

第 1 部分

JSP 基础

- » 第 1 章 走进 JSP
- » 第 2 章 掌握 JSP 语法
- » 第 3 章 JSP 内置对象
- » 第 4 章 Servlet 技术
- » 第 5 章 JSP 使用 Model2 实现登录模块




12P 基盤

12P 設置	第 1 章	12P
12P 設置	第 2 章	12P
12P 設置	第 3 章	12P
12P 設置	第 4 章	12P
12P 設置	第 5 章	12P

第 1 章

走进 JSP

( 视频讲解：10 分钟)

本章将带领读者走进 JSP 开发领域，开始学习 Java 语言的 Web 开发技术。JSP 的全称是 Java Server Pages，它是 Java 开发 Web 程序的基础与核心，也是目前流行的 Web 开发技术中应用最广泛的一种，主要用于开发企业级 Web 应用，属于 JavaEE 技术范围。

通过阅读本章，您可以：

- » 了解什么是 JSP
- » 了解 JSP 的工作原理
- » 掌握学习 JSP 技术的方法
- » 搭建 MySQL 数据库
- » 配置 MyEclipse 开发工具
- » 了解 JSP 程序的编写步骤
- » 掌握 JSP 常用资源

1.1 JSP 概述

本节将带领读者初探 JSP 技术, 熟悉其应用领域, 了解 JSP 的项目成功案例, 并指导读者如何学习 JSP 开发技术。

1.1.1 什么是 JSP

JSP (Java Server Pages) 是由 Sun 公司倡导、许多公司参与而建立的动态网页技术标准。它在 HTML 代码中嵌入 Java 代码片段 (Scriptlet) 和 JSP 标签, 构成了 JSP 网页。在接收到用户请求时, 服务器会处理 Java 代码片段, 然后生成处理结果的 HTML 页面返回给客户端, 客户端的浏览器将呈现最终页面效果。其工作原理如图 1.1 所示。

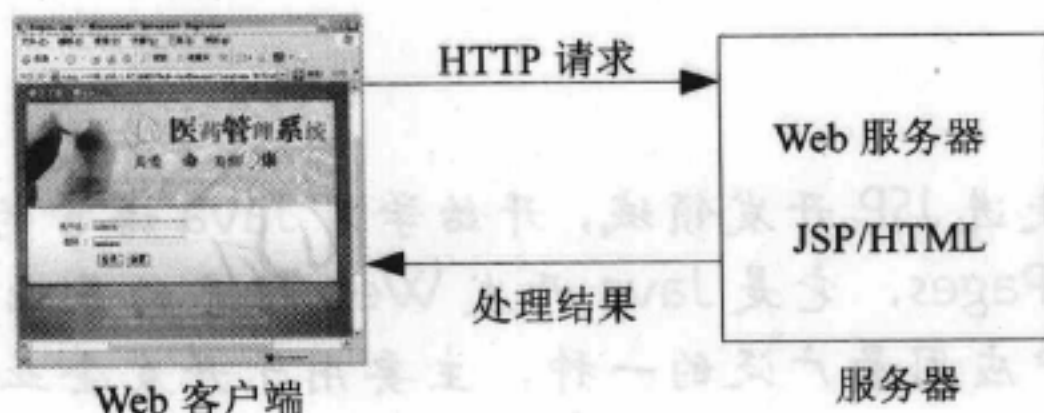


图 1.1 JSP 工作原理

1.1.2 项目成功案例

JSP 技术主要用于开发 Web 项目, 广泛应用于实际生活中的各行各业, 迄今已有很多成功案例, 其中金融、政治和企业类的较多。例如, 中国工商银行网站、中国光大银行网站、中国邮政储蓄银行网站、中国债券信息网、清华大学的本科招生网、金网在线网站、中国农业银行网站、中国建设银行网站、交通银行网站、深圳发展银行网站等都使用了 JSP 技术, 其首页效果分别如图 1.2~图 1.11 所示。

