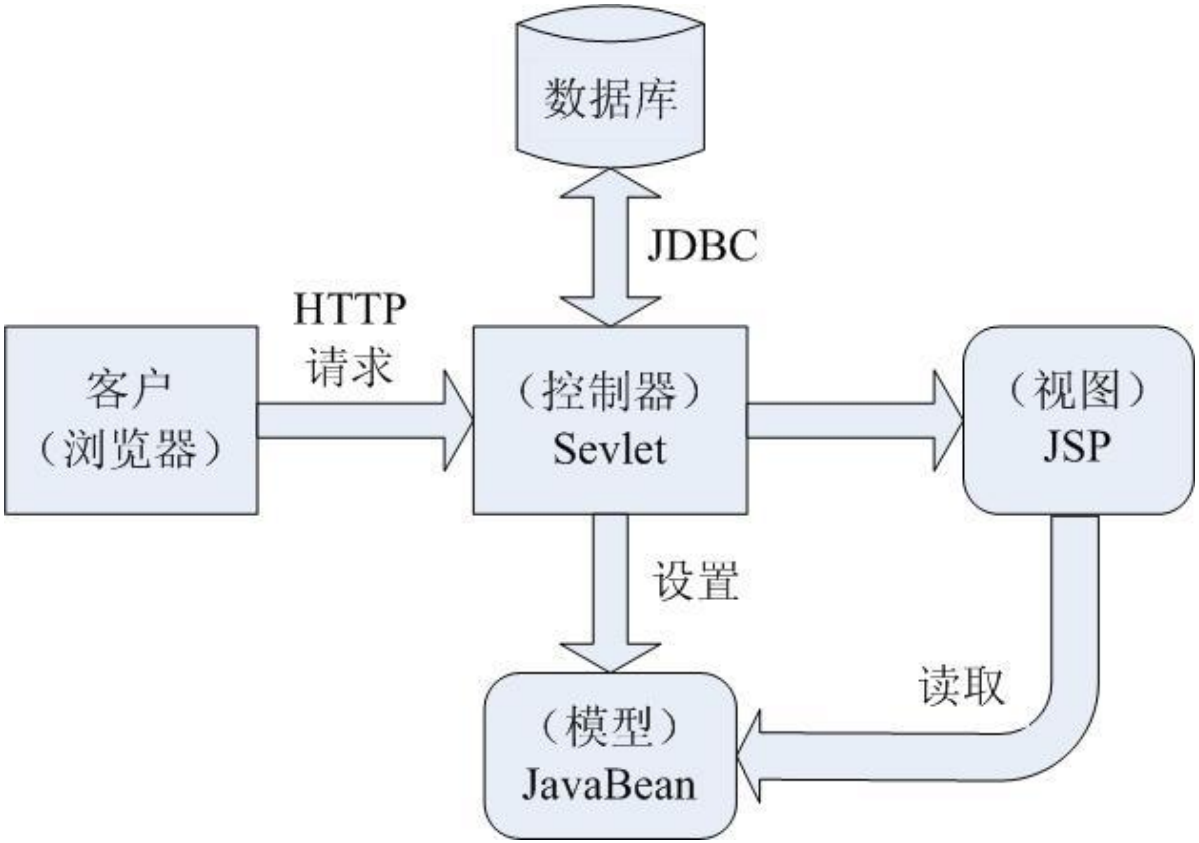


图6-3 JSP模型二体系结构

6.2.3 两种模式的技术特点比较

- 从以上对两种模型的分析来看，模型一和模型二的整体结构都比较清晰，易于实现。它们的基本思想都是实现显示和处理的分开，用JSP页面作为视图，用JavaBean处理业务逻辑。这样的设计便于系统的维护和修改。
- 两种模型的主要区别在于：

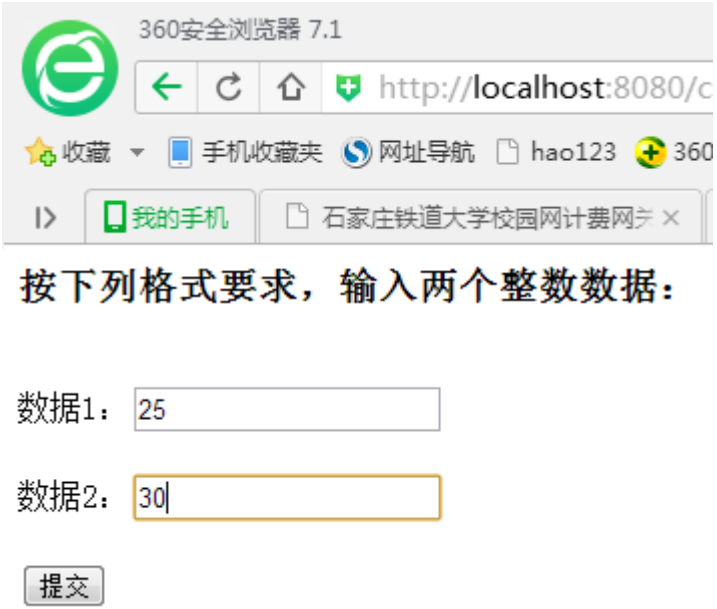
处理用户请求的位置不同，这也是两种模型的最大区别。在模型一中，JSP页面负责处理用户请求，而在模型二中，用户的所有请求提交给控制器Servlet进行处理。

