第1章

Java Web技术概述

本章内容

- 1.1 Internet与万维网
- 1.2 Web常用技术
- 1.3 服务器资源
- 1.4 Tomcat服务器
- 1.5 Servlet与JSP入门
- 1.6 MVC设计模式

1.1 Internet与万维网

- Internet正式中文译名为“因特网”，是一个全球性的、开放的计算机互联网络，Internet联入的计算机覆盖了全球绝大多数的国家和地区，存储了丰富的信息资源，是世界上最大的计算机网络。

1.1.1 主机和IP地址

- 连接到Internet上的所有计算机，从大型机到微型机都是以独立的身份出现，我们称它为主机。
- 为了实现各主机间的通信，每台主机都必须有一个唯一的网络地址，叫做IP（Internet Protocol）地址 。

IP地址

- 目前使用的IP地址是用四个字节32位二进制数表示的，如某计算机的IP地址可表示为10101100 00010000 11111110 00000001。为便于记忆，将它们分为4组，每组一个字节8位，由小数点分开，且将每个字节的二进制用十进制数表示，上述地址可表示为172.16.254.1，这种书写方法叫做点分十进制表示法。用点分开的每个字节的十进制整数数值范围是0~255。

1.1.2 域名和DNS

- 不管用哪种方法表示IP地址，这些数字都很难记住，为了方便人们的记忆，在Internet中经常使用域名来表示主机。域名（domain name）是由一串用点分隔的名字组成的某一台主机或一组主机的名称，用于在数据传输时标识主机的位置。
- 域名系统采用分层结构 ，例如，www.tsinghua.edu.cn是一个域名

DNS

- 由于IP地址是Internet内部使用的地址，因此当Internet主机间进行通信时必须采用IP地址进行寻址，所以当使用域名时必须把域名转换成IP地址。这种转换操作由一个名为“域名服务器”的软件系统来完成，该域名服务器实现了域名系统（Domain Name System，DNS）。

1.1.3　万维网概述

- WWW是World Wide Web的简称，称为万维网，也简称为Web。
- Web是基于客户/服务器（C/S）的一种体系结构，客户在计算机上使用浏览器向Web服务器发出请求，服务器响应客户请求，向客户送回所请求的网页，客户在浏览器窗口上显示网页的内容。

Web体系结构

- Web体系结构主要由三部分构成：

（1）Web服务器。用户要访问Web页面或其他资源，必须事先有一个服务器来提供Web页面和这些资源，这种服务器就是Web服务器。

（2）Web客户端。用户一般是通过浏览器访问Web资源的，它是运行在客户端的一种软件。

（3）通信协议。客户端和服务器之间采用HTTP协议进行通信。HTTP协议是浏览器和Web服务器通信的基础，是应用层协议。

1.1.4 服务器和浏览器

- 在万维网上，如果一台连接到Internet的计算机希望给其他Internet系统提供信息，则它必须运行服务器软件，这种软件称为Web服务器。
- 对Web系统来说，客户软件通常是Web浏览器。

1．Web服务器

- Web服务器是向浏览器提供服务的程序，主要功能是提供网上信息浏览服务。Web服务器应用层使用HTTP协议，信息内容采用HTML文档格式，信息定位使用URL。
- 最常用的Web服务器是Apache服务器，它是Apache软件基金会（Apache Software Foundation）提供的开放源代码软件，是一个非常优秀的专业的Web服务器。

2. Web浏览器

- 浏览器是Web服务的客户端程序，可向Web服务器发送各种请求，并对从服务器发来的网页和各种多媒体数据格式进行解释、显示和播放。
- 浏览器的主要功能是解析网页文件内容并正确显示，网页一般是HTML格式。常见的浏览器有Internet Explorer、Firefox、Opera、和Chrome，浏览器是最常使用的客户端程序。

1.2 Web常用技术

- 1.2.1 HTTP概述
- 1.2.2 URL和URI
- 1.2.3 HTML和XML概述
- 1.2.4 CSS概述
- 1.2.5 JavaScript