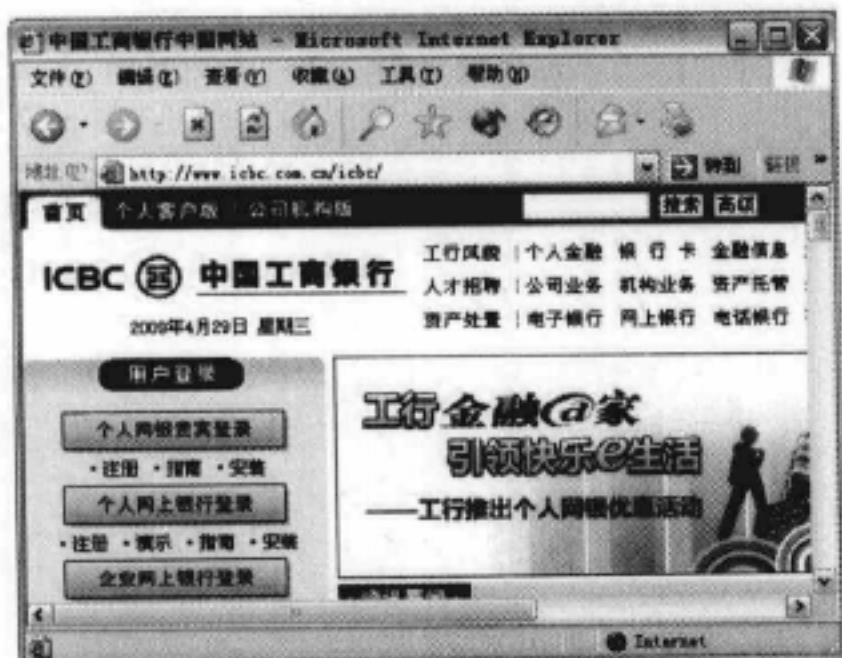


图 1.2 中国工商银行网站

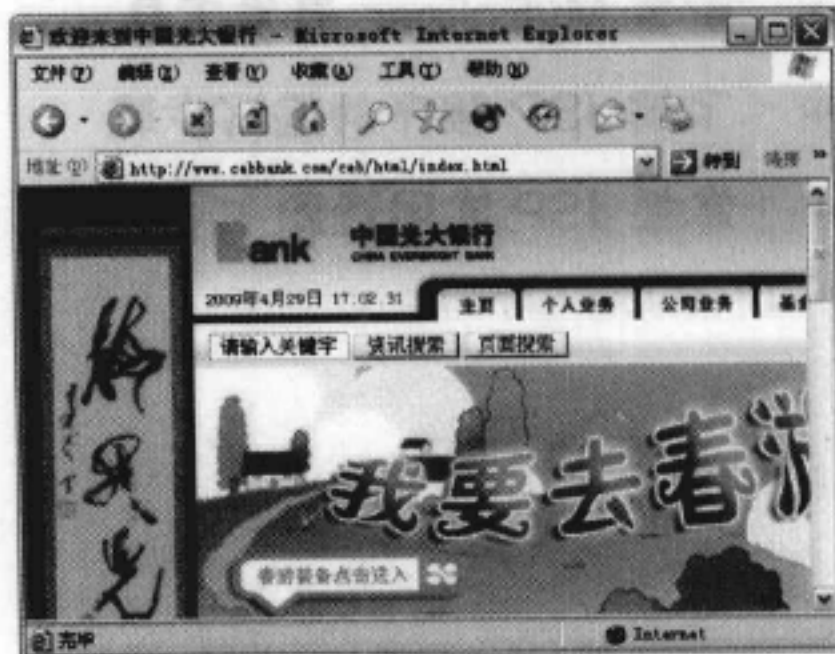


图 1.3 中国光大银行网站

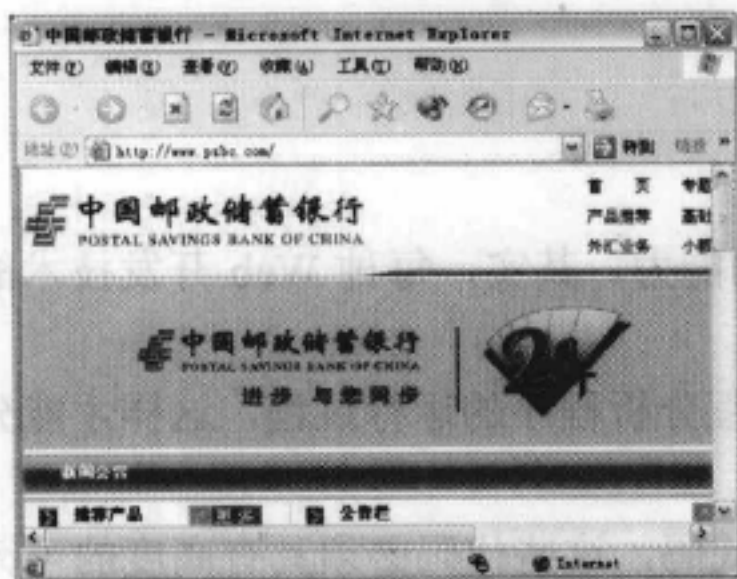


图 1.4 中国邮政储蓄银行网站



图 1.5 中国债券信息网

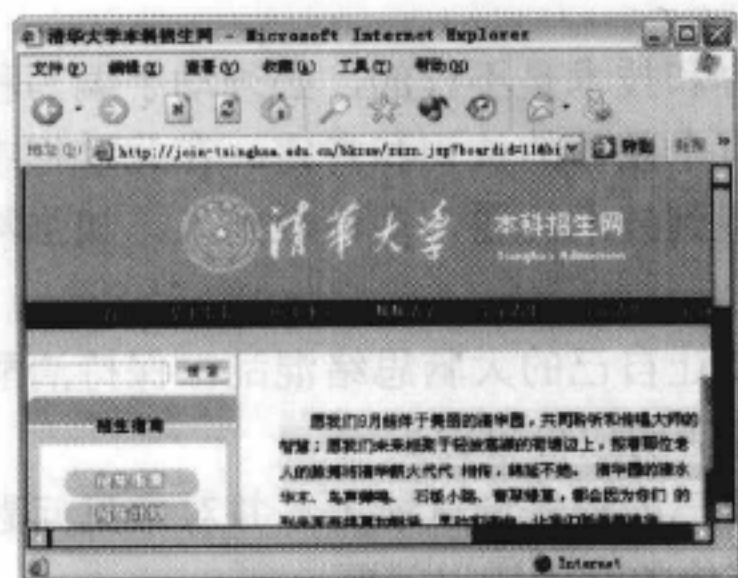


图 1.6 清华大学的本科招生网



图 1.7 金网在线网站



图 1.8 中国农业银行网站



图 1.9 中国建设银行网站



图 1.10 交通银行网站



图 1.11 深圳发展银行网站

1.1.3 如何学好 JSP

学好 JSP 技术，就是掌握 Java Web 网站程序开发的能力。其实，每种 Web 开发技术的学习方法都大同小异，需要注意的主要有以下几点。

- ☑ 了解 Web 设计流程与工作原理，能根据工作流程分析程序的运行过程，这样才能分析问题所在，快速进行程序调试。
- ☑ 了解 MVC 设计模式。开发程序必须编写程序代码，这些代码必须具有高度的可读性，这样编写的程序才有调试、维护和升级的价值。学习一些设计模式，能够更好地把握项目的整体结构。
- ☑ 多实践，多思考，多请教。只读懂书本中的内容和技术是不行的，必须动手编写程序代码，并运行程序、分析其结构，从而对学习的内容有个整体的认识和肯定。在此过程中，可用自己的方式思考问题，逐步总结提高编程思想。遇到技术问题，多请教别人，加强沟通，提高自己的技术和见识。
- ☑ 不要急躁。遇到技术问题，必须冷静对待，不要让自己的大脑思绪混乱，保持清醒的头脑才能分析和解决各种问题，可以尝试听歌、散步等活动放松自己。
- ☑ 遇到问题，首先尝试自己解决，这样可以提高自己的程序调试能力，并对常见问题有一定的了解，明白出错的原因，甚至举一反三，解决其他关联的错误问题。
- ☑ 多查阅资料。可以经常到 Internet 上搜索相关资料或者解决问题的办法，网络上已经摘录了很多人的问题和不同的解决办法，分析这些解决问题的方法，从中找出最好、最适合自己的。
- ☑ 多阅读别人的源代码，不但要看懂，还要分析编程者的编程思想和设计模式，并融为己用。
- ☑ HTML、CSS、JavaScript 技术是网页页面布局和动态处理的基础，必须熟练掌握，才能够设计出完美的网页。
- ☑ 掌握主流的框架技术，如 Struts、Hibernate 和 Spring 等。各种开源的框架很多，它们能够提高 JSP 程序的开发和维护效率，并减少错误代码，使程序结构更加清晰。
- ☑ 掌握 SQL 和 JDBC 对关系型数据库的操作。企业级程序开发离不开数据库，作为一名合格的程序开发人员，必须拥有常用数据库的管理能力，掌握 SQL 标准语法或者本书介绍的 Hibernate 框架。
- ☑ 要熟悉常用的 Web 服务器的管理，如 Tomcat，并且了解如何在这些服务器中部署自己的 Web 项目。

