21世纪高等职业教育计算机系列规划教材

Java EE 项目应用开发

基于Struts 2, Spring, Hibernate

刘勇军　王电钢　主　编
孙　璐　罗国涛　副主编

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Java EE 项目应用开发

（基于 Struts 2，Spring，Hibernate）

刘勇军　王电钢　主　编

孙　璐　罗国涛　副主编

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书包括基于 Java EE 核心的 MVC 开发基础；基于 Java EE 开发的核心技术和 Java EE 高级 Web 应用开发专业项目——权限管理系统三部分内容。第一部分介绍基于 B/S 的 Web 应用开发模型、Java EE 体系结构、Java EE 开发环境搭建、基于 MVC 的常用 Web 开发模式、典型的 MVC 框架 Struts，其中通过样例开发的讲解用于掌握 MVC 开发基础，并通过习题和实训应用内容用于强化基本 MVC 开发能力。第二部分系统介绍了 Struts 2 应用、Hibernate 应用、Spring 应用及它们之间的整合应用，对这些核心框架技术及整合应用都提供样例开发实践，可以让读者清晰地了解它们之间的应用方式，并且利用习题和实训应用的训练，可以强化读者对这些核心框架技术的应用能力。第三部分详细地介绍了一个真实工程应用项目——XX 信息管理系统之权限管理子系统的分析、设计、开发实现过程。

本书可作为高职高专相关专业课程教材和教学参考书，也可供从事 Java EE 应用系统开发的用户学习和参考之用。

图书在版编目（CIP）数据

Java EE 项目应用开发：基于 Struts 2，Spring，Hibernate / 刘勇军，王电钢主编. —北京：电子工业出版社，2012.5
（21 世纪高等职业教育计算机系列规划教材）
ISBN 978-7-121-16194-0

Ⅰ. ①J… Ⅱ. ①刘… ②王… Ⅲ. ①JAVA 语言－程序设计－高等职业教育－教材 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2012）第 039550 号

策划编辑：徐建军（xujj@phei.com.cn）
责任编辑：徐建军　　特约编辑：俞凌娣　赵海红
印　　刷：
装　　订：涿州市京南印刷厂
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1 092　1/16　印张：20.25　字数：518.4 千字
印　　次：2012 年 5 月第 1 次印刷
印　　数：3 000 册　　定价：35.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

Java EE 技术从最初提出构建企业级应用，经过多年的发展日臻成熟，目前已经成为电子商务应用最佳解决方案之一，得到行业的广泛认可和应用。为了适应形势发展需要，我国许多高校（特别是专门培养学生实践技能的高职高专院校）开设了 Java EE 课程，并且已经构成了一个系列。但目前讲解 Java EE 的书主要分成两类：一类是纯粹面向企业应用开发的高级应用而编写的，对 Java EE 技术基础讲解不够；另一类是纯粹的 Java EE 基础讲解，而没有把 Java EE 技术应用到真实案例。这两类书籍并不能很好地适应学生学习的需要。

作者学校从 2005 年起开始与 IBM 教育学院合作开设软件技术（Java EE 应用开发方向），从最开始直接使用 IBM 教育学院提供的培训类教程，到后面逐步结合工程应用积累的经验所形成的讲义，经过几年的教学总结和研究思考，我们发现，现有教材+讲义的方式已经不适合专业学生能力的提升，于是决定根据多年教学经验和 Java EE 工程应用实际，编写这本 Java EE 项目应用开发。

Java EE 技术包含很多内容，本书从基于 Java EE 核心的 MVC 开发基础入手，掌握基本的 Web 应用开发技术，接着分别介绍基于 Java EE 开发的核心框架技术 Struts 2、Spring、Hibernate 以及它们的相互整合应用，每一种核心框架技术都从基础的例子着手，一步一步引导读者学习和应用这些技术，并最后通过一个真实工程应用项目提升学生利用 SSH 技术进行工程应用开发实践能力。