1.2.1 HTTP概述

- 超文本传输协议（Hypertext Transfer Protocol，HTTP）是Web使用的协议。该协议详细规定了Web客户与服务器之间如何通信。
- 它是一个基于请求-响应（request-response）的无状态的协议，这种请求-响应的过程如图所示。

HTTP请求-响应过程

- 客户首先通过浏览器程序建立到Web服务器的连接并向服务器发送HTTP请求消息。
- Web服务器接收到客户的请求后，对请求进行处理，然后向客户发送回HTTP响应。
- 客户接收服务器发送的响应消息，对消息进行处理并关闭连接。

1.2.2 URL和URI

- URL（Uniform Resource Locator）称为统一资源定位器，指向Internet上位于某个位置的某个资源。资源包括HTML文件、图像文件和Servlet等。例如，下面是一些合法的URL。
- http://www.baidu.com/index.html
- http://www.mydomain.com/files/sales/report.html
- http://localhost:8080/helloweb/

URL组成

- URL通常由4部分组成：协议名称、所在主机的DNS名、可选的端口号和资源的名称。端口号和资源名称可以省略。

（1）最常使用的协议是HTTP协议，其他常用协议包括FTP协议、TELNET协议、MAIL协议和FILE协议等。

（2）DNS即为服务器的域名，如www.tsinghua.edu.cn。

（3）端口号标明该服务是在哪个端口上提供的

（4）URL的最后一部分为资源在服务器上的相对路径和名称，如/index.html，它表示服务器上根目录下的index.html文件。

URI

- URI（Uniform Resource Identifier）称为统一资源标识符，是以特定语法标识一个资源的字符串。
- URI由模式和模式特有的部分组成，它们之间用冒号隔开，一般格式如下：

  schema:schema-specific-part

- URI的常见模式包括：file（表示本地磁盘文件）、ftp（FTP服务器）、http（使用HTTP协议的Web服务器）、mailto（电子邮件地址）等。

1.2.3 HTML和XML概述

- 超文本标记语言（HyperText Markup Language，HTML）是一种用来制作超文本文档的简单标记语言。所谓超文本是指用HTML编写的文档中可以包含指向其他文档或资源的链接，该链接也称为超链接（hyperlink）。通过超链接，用户可以很容易访问所链接的资源。

HTML标签

- HTML文档是由一些标签（tag）组成的文本文件，标签标识了内容和类型，Web浏览器通过解析这些标签进行显示。

HTML常用标签

标签名	说　明	标签名	说　明
<html>	HTML文档的开始	 	换行
<head>	文档的头部	<hr>	水平线。
<title>	文档的标题	<a>	锚
<meta>	关于XHTML文档的元信息	<img>	图像
<link>	文档与外部资源的关系	<table>	表格
<script>	客户端脚本	<tr>	表格中的行
<style>	样式信息	<td>	表格中的单元
<body>	文档的主体	<form>	表单
<h1>~<h6>	标题	<input>	输入控件
<p>	段落		列表的项目
<b>	粗体字		文档中的节、块或区域

1.2.3 HTML和XML概述

- 程序1.1 register.html
- 该页面运行结果如图所示。

1.2.4 CSS概述

- CSS（Cascading Style Sheets）是层叠样式表的意思，它是一种用来表现HTML或XML等文件样式的语言。
- CSS是能够真正做到网页表现与内容分离的一种样式设计语言。

样式表的三种使用方法

（1）内联样式，在元素标签内使用style属性指定样式，style属性可以包含任何CSS样式声明，如设置段落首行缩进：

```
<p style="text-indent:2em">该段落首行缩进2em。</p>
```

样式表的三种使用方法

（2）内部样式表，在单个页面中使用<style>标签在文档的头部定义样式表，这种样式只能被定义它的页面使用，例如：

```
<style type="text/css">
  h1 {color:#f00}
  body{background-
  image:url(images/bg.gif)}
</style>
```