1.2 JSP 技术特征

JSP 技术所开发的 Web 应用程序是基于 Java 的，它拥有 Java 语言跨平台的特性，以及业务代码分离、组件重用、基础 Java Servlet 功能和预编译等特征。

1.2.1 跨平台

既然 JSP 是基于 Java 语言的, 那么它就可以使用 Java API, 所以它也是跨平台的, 可以应用在不同的系统中, 如 Windows、Linux、Mac 和 Solaris 等。这同时也拓宽了 JSP 可以使用的 Web 服务器的范围。另外, 应用于不同操作系统的数据库也可以为 JSP 服务, JSP 使用 JDBC 技术操作数据库, 从而避免了代码移植导致更换数据库时的代码修改问题。

正是因为跨平台的特性, 使得采用 JSP 技术开发的项目可以不加修改地应用到任何不同的平台上, 这也应验了 Java 语言的“一次编写, 到处运行”的特点。

1.2.2 业务代码分离

采用 JSP 技术开发的项目, 通常使用 HTML 语言来设计和格式化静态页面的内容, 而使用 JSP 标签和 Java 代码片段来实现动态部分。程序开发人员可以将业务处理代码全部放到 JavaBean 中, 或者把业务处理代码交给 Servlet、Struts 等其他业务控制层来处理, 从而实现业务代码从视图层分离。这样 JSP 页面只负责显示数据即可, 当需要修改业务代码时, 不会影响 JSP 页面的代码。

1.2.3 组件重用

JSP 中可以使用 JavaBean 编写业务组件, 也就是使用一个 JavaBean 类封装业务处理代码或者作为一个数据存储模型, 在 JSP 页面甚至整个项目中都可以重复使用这个 JavaBean。JavaBean 也可以应用到其他 Java 应用程序中, 包括桌面应用程序。

1.2.4 继承 Java Servlet 功能

Servlet 是 JSP 出现之前的主要 Java Web 处理技术。它接受用户请求, 在 Servlet 类中编写所有 Java 和 HTML 代码, 然后通过输出流把结果页面返回给浏览器。其缺点是: 在类中编写 HTML 代码非常不便, 也不利于阅读。使用 JSP 技术之后, 开发 Web 应用便变得相对简单快捷多了, 并且 JSP 最终要编译成 Servlet 才能处理用户请求, 因此我们说 JSP 拥有 Servlet 的所有功能和特性。

1.2.5 预编译

预编译就是在用户第一次通过浏览器访问 JSP 页面时, 服务器将对 JSP 页面代码进行编译, 并且仅执行一次编译。编译好的代码将被保存, 在用户下一次访问时, 直接执行编译好的代码。这样不仅节约了服务器的 CPU 资源, 还大大提升了客户端的访问速度。

1.3 Java Web 服务器

Web 服务器是运行及发布 Web 应用的容器, 只有将开发的 Web 项目放置到该容器中, 才能使网

络中的所有用户通过浏览器进行访问。开发 Java Web 应用所采用的服务器主要是与 JSP/Servlet 兼容的 Web 服务器, 比较常用的有 Tomcat、Resin、JBoss、WebSphere 和 WebLogic 等, 下面将分别进行介绍。

1.3.1 Tomcat 服务器

目前最为流行的 Tomcat 服务器是 Apache-Jakarta 开源项目中的一个子项目, 是一个小型、轻量级的支持 JSP 和 Servlet 技术的 Web 服务器, 也是初学者学习开发 JSP 应用的首选。本书中的所有项目都使用 Tomcat 作为 Web 服务器。

1.3.2 Resin 服务器

Resin 是 Caucho 公司的产品, 是一个非常流行的支持 Servlet 和 JSP 的服务器, 速度非常快。Resin 本身包含了一个支持 HTML 的 Web 服务器, 这使它不仅可以显示动态内容, 而且显示静态内容的能力也毫不逊色, 因此许多网站都使用 Resin 服务器构建。

1.3.3 JBoss 服务器

JBoss 是一种遵从 JavaEE 规范的、开放源代码的、纯 Java 的 EJB 服务器, 对于 J2EE 有很好的支持。JBoss 采用 JML API 实现软件模块的集成与管理, 其核心服务仅是提供 EJB 服务器, 不包括 Servlet 和 JSP 的 Web 容器, 不过它可以和 Tomcat 完美结合。

1.3.4 WebSphere 服务器

WebSphere 是 IBM 公司的产品, 可进一步细分为 WebSphere Performance Pack、Cache Manager 和 WebSphere Application Server 等系列, 其中 WebSphere Application Server 是基于 Java 的应用环境, 可以运行于 Sun Solaris、Windows NT 等多种操作系统平台, 用于建立、部署和管理 Internet 和 Intranet Web 应用程序。

1.3.5 WebLogic 服务器

WebLogic 是 BEA 公司的产品, 可进一步细分为 WebLogic Server、WebLogic Enterprise 和 WebLogic Portal 等系列, 其中 WebLogic Server 的功能特别强大。WebLogic 支持企业级的、多层次的和完全分布式的 Web 应用, 并且服务器的配置简单、界面友好。对于那些正在寻求能够提供 Java 平台所拥有的一切应用服务器的用户来说, WebLogic 是一个十分理想的选择。

1.4 MySQL 数据库的下载与安装

本书所有程序均采用 MySQL 作为数据库。MySQL 是开发源代码的关系型数据库管理系统, 可以

在多个系统平台中运行，目前广泛应用于 Internet 上的中小型网站。由于其体积小、运行速度快、成本低、开源等特性，许多 PHP、JavaWeb 等网站都采用了该数据库。本节将介绍 MySQL 数据库及其 GUI 工具的下载与安装。

1.4.1 下载 MySQL

(1) 在浏览器的地址栏中输入“http://www.mysql.com”，单击“转到”按钮或按 Enter 键打开 MySQL 数据库的官方网站，单击页面上的 Downloads 超链接，如图 1.12 所示。