本书主要包含基于 Java EE 核心的 MVC 开发基础、基于 Java EE 开发的核心技术、Java EE 高级 Web 应用开发专业项目三部分，涵盖基础、核心技术和工程应用的全过程。基础部分介绍基于 B/S 的 Web 应用开发模型、Java EE 体系结构、Java EE 开发环境搭建、基于 MVC 的常用 Web 开发模式、典型的 MVC 框架 Struts。核心框架技术部分系统介绍 Struts2 应用、Hibernate 应用、Spring 应用及它们之间的整合应用。工程应用部分详细介绍了一个真实工程应用项目——XX 信息管理系统之权限管理子系统，利用 SSH 技术，结合软件开发生命周期，系统地展现了权限管理系统的分析、设计、详细实现的完整过程，从而让读者体会到整合框架的妙处，深刻理解 SSH 框架的详细应用。

本书在核心技术内容选取上以基础、实用、够用为原则，项目实践按照需求分析、项目设计、代码迭代、软件测试及项目部署等实际开发流程编写，注重培养读者的 Java EE 工程应用能力。本书在编写上立足于高职高专，语言浅显易懂，对核心技术说明均采用工程应用项目中实用的程序分析问题。

本书可作为高职高专相关专业课程教材和教学参考书，也可供从事 Java EE 应用系统开发的用户学习和参考。

本书由四川托普信息技术职业学院刘勇军副教授和四川电力职业技术学院王电钢博士担任主编，四川托普信息技术职业学院孙璐、罗国涛担任副主编，陈虹君、黄琨、杜毅等老师参与编写工作，另外四川托普信息技术职业学院软件研究所部分老师和学生帮助完成了项目组织，代码调试等基础工作。其中刘勇军负责编写第三部分中的第 12、13 章，王电钢负责编写第一部分中的第 1 章和第二部分中的第 3 章，孙璐负责编写第二部分中的第 9、10 章，罗国涛负责编写第二部分中的第 2、4 章，陈虹君负责编写第二部分中的第 5、6 章，黄琨负责编写第二部分中的第 7、8 章，杜毅负责编写第二部分中的第 11 章。本书的编写过程是一个不断解决困难的

过程，有时举步维艰，有时进展顺畅。幸好有编写团队朋友们的鼓励和支持，没有大家的不遗余力，兢兢业业的努力，本书不可能成形，在此对所有参与编写工作的同仁表示由衷的感谢。

为了方便教师教学，本书配有电子教学课件，请有此需要的教师登录华信教育资源网（www.hxedu.com.cn）免费注册后进行下载，有问题时请在网站留言板留言或与电子工业出版社联系（E-mail:hxedu@phei.com.cn），也可直接与作者联系（E-mail：slllyj@163.com）

由于编者水平有限和时间仓促，书中难免存在疏漏之处，欢迎广大读者批评指正。

目　录

第一部分　基于 Java EE 核心的 MVC 开发基础

第二部分　基于 Java EE 开发的核心技术

第一部分　基于 Java EE 核心的 MVC 开发基础

第 1 章　Web 应用构架及 Java EE

课程目标

- ◆ 掌握 B/S 结构模式的 Web 应用程序中客户端和服务器端的交互过程。
- ◆ 熟悉 HTTP 协议请求响应模型。
- ◆ 了解 Web 应用发展过程。
- ◆ 了解 Java EE 平台的体系结构、规范、应用程序开发模型及 Java EE 打包部署。
- ◆ 了解 MVC 设计模型，熟悉其在 Java EE 架构中的应用。
- ◆ 熟练基于 Java EE 的 Web 应用开发环境的搭建。
- ◆ 掌握常用的 Web 开发模式，能应用 JSP Model1、JSP Model2 开发模式进行 Java EE 工程项目开发。
- ◆ 掌握典型 MVC 框架 Struts，能应用 Struts 开发 Java EE 工程应用。

今天，人们见到的绝大部分应用，都是基于 B/S（浏览器/服务器）架构的，客户端是浏览器，服务器是 Web 服务器。可见，Web 应用是目前广泛使用的应用模式。

在本章中，首先介绍一些基于 Java EE 核心的 MVC 开发基础知识。包括基于 Web 应用的 B/S 结构模式，HTTP 请求响应模型，Web 应用发展历程，Java EE 体系架构，Java EE 组件和容器，Java EE 打包与部署，MVC 设计模式及其在 Java EE 架构中的应用，Web 开发模式，典型 MVC 框架 Struts，以及部分样例分析。