主要内容

1. 设计模式
2. JSP的两种开发模型
3. MVC实例

MVC实例

- 例：利用JSP+Servlet+JavaBean技术，实现任意两个整数之间的所有数的累加值，并显示结果。



6.3 MVC实例

- 使用MVC设计模式开发Web应用一般步骤。

1. 定义JavaBean存储数据或实现业务逻辑功能。

- 在Web应用中通常使用JavaBean对象或实体类存放数据或实现业务逻辑功能，JSP页面在作用域中取出数据。因此，首先需要根据应用处理的实体设计合适的JavaBean。
- 例如：设计JavaBean实现在该项目中“计算累加值”业务逻辑。

6.3 MVC实例—JavaBean（compute）

```
public class compute{
private int a;
private int b;

public compute (int a, int b) {
        super();
        this.a = a;
        this.b = b;
}
public int getA() {
        return a;
}
public void setA(int a) {
        this.a = a;
}
```

6.3 MVC实例—JavaBean（compute）

```
public int getB() {
    return b;
 }
public void setB(int b) {
    this.b = b;
 }
public int sum() {
int c, s = 0;
if (a > b) {
c = a;
a = b;
b = c;
}
int x = a;
while (x <= b) {
s += x;
++x;
 }
return s;
} }
```

2. 使用Servlet处理用户请求

- 在MVC模式中，Servlet充当控制器功能，它从请求中读取请求信息（如表单数据）、创建JavaBean对象、执行业务逻辑，最后将结果转发到视图组件。
- Servlet通常并不直接向客户输出数据。控制器创建JavaBean对象后需要填写该对象的值。可以通过请求参数值或访问数据库得到有关数据。
- 该实例中，Servlet实现由提交页面获取数据，并实施计算和保存计算结果，然后实现跳转到JSP显示页面。

6.3 MVC实例—Servlet（Servlet_chuli）

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Servlet_chuli extends HttpServlet {
    public void doGet(HttpServletRequest request,
HttpServletResponse response)
     throws ServletException, IOException {
request.setCharacterEncoding("GB2312");
String s1=request.getParameter("shuju1");
String s2=request.getParameter("shuju2");
int d1=Integer.parseInt(s1);
int d2=Integer.parseInt(s2);
compute two=new compute(d1,d2);
```

6.3 MVC实例—Servlet（Servlet_chuli）

```
int sum=two.sum();
request.setAttribute("d1", d1);
request.setAttribute("d2", d2);
request.setAttribute("sum",sum);
request.getRequestDispatcher("show.jsp").forward(
request,response);
        }
public void   doPost(HttpServletRequest request,
HttpServletResponse response)
        throws ServletException, IOException {
        doGet(request, response);
        } }
```

3. JSP页面完成信息输入和显示。

6.3 MVC实例—JSP（tijiao.jsp）

```
<%@ page language="java"
pageEncoding="GB2312"%>
<html>
  <head>
    <title>提交任意2个整数给Servlet的页面</title>
  </head>
  <body>
  <h3> 按下列格式要求，输入两个整数数据：</h3><br>
   <form action="Servlet_chuli" method="post">
        数据1：<input name="shuju1"><br><br>
        数据2：<input name="shuju2"><br><br>
        <input type=submit  value="提交">
   </form>
  </body>
</html>
```

6.3 MVC实例—JSP（show.jsp）

```
<%@ page language="java" import="java.util.Date"
pageEncoding="GB2312"%>
<html>
  <head> <title>只显示两数之间累加值的页面</title> </head>
  <body>
    <%
      Integer d1=(Integer)request.getAttribute("d1");
      Integer d2=(Integer)request.getAttribute("d2");
      Integer sum=(Integer)request.getAttribute("sum");
    %>
      <p><%=d1%>加到<%= d2%>的和值是:
      <%=sum%> </p>
</body>
</html>
```

本章小结

MVC模式

两种开发模型：

模型一是JSP+JavaBeans的结合，

模型二是JSP+JavaBeans+Servlet的结合。

习题

1. 什么是MVC设计模式，它有什么优点？

答案：MVC模式称为模型-视图-控制器模式。该模式将Web应用的组件分为模型、视图和控制器，每种组件完成各自的任务。 MVC设计模式的最大优点是将业务逻辑和数据访问从表示层分离出来。

习题

2. 简述实现MVC设计模式的一般步骤。

答案：（1）定义JavaBean表示数据；

（2）使用Servlet处理请求；

（3）填写JavaBean对象数据；

（4）将结果存储在作用域对象中；

（5）将请求转发到JSP页面；

（6）最后在JSP页面中显示数据。