图 1.12 单击 Downloads 超链接

(2) 在 MySQL 下载页面左侧依次单击 MySQL Community Servers/5.0 超链接，然后单击右侧的 Download 超链接，如图 1.13 所示。

(3) 向下滚动网页，找到 Windows downloads 标题下的 ZIP/Setup.EXE 格式的安装包，单击 Download 超链接，如图 1.14 所示。

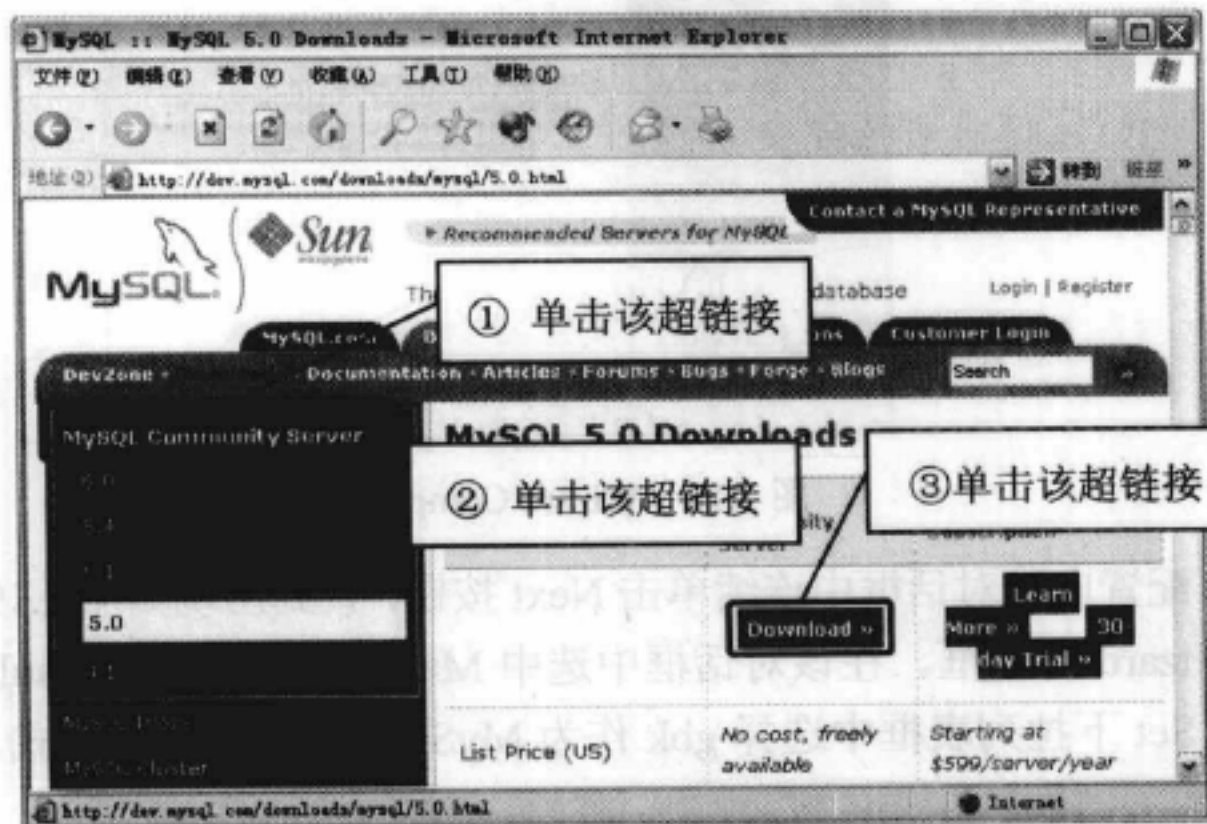


图 1.13 MySQL 下载页面



图 1.14 下载 ZIP/Setup.EXE 格式的安装包

1.4.2 安装与配置 MySQL

(1) 运行下载的 MySQL 安装文件，将出现 MySQL Server 5.0-Setup Wizard（安装向导）对话框，单击 Next 按钮，如图 1.15 所示。