样式表的三种使用方法

（3）外部样式表，把声明的样式保存在样式文件中，当某个页面需要样式时，通过<link>标签或<style>标签连接外部样式表文件。外部样式表以.css作为文件扩展名，例如 styles.css。

下面标签引用外部样式表css\layout.css。

```
<link href="css\layout.css" rel="stylesheet"
      type="text/css" />
```

1.2.4 CSS概述

- 程序1.2　index.html
- 程序1.3　layout.css
- 该页面运行结果如图所示。

1.2.5 JavaScript

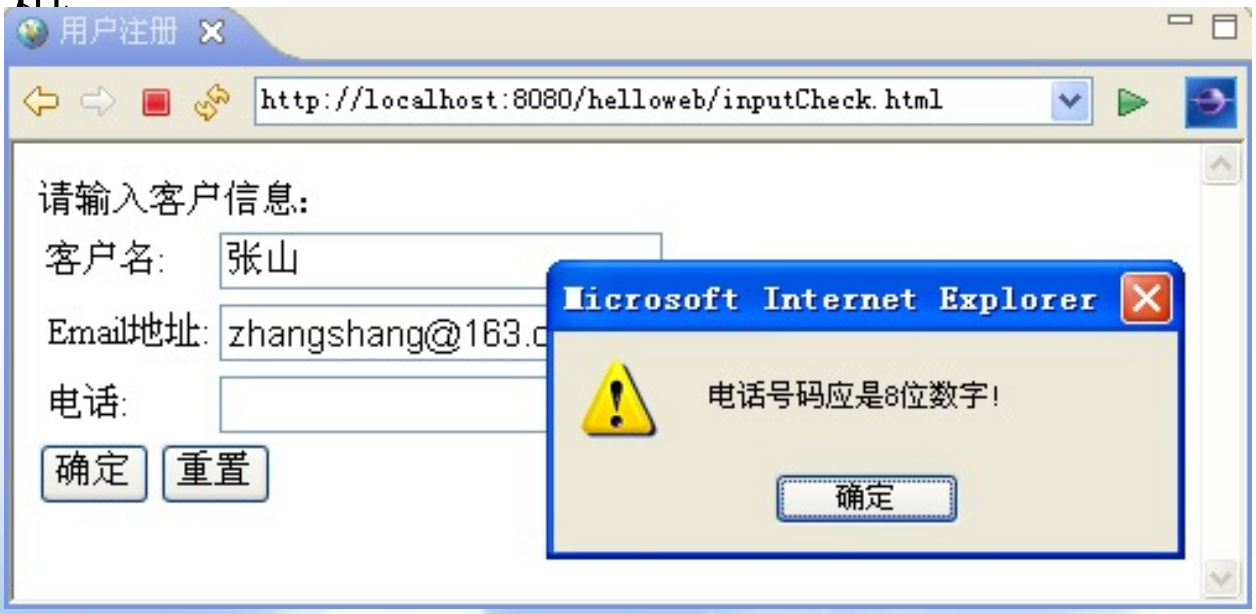
- JavaScript是一种广泛用于客户端Web开发的脚本语言，常用来给HTML网页添加动态功能。
- JavaScript是一种基于对象和事件驱动并具有相对安全性的客户端脚本语言 。
- 在HTML页面中通过<script>标签定义JavaScript脚本。<script>标签内既可以包含脚本语句，也可以通过src属性指向外部脚本文件。

```
<script type="text/javascript"
src="js/check.js"></script>
```

- 程序1.4 inputCheck.html

1.2.5 JavaScript

- 该HTML页面中，通过<script>和</script>在页面中嵌入了JavaScript语言代码。这里定义了一个名为custCheck函数，然后在页面的表单中，通过表单元素的onsubmit事件调用该函数，函数检查用户输入的数据，如果输入错误将弹出警告框