1.1　Web 应用架构

1.1.1　Web 应用模型

Web 应用是基于 B/S 结构的，也就是浏览器/服务器结构。

基于 B/S 的应用程序部署在服务器端，客户端通过浏览器访问应用程序。如图 1-1 所示。

图 1-1　服务器客户端模型

从图 1-1 中可以看到，客户端发送 HTTP 请求消息传给服务器，服务器将请求传递给服务

器中所部署的 Web 应用程序，Web 应用程序处理请求，并把响应的 HTML 页面通过服务器传给客户端显示。

在应用中，人们常用的浏览器有微软的 IE，网景公司的 Netscape Navigator，Mozilla 的 Firefox，以及目前国内用户常用的 360 浏览器等。

服务器分两种：Web 服务器和 Web 应用服务器。

● Web 服务器

可以解析 HTTP 协议。当 Web 服务器接收到一个 HTTP 请求，会返回一个 HTTP 响应，例如送回一个 HTML 页面。为了处理一个请求，Web 服务器可以响应一个静态页面或图片，进行页面跳转，或者把动态响应的产生委托给一些其他的程序，如 CGI 脚本，JSP 脚本，Servlets，ASP 脚本，服务器端 JavaScript，或者一些其他的服务器端技术。无论它们的目的如何，这些服务器端的程序通常产生一个 HTML 的响应来让浏览器可以浏览。

Web 服务器不支持事务处理或数据库连接池。但它可以配置各种策略来实现容错性和可扩展性，如负载平衡，缓冲。

常用的 Web 服务器有 IBM 的 HTTP Server，微软的 IIS，还有 Apache 的 Web 服务器。本书将采用 Apache Tomcat 的 Web 服务器来部署各工程应用样例。

● Web 应用服务器

可以通过各种协议，包括 HTTP，把商业逻辑暴露给客户端应用程序。Web 服务器主要是处理向浏览器发送 HTML 以供浏览，而 Web 应用服务器提供访问商业逻辑的途径以供客户端应用程序使用。应用程序使用这些商业逻辑就好像是调用对象的一个方法一样。应用程序服务器的客户端（包括有图形用户界面 GUI 的）可能会运行在一台 PC、一个 Web 服务器甚至是其他的应用程序服务器上。

在大多数情形下，Web 应用服务器是通过组件的应用程序接口（API）把商业逻辑暴露给客户端应用程序的，例如基于 Java EE 应用程序服务器的 EJB 组件模型。此外，Web 应用服务器可以管理自己的资源，如安全、事务处理、资源池、消息等，就像 Web 服务器一样，Web 应用服务器配置了多种可扩展和容错技术。

常用的 Web 应用服务器有 IBM 的 WebSphere Application Server，BEA 公司的 WebLogic，Oracle 的 IAS，以及市场使用较为广泛的开源应用服务器 Jboss，等等。

1.1.2 HTTP 请求/响应模型

HTTP 是一个属于应用层的面向对象的协议，由于简洁、快速的方式，适用于分布式超媒体信息系统。

HTTP 协议基于请求/响应模型，因此存在两种 HTTP 消息：请求消息和响应消息。一个完整的 HTTP 会话过程如图 1-2 所示。

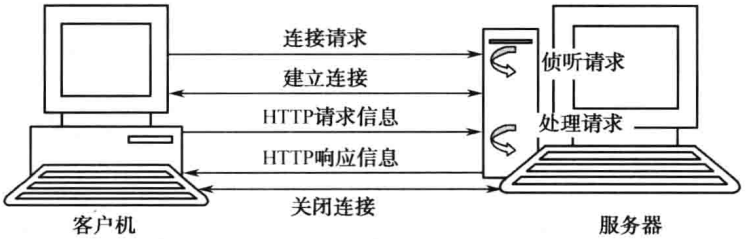

图 1-2 HTTP 请求响应模型

首先，客户端与 Web 服务器建立连接，通常通过默认的 80 端口：服务器 Server 侦听 HTTP 协议端口 80 等待客户端 Browser 的 Socket 连接请求；Browser 向 Server 发出 Socket 连接请求，两边 Socket 建立连接。

建立连接后，客户端向 Web 服务器发送 HTTP 请求消息：Browser 通过 Socket 连接的 OutputStream 向 Server 发送 HTTP 请求消息。

Web 服务器处理请求，并将响应消息传送给客户端：Server 通过 Socket 连接的 InputStream 得到请求，分析处理后通过 Socket 连接的 OutputStream 返回响应消息。