(2) 在弹出的如图 1.16 所示对话框中选中 Custom 单选按钮，自定义 MySQL 的安装内容，然后单击 Next 按钮。

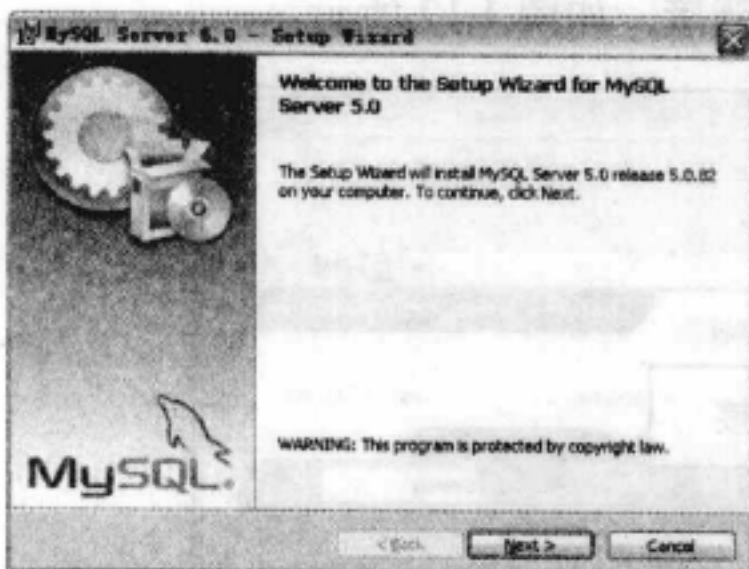


图 1.15 MySQL Server 5.0-Setup Wizard（安装向导）对话框

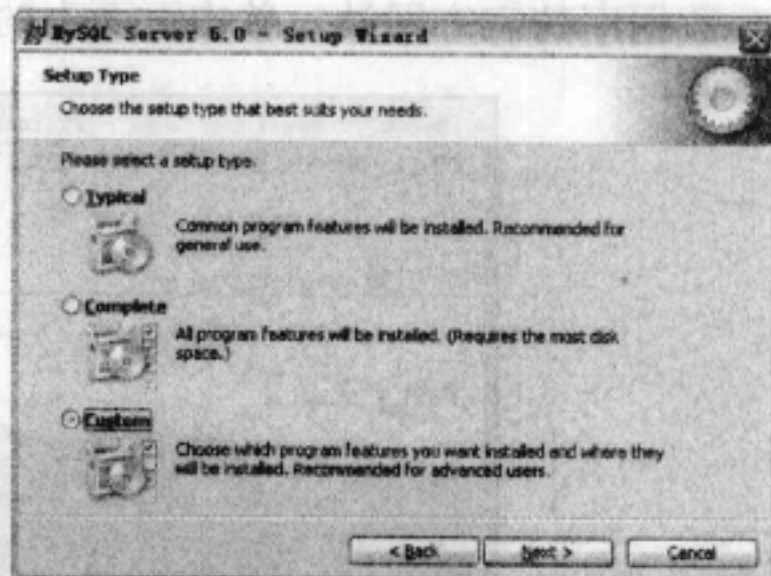


图 1.16 Setup Type 对话框

(3) 弹出 Custom Setup 对话框（如图 1.17 所示），在其中单击 Change 按钮选择 MySQL 的安装路径，例如 D:\MySQL\MySQL Server 5.0 文件夹，如图 1.17 所示。然后单击 Next 按钮，在弹出对话框中单击 Install 按钮，即可开始安装。

(4) 安装完成后，弹出 Wizard Completed 对话框，其中 Configure the MySQL Server now 复选框默认处于选中状态，也就是随后将启动 MySQL 服务器的配置向导，如图 1.18 所示。

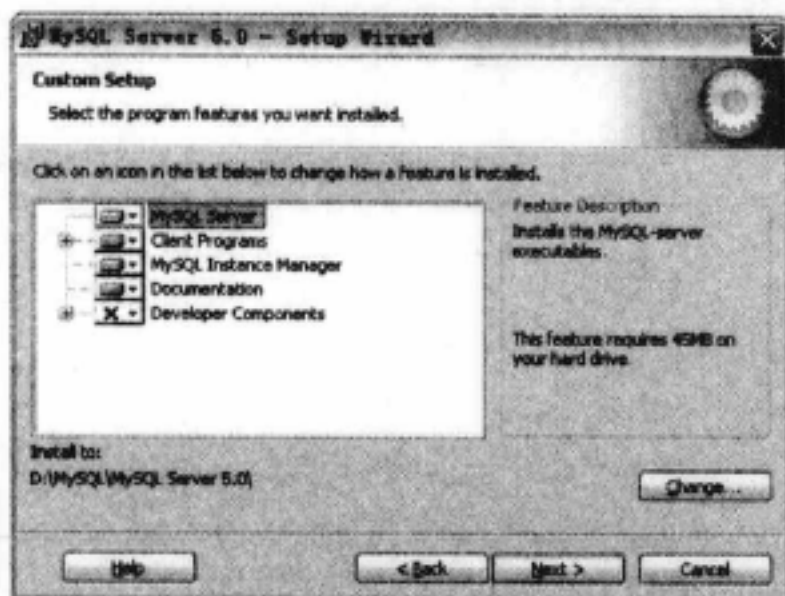


图 1.17 Custom Setup 对话框

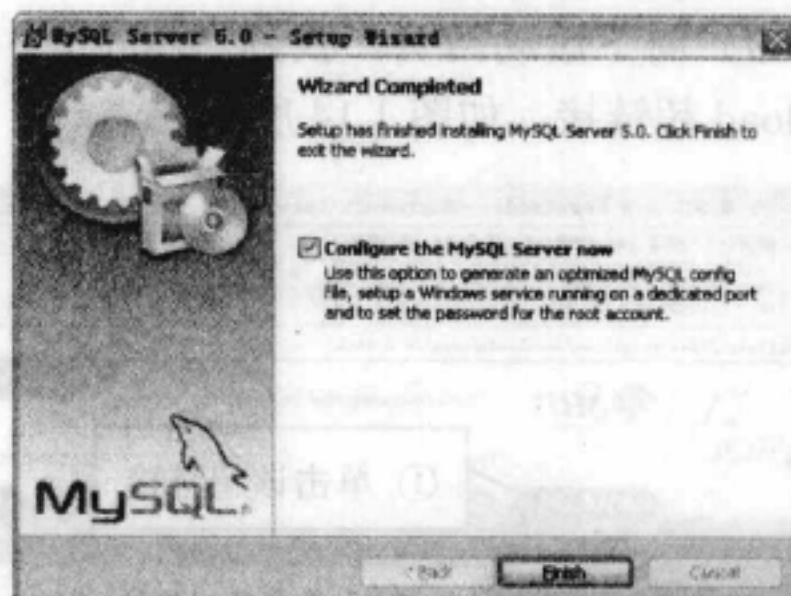


图 1.18 Wizard Completed 对话框

(5) 单击 Finish 按钮，在弹出的 MySQL 配置向导对话框中连续单击 Next 按钮，直到出现如图 1.19 所示 MySQL Server Instance Configuration Wizard 对话框。在该对话框中选中 Manual Selected Default Character Set/Collation 单选按钮，在 Character Set 下拉列表框中选择 gbk 作为 MySQL 的默认编码格式，然后单击 Next 按钮。

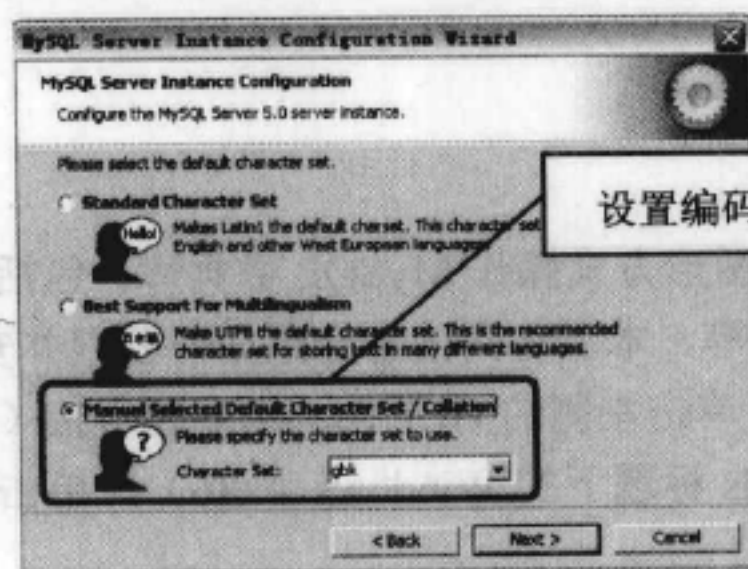


图 1.19 MySQL Server Instance Configuration Wizard 对话框

说明：如果 MySQL 的编码格式设置得与本书不同，在运行本书实例时可能会出现乱码的情况。

(6) 在弹出的如图 1.20 所示对话框中选中 Include Bin Directory in Windows PATH 复选框，将 MySQL 的命令文件夹包含在系统 PATH 变量中，这样在任何地方都可以使用 MySQL 命令文件夹中的命令文件了，然后单击 Next 按钮。