1.3 服务器资源

- 1.3.1 主动资源与被动资源
- 1.3.2 静态文档和动态文档
- 1.3.3 服务器端动态Web文档技术
- 1.3.4 客户端动态Web文档技术

1.3.1 主动资源与被动资源

- 可以把Web资源分为被动的和主动的。如果资源本身没有任何处理功能它就是被动的，如果资源有自己的处理能力，它就是主动的。
- Web应用程序通常是主动资源和被动资源的混合。

1.3.2 静态文档和动态文档

- Web文档是一种重要的Web资源，它通常是使用某种语言（如HTML，JSP等）编写的页面文件，因此也称为Web页面。Web文档又分为静态文档和动态文档。
- 在Web发展的早期，Web文档只是一种以文件的形式存放在服务器端的文档。客户发出对该文档的请求，服务器返回这个文件。这种文档称为静态文档（static document）。

1.3.2　静态文档和动态文档

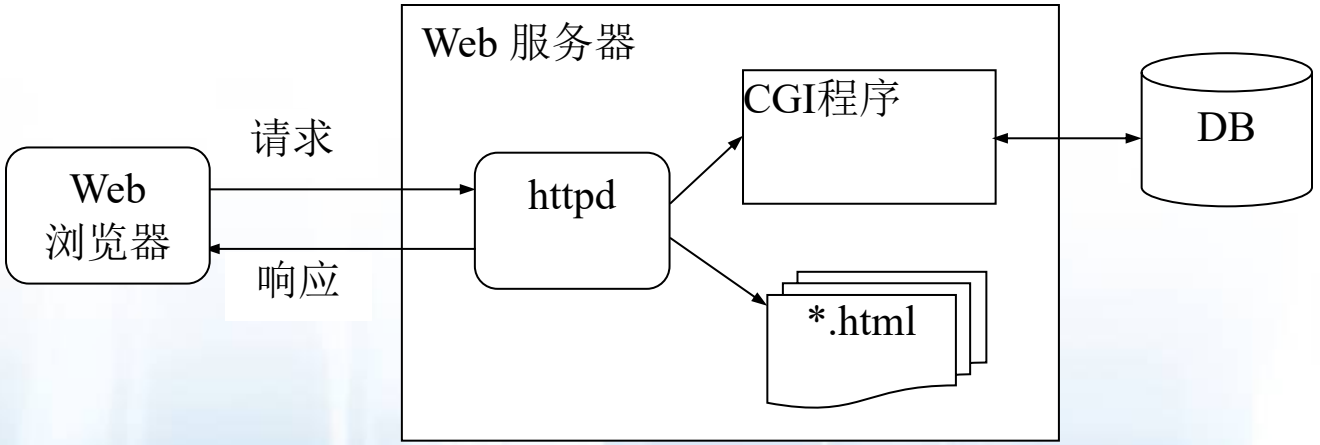
- 动态文档（dynamic document）是指文档的内容可根据需要动态生成。动态文档技术又分为服务器端动态文档技术和客户端动态文档技术。

1.3.3 服务器端动态Web文档技术

- 目前，在服务器端动态生成Web页面有多种方法。
  - **CGI技术**
  - **服务器扩展技术**
  - **在HTML页面中嵌入脚本技术**

1. CGI技术

- CGI是一种标准化的接口，允许Web服务器与后台程序和脚本通信，这些后台程序和脚本能够接受输入信息，访问数据库，最后生成HTML页面作为响应。



2. 服务器扩展技术

- 使用CGI方法的主要问题是效率低。对CGI程序的每次调用都创建一个操作系统进程，当多个用户同时访问CGI程序时，将加重处理器的负载。
- 使服务器支持单独的可执行模块，当服务器启动时该模块就装入内存并只初始化一次。然后，就可以通过已经驻留在内存的模块副本为每个请求提供服务。这些独立的可执行的模块称为服务器扩展。