这样一个来回后，这个连接就关闭了（Browser/Server 关闭 Socket 连接）。HTTP 协议默认使用 80 端口进行访问。

HTTP 协议是一个无状态的协议。也就是说，每当客户端访问 Web 服务器上某个 Web 页面时，都要建立与服务器的一个独立的连接。服务器不保留前一次访问的任何信息。Web 服务器将客户端对某个页面的每次访问都当作相互无关的访问来处理；服务器不会自动保留客户机的状态信息。所以 HTTP 协议是一个无状态的协议。正因为如此，服务器端需要采取一定的措施来保留用户的状态数据，这就是状态管理。

请求消息和响应消息格式都包括起始行、零个或多个题头域、一个空行后的消息体。这里空行表示消息题头的结束，消息体是可选的。

请求消息的起始行就是请求行。它通常都是请求消息的首行，包含三个域：HTTP 方法、通用资源标示符（URI）、HTTP 协议版本。

尽管有几种 HTTP 方法可以从服务器中检索数据，但是最常用的通常仅有 GET 和 POST 方法。GET 方法向服务器请求资源，由请求 URI 指示请求地址。如果 URI 指向生成数据的资源，如 Servlet，则数据在响应消息中返回。如果要提交表单数据，可用 POST 方法。尽管 GET 方法可以在查询字符串中传递信息，但是一般使用 POST 方法明确传递服务器数据，这些数据可以被请求 URI 用来进行处理。

URI 标志用来处理请求的资源。它可以是绝对路径，也可以是相对路径。无效的 URI 请求会返回错误代码（通常是 404）。

HTTP 请求协议版本告诉服务器：请求遵循了那个 HTTP 规范版本。

HTTP 请求可能包含零个或多个题头域。请求题头域允许客户端向服务器传递有关请求和客户端本身的一些附加信息。请求消息和响应消息的题头域的格式是相同的，首先是题头域的名称，接着是冒号和值。如果对同一题头域规定了多个值，它们必须用逗号隔开。如表 1-1 是常见的题头域。

表 1-1　HTTP 消息题头域

名称	目的	举例
Accept	响应可以接受的媒体类型。如果没有 Accept 题头域，则服务器可以安全地假设客户端接受所有的媒介类型	accept:image/gif accept:image/jpeg
Accept-Language	客户端希望响应优先使用的语言	Accept-Language:en-us
Content-Type	发送到接受者的消息体的媒介类型	Content-Type:text/html
Host	所请求的资源的主机名称和端口号	Host:172.23.77.93
User-Agent	包含发出请求消息的客户端的信息	User-Agent:Mozilla/4.0

下面来构造一个实例查看请求消息格式：编写一个 Java 应用程序 HttpRequestViewer，该应用程序模拟 Web 服务器监听浏览器发出的 HTTP 请求，截获 HTTP 请求消息并在控制台将

请求消息格式打印出来。详见【代码 1-1】。

【代码 1-1】　HttpRequestViewer.java。

```
/**此示例使用 Java 网络编程技术实现*/
public class HttpRequestViewer{
    public static void main(String args[]){
        try{
            ServerSocket serversocket=new ServerSocket(8080);//服务器套接字,
            绑定 8080 端口
            Socket socket=serversocket.accept();//与客户端建立 Socket 连接
            InputStream is=socket.getInputStream();//从套接字对象中得到输入流对象
            InputStreamReader isr=new InputStreamReader(is);//字节流转换为字符流
            BufferedReader br=new BufferedReader(isr);//包装成缓冲流

            char charry[]=new char[1024];//定义缓冲区
            int charrylength=br.read(charry);//读取内容
            for(int i=0;i<charrylength;i++){//将读取内容循环输出到控制台
                System.out.print(charry[i]);
            }
            br.close(); //关闭流对象
            socket.close(); serversocket.close();//关闭套接字连接
        }catch(Exception e){
            System.out.println("异常发生: "+e.getMessage());
        }
    }
}
```

编译并运行该 Java 应用程序，此时该应用程序将绑定 8080 端口进行监听。打开 IE 浏览器，在地址栏中向该模拟服务器发出资源请求，确保是向 8080 端口请求，例如，输入：http://localhost:8080/testweb/testhello.html，此时 Java 应用程序模拟的 Web 服务器正侦听 8080 端口，当与浏览器建立连接后，程序就会获取浏览器发出的请求消息，并打印在控制台上。可在控制台看到如图 1-3 所标示的请求消息格式。