(7) 弹出如图 1.21 所示对话框，输入 MySQL 的密码，例如“111”（该密码将作为高级用户 root 的管理密码），然后单击 Next 按钮。

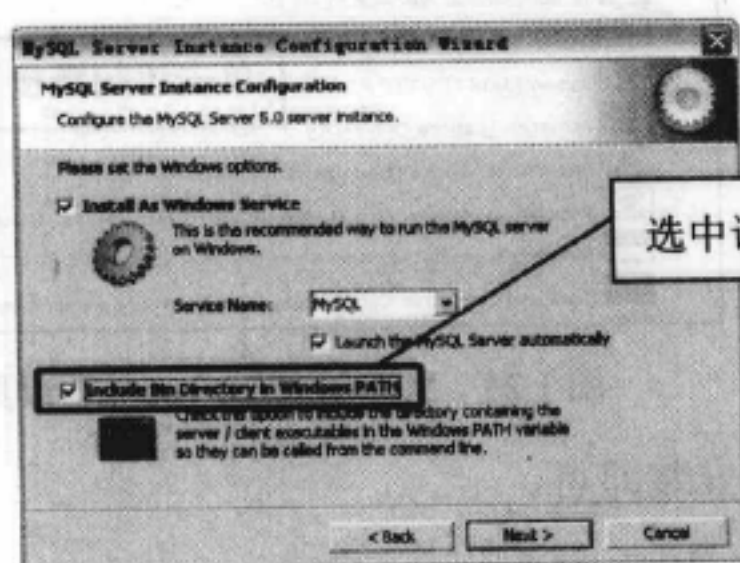


图 1.20 选中 Include Bin Directory in Windows PATH 复选框

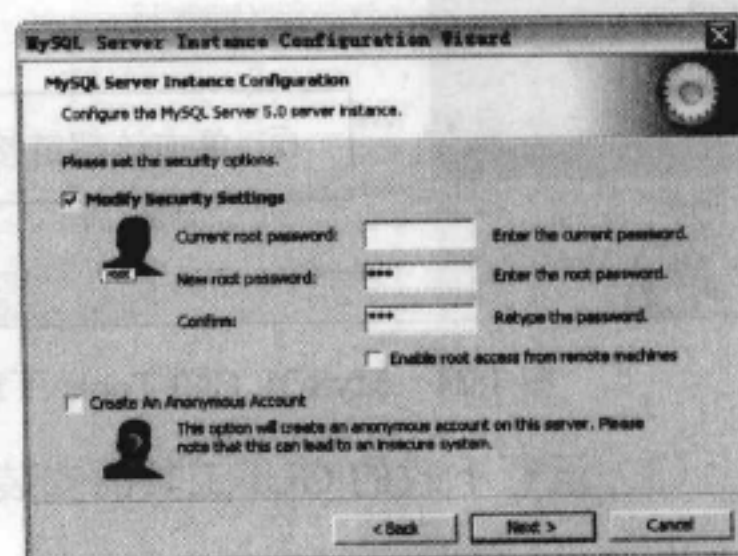


图 1.21 输入密码

(8) 弹出如图 1.22 所示对话框，其中显示了向导最后要完成的几项配置，单击 Execute 按钮执行配置。

(9) 正常完成所有配置后，单击 Finish 按钮，如图 1.23 所示。

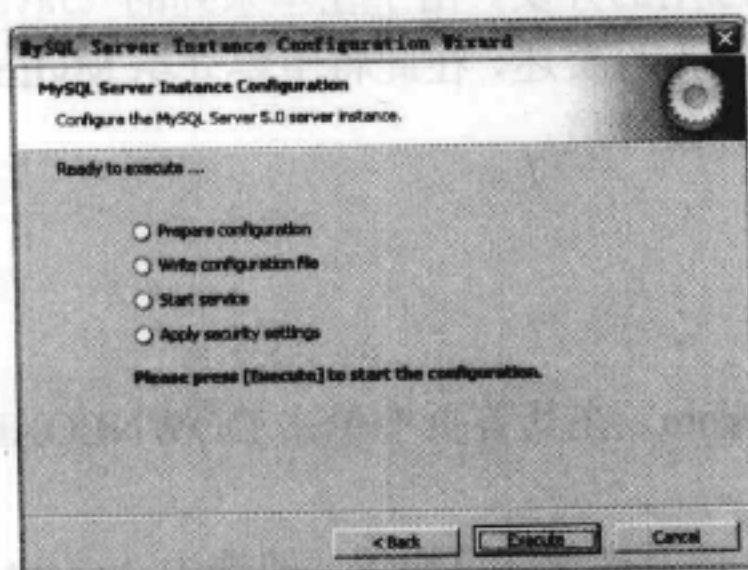


图 1.22 执行配置

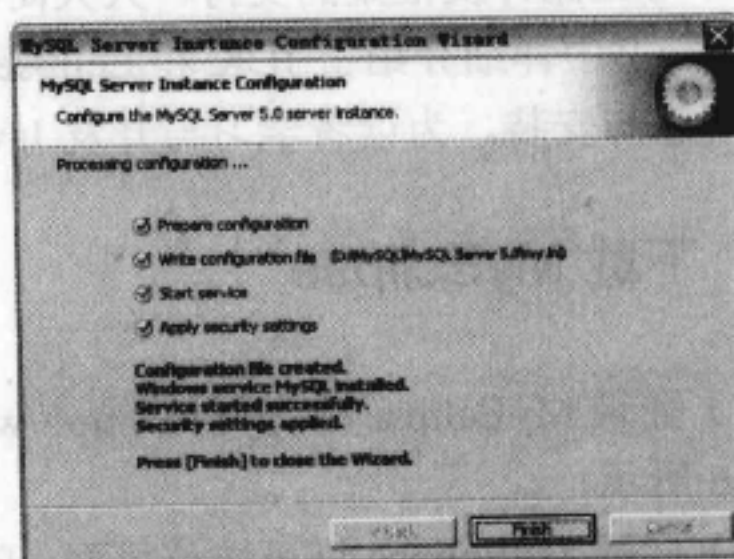


图 1.23 完成配置

1.4.3 下载并安装 GUI 工具

MySQL 的 GUI 工具能够以图形方式操作 MySQL 数据库，包括创建数据库、数据表、数据库备份与恢复、用户管理、服务器管理等。本节将向读者介绍如何获取并安装这个 GUI 工具。