3. 在HTML页面中嵌入脚本技术

- 另一种常见的实现动态文档技术是在Web页面中嵌入某种语言的脚本，然后让服务器来执行这些脚本以便生成最终发送给客户的页面。
- 目前比较流行的技术有
  - ASP.NET
  - PHP
  - JSP

1.3.4　客户端动态Web文档技术

- CGI、ASP、PHP和JSP脚本解决了处理表单以及与服务器上的数据库进行交互的问题。它们都可以接受来自表单的信息，在一个或多个数据库中查找信息，然后利用查找的结果生成HTML页面。
- 通常使用JavaScript结合DOM技术实现客户端动态Web文档技术。

1.4 Tomcat服务器

- Tomcat是Apache软件基金会（Apache Software Foundation，ASF）的开源产品，是 Servlet和JSP（JavaServer Pages）技术的实现。
- Tomcat服务器的最新版本Tomcat 7.0.39实现了Servlet 3.0和JSP 2.2的规范，另外它本身具有作为Web服务器运行的能力，因此不需要一个单独的Web服务器。本书所有程序都在Tomcat服务器中运行。

1.4.1 Tomcat下载与安装

- 可以到http://tomcat.apache.org/网站下载各种版本的Tomcat服务器。
- 必须先安装Java运行时环境
- 下载后的文件名为apache-tomcat-7.0.39.exe