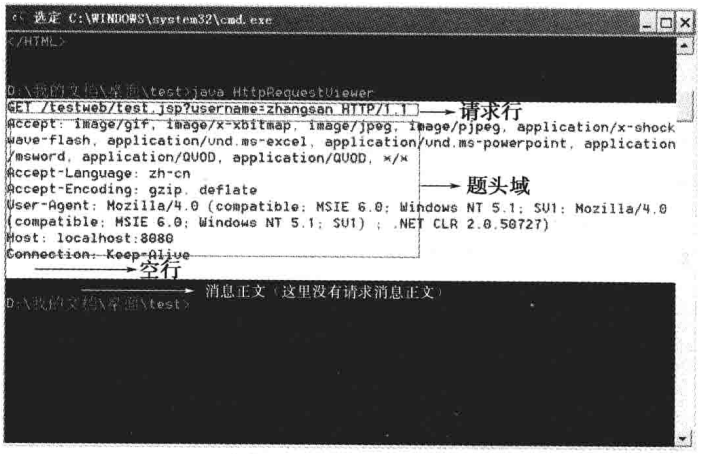

图 1-3　请求消息格式

一旦服务器接收和处理了请求消息，它就必须向客户端返回一条响应消息。响应消息的起

始行称为状态行，此外同样包含零个或多个题头域，空行后一个可选的消息体。

HTTP 响应的状态行包括响应消息所采用的 HTTP 协议版本，之后是响应状态码和状态描述。这些字段之间以空格隔开。状态码是三位数字值，用于描述服务器的响应状态。如表 1-2 所示是常见的 HTTP 响应状态码。

表 1-2　HTTP 响应状态码

响应状态码	文本描述	含义
200	OK	请求成功
400	不良请求	由于语法错误而导致服务器无法理解请求信息
401	未授权	请求要求身份验证和/或授权
404	未发现	服务器未发现与请求 URI 匹配的内容
500	内部服务器错误	服务器出错，无法处理请求

下面同样来构造一个实例查看响应消息格式：编写一个 Java 应用程序 HttpResponseViewer，模拟浏览器查看服务器返回的 HTTP 响应，在控制台将 HTTP 响应消息格式打印出来，详见【代码 1-2】。

【代码 1-2】　HttpResponseViewer.java。

```
public class HttpResponseViewer{
    public static void main(String args[]){
        try{
            Socket socket=new Socket("localhost",8080); //构建套接字向服务
            器发出连接请求
            InputStreamReader isr=new InputStreamReader(socket.
            getInputStream());
            BufferedReader br=new BufferedReader(isr);
            PrintWriter pw=new PrintWriter(socket.getOutputStream());//
            从套接字中得到输出流并包装
            pw.println("GET /index.html HTTP/1.1");//向服务器发送请求消息
            pw.println("HOST:localhost:8080");
            pw.println();
            pw.flush();
            char charry[]=new char[1024*100];
            int charrylength=br.read(charry);//读取从服务器反馈的响应消息
            for(int i=0;i<charrylength;i++){//在控制器台环把响应消息打印出来
                System.out.print(charry[i]);
            }
            br.close(); pw.close();     socket.close();
        }catch(Exception e){
            System.out.println("异常发生: "+e.getMessage());
        }
    }
}
```

编译该 Java 应用程序得到字节码文件，首先启动一个 Web 服务器如 Tomcat6.0，这个 Web 服务器默认会侦听 8080 端口的 HTTP 请求，接着使用 Java 解释器解释执行 Java 应用程序的字节码文件，模拟浏览器向 Web 服务器发出请求，并且接受 Web 服务器发出的响应消息。可以

在控制台看到如图 1-4 所示的响应消息格式。

图 1-4　HTTP 响应消息格式

1.1.3　Web 应用发展

早期的或者最简单的 Web 站点只提供 Web 页面的静态信息访问，如图 1-5 所示。也就是说，Web 服务器所做的工作仅仅是将客户端发来的 HTTP 请求映射到文件系统的某个页面，然后把这个页面，通常就是普通的 HTML 页面传送给客户端。应用程序不会根据用户传入数据的不同而修改 Web 页面。