(1) 在 MySQL 的下载页面单击左侧的 GUI Tools/5.0 超链接（这是目前最新的版本），然后在页面的右侧找到 Windows downloads 标题下的 Windows (x86)，单击其右侧的 Pick a mirror 超链接，如图 1.24 所示。

(2) 在打开的下载页面中包含世界各地不同地区、不同国家的服务器镜像。向下滚动页面，查找 Asia（亚洲）各国或地区的镜像站点。找到最近的服务器后，单击 HTTP 超链接下载安装文件，如图 1.25 所示。

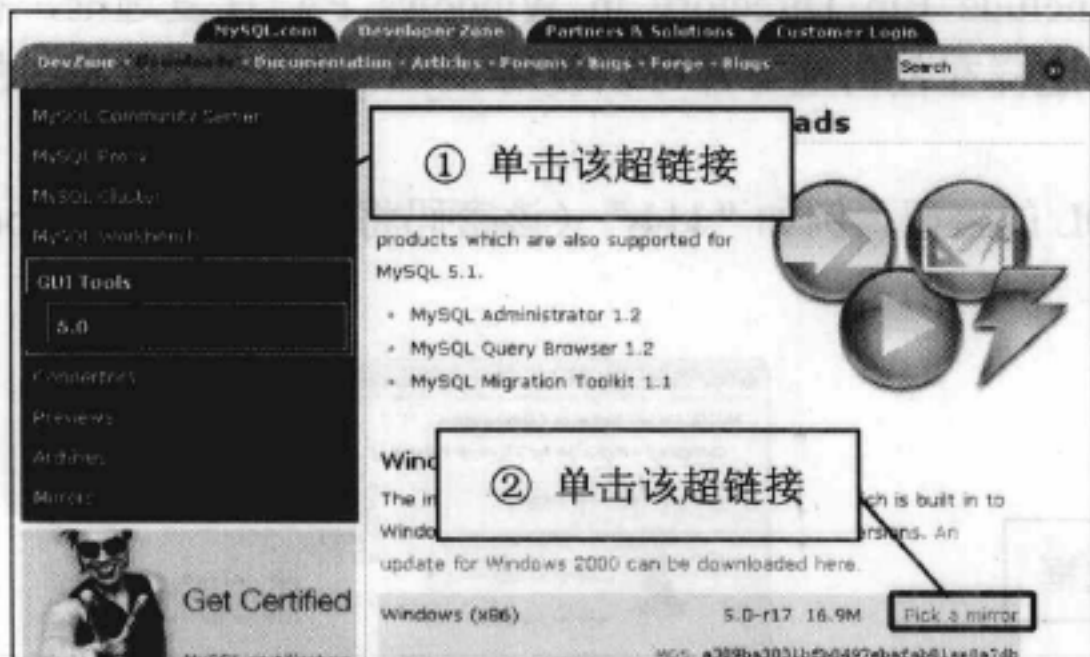


图 1.24 MySQL GUI Tools 下载页面



图 1.25 MySQL GUI Tools 下载页面

(3) 运行下载的 GUI 工具的安装文件，采用默认值安装即可。

1.5 MyEclipse 的下载、安装与配置

MyEclipse 是 Eclipse 的一个插件，但其也可以作为一个独立的软件。它集成了对 Struts、Spring 和 Hibernate 等主流开发框架的支持，大大简化了 Java 企业级应用的开发。由于在本系列的《Java 开发实战宝典》一书中详细介绍了 JDK 和 Eclipse 的下载与安装，恕不再赘述，在此将主要介绍 MyEclipse 7.0 插件的下载与安装，为读者学习与开发 Java Web 项目搭建开发环境。

1.5.1 下载 MyEclipse

(1) 登录 MyEclipse 官方网站 <http://www.myeclipseide.com>，在其首页中单击 DOWNLOAD 按钮，如图 1.26 所示。

(2) 打开 MyEclipse 许可协议页面，阅读协议内容后，必须接受许可协议（即选中 Accept License Agreement 复选框），才能单击 DOWNLOAD 按钮，如图 1.27 所示。