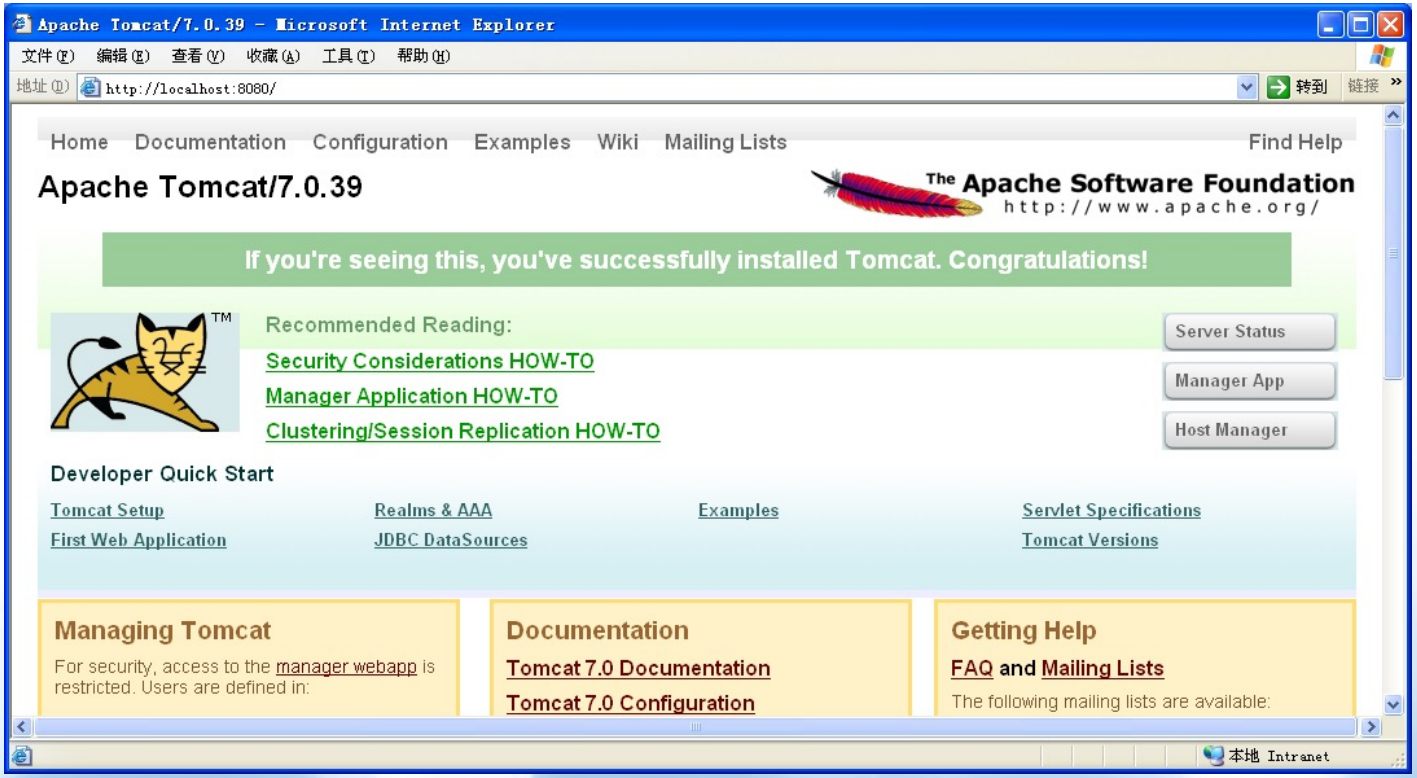
1.4.2 Tomcat的安装目录

- 安装结束后，打开资源管理器查看Tomcat 安装的目录结构

目　录	说　明
/bin	存放启动和关闭Tomcat的脚本文件
/conf	存放Tomcat服务器的各种配置文件，其中包括servler.xml、tomcat-users.xml和web.xml等文件
/lib	存放Tomcat服务器及所有Web应用程序都可以访问的库文件
/logs	存放Tomcat的日志文件
/temp	存放Tomcat运行时产生的临时文件
/webapps	存放所有Web应用程序的根目录
/work	存放JSP页面生成的Servlet源文件和字节码文件

1.4.3 测试Tomcat

- 打开浏览器，输入http://localhost:8080/，如能看到下图所示的页面，说明Tomcat服务器工作正常。



1.4.4 配置Tomcat的服务端口

- 在Tomcat安装时如果没有修改端口号，则默认的端口号为8080。
- 要修改Tomcat的端口号需要编辑<*tomcat-install*>\conf\server.xml文件，将Connector元素的port属性从8080修改为80，并重新启动服务器。

1.4.5 Tomcat的启动和停止

- 在使用Tomcat服务器开发Web应用程序时，经常在做了某种配置后为使配置生效需要重新启动Tomcat服务器。这可通过<*tomcat-install*>\bin中的tomcat7w.exe工具实现，双击该文件 ，单击【General】页面中的【Stop】按钮即停止服务器，再单击【Start】按钮即重新启动服务器。
- 打开【控制面板】中【管理工具】的【服务】窗口可以查看服务的启动情况 。

1.5 Servlet与JSP入门

- 1.5.1 Servlet
- 1.5.2 Web容器
- 1.5.3 JSP页面

1.5.1 Servlet

- Servlet可翻译成服务器端小程序，它是使用Servlet API以及相关的类编写的Java程序，这种程序运行在Web容器中，主要用来扩展Web服务器的功能。
- Servlet技术实际上是CGI技术的一种替代。下面是一个简单的Servlet程序。
- 程序1.5 HelloServlet.java

1.5.2 Web容器

- Web服务器使用一个单独的模块装载和运行Servlet和JSP页面，这个模块称为**Servlet容器**（container），或称Web容器。
- Tomcat就是一个**Web容器**。 Tomcat又具有Web服务器的功能，有时我们也称其为Web服务器。

1.5.3　JSP页面

- **JSP（JavaServer Pages）**页面是在HTML页面中嵌入JSP元素的页面，这些元素称为JSP标签。
- JSP元素具有严格定义的语法并包含完成各种任务的语法元素，比如声明变量和方法、JSP表达式、指令和动作等。
- 程序1.6　hello.jsp