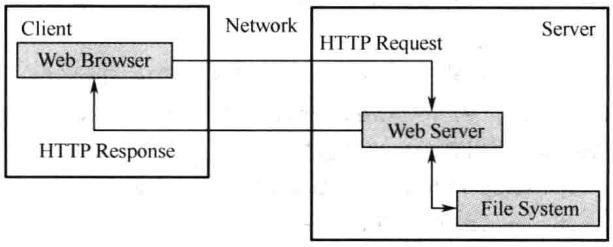

图 1-5　静态 Web

后来出现了 Applet，Applet 被称为客户端的 Java 小应用程序，它必须嵌入到网页中，由浏览器内嵌的 JVM 执行。当用户发出 HTTP 请求后，Applet 随着 Web 页面一起被下载到客户端，Applet 在浏览器的 JVM 中运行，能够提供动态的页面内容，从一定程度上扩展了 Web 服务器的功能，如图 1-6 所示。

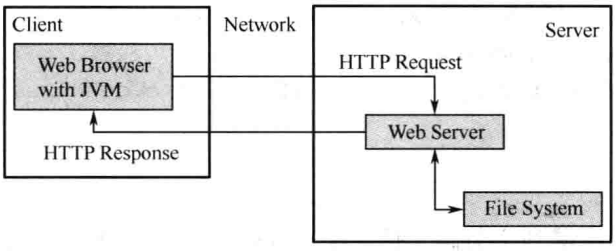

图 1-6　Applet

但是，基于安全性考虑，Applet 被限制不能访问后台数据。这就出现了运行在服务器端的 Servlet，如图 1-7 所示。

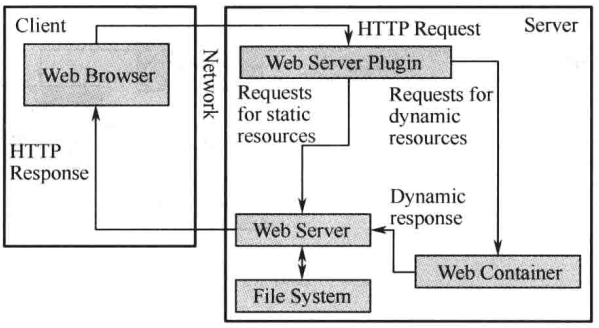

图 1-7　Servlet

Servlet 同样也是一段 Java 程序，称为服务器端 Java 小应用程序，它会根据用户提交的数据不同而产生不同的响应页面，从而提供与用户动态的交互。

Servlet 是运行在服务器上的 Web Container 中，也就是 Web 容器。Web 应用服务器提供 Container，用于管理像 Servlet 这样的服务器端组件。

所以，当一个 HTTP 请求传到服务器，Web Server Plugin 会检测这次请求的是静态资源还是动态资源。如果是静态资源，请求会传递给 Web Server，Web Server 将请求直接映射到文件系统中的某个页面，原封不动地将页面传回给客户端。但如果请求的是动态资源，Web Server Plugin 会将请求传递给 Web Container，Web Container 会调用相应的 Servlet，根据用户提交数据的不同，将动态内容传回给客户端。

由于这些动态内容是由 Servlet 通过硬编码方式输出 HTML 页面，这带来一个问题：就是仅使用 Servlet 往往会把业务逻辑和显示逻辑混合在一起。

为了更好地分离视图、控制和业务逻辑，JSP 技术应运而生。通常用 JSP 页面来显示给用户的数据，用 Servlet 控制页面的流程，如图 1-8 所示。Servlet 会根据请求信息的不同或应用程序逻辑的设定而调用相应的 JSP 页面。JSP 和 Servlet 都可以调用 JavaBean，从而产生动态内容。

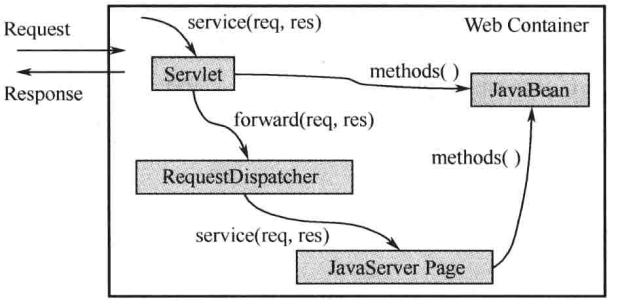

图 1-8　JSP