1.6 MVC设计模式

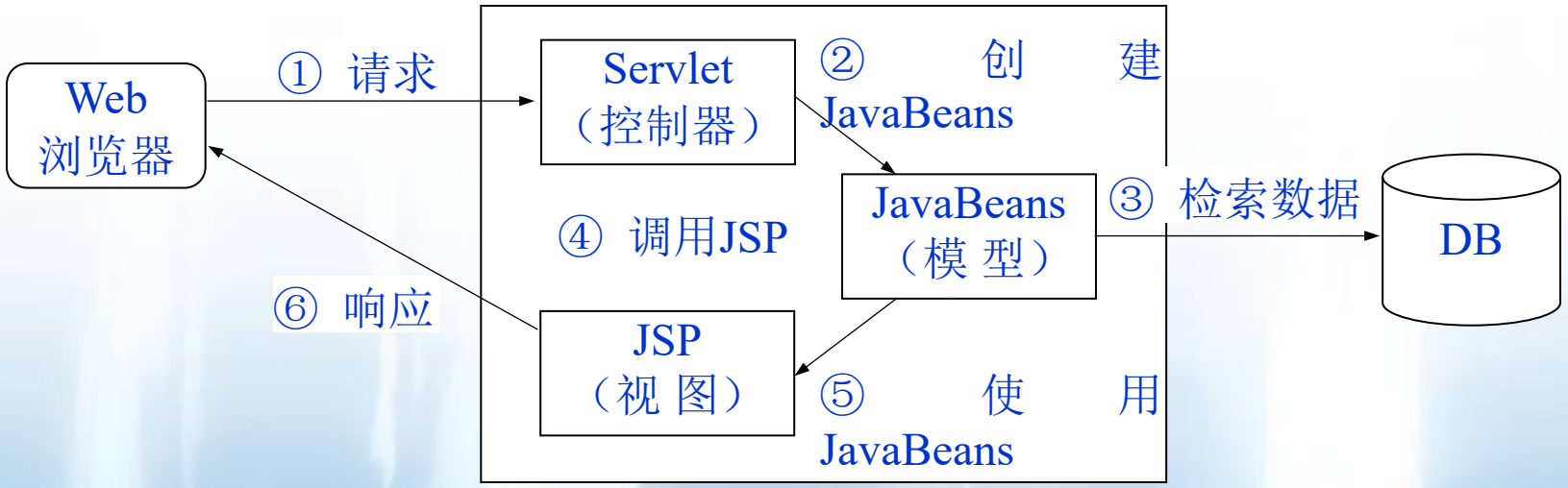
- Sun 公司推出Servlet技术的主要目的是代替CGI编程。可以把Servlet看成是含有HTML的Java代码。
- Sun公司在推出JSP技术后提出了建立Web应用程序的两种体系结构方法。这两种方法分别称为**JSP Model 1**体系结构和**JSP Model 2**体系结构，二者的差别在于处理请求的方式不同。

1.6.1 Model 1体系结构

- 在Model 1体系结构中，每个请求的目标都是JSP页面。
- 在该结构中没有一个核心组件控制应用程序的工作流程，所有的业务处理都使用JavaBeans实现。
- 该结构具有严重的缺点。首先，它需要将实现业务逻辑的大量Java代码嵌入到JSP页面中，这对不熟悉服务器端编程的Web页面设计人员将产生困难。第二，这种方法并不具有代码可重用性。

1.6.2 Model 2体系结构

- Model 2体系结构如图所示。这种体系结构又称为**MVC（Model-View-Controller）设计模式**。在这种结构中，将Web组件分为**模型（model）**、**视图（view）**和**控制器（controller）**，每种组件完成各自的任务。



1.7 小 结

- 本章概述了Web应用开发的主要技术和基本原理。其中包括Web技术的基本概念、浏览器和服务器的概念、 HTTP协议、动态Web文档技术等 。
- 本章还简要介绍了Tomcat服务器的安装与配置，讨论了什么是Servlet和Servlet容器，给出了一个简单的Servlet的开发执行过程。
- 简单介绍了MVC设计模式。