图 1.26 MyEclipse 首页截图



图 1.27 许可协议页面

(3) 打开 MyEclipse 的下载页面，其中提供了站点更新和独立下载两种方式，并且包含适用于多个系统平台的下载超链接，如图 1.28 所示。

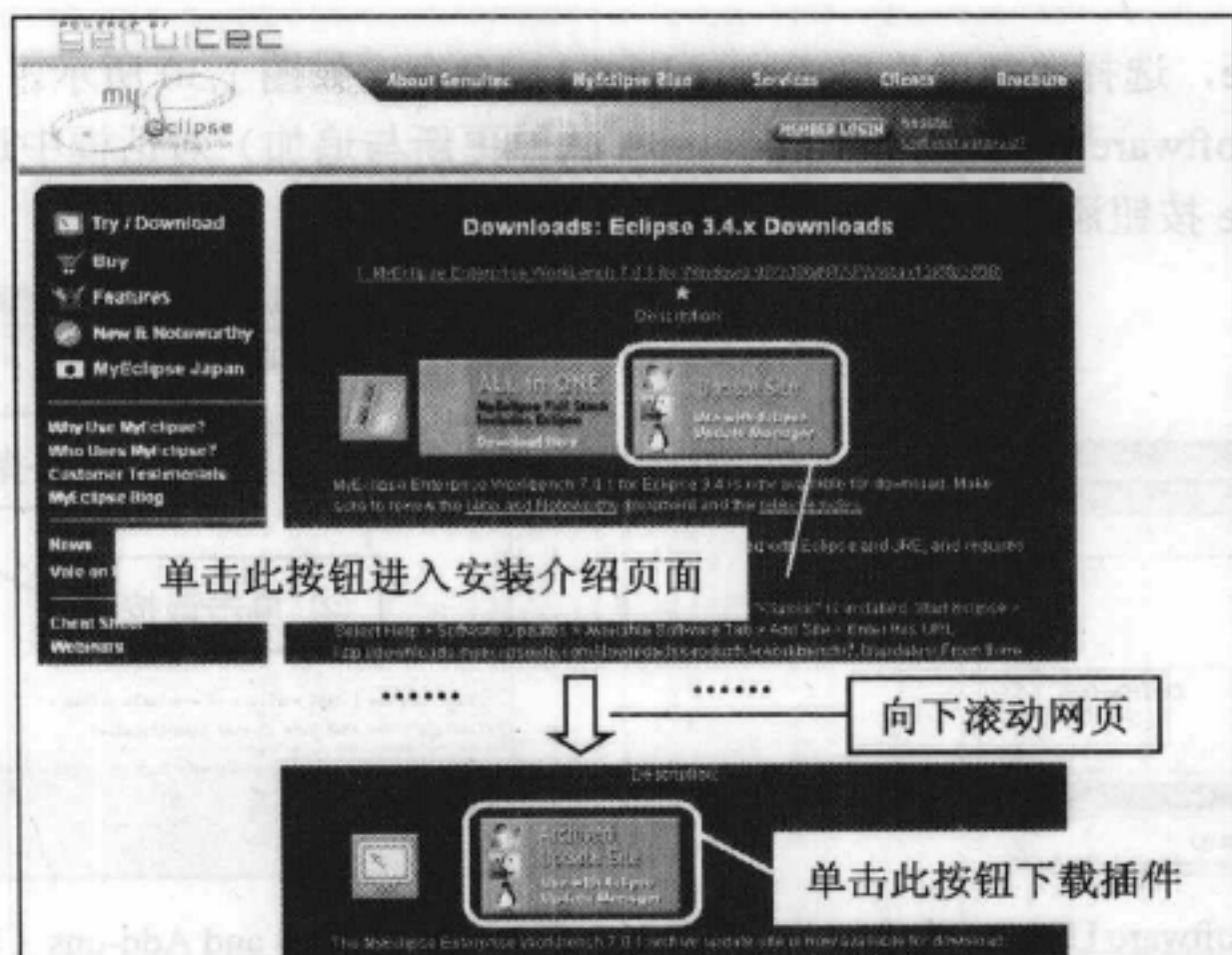


图 1.28 MyEclipse 的下载页面

本书以插件方式安装 MyEclipse，所以采用站点更新方式。这种方式还可细分为两种下载方法：

- ☑ 向下滚动网页到最底部，单击 Archived Update Site 按钮，下载整个 MyEclipse 的插件压缩文档，然后在 Eclipse 中根据压缩包进行软件的更新与安装。
- ☑ 单击 Update Site 按钮，进入 Installing MyEclipse 7.0 from an Update Site 页面，在其中即可看到从更新站点安装 MyEclipse 7.0 插件的 URL 地址（在撰写本书时，该 URL 地址是 <http://downloads.myeclipseide.com/downloads/products/eworkbench/7.0/updates/>），如图 1.29 所示。复制该 URL 地址，在安装步骤中会用到。

提示：这种方法主要是针对 Windows 系统平台。

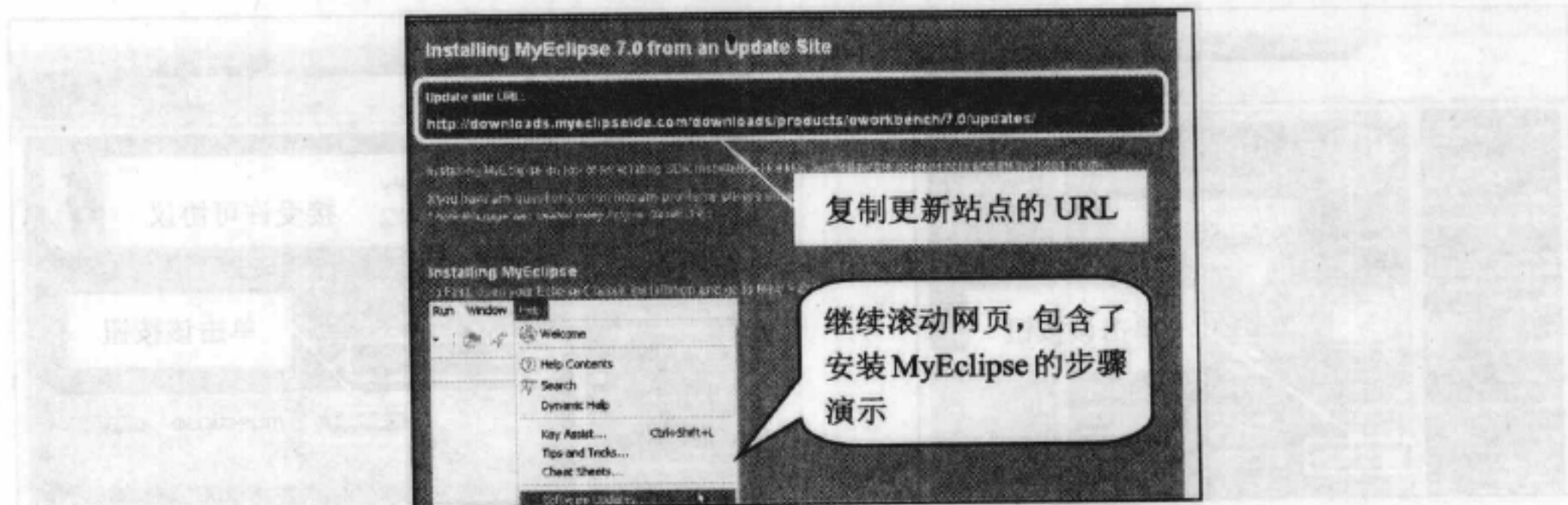


图 1.29 Installing MyEclipse 7.0 from an Update Site 页面

1.5.2 安装 MyEclipse

- (1) 启动 Eclipse, 选择“帮助”/Software Updates 命令, 如图 1.30 所示。
- (2) 在弹出的 Software Updates and Add-ons (软件更新与追加) 对话框中选择 Available Software 选项卡, 单击 Add Site 按钮添加更新站点的位置, 如图 1.31 所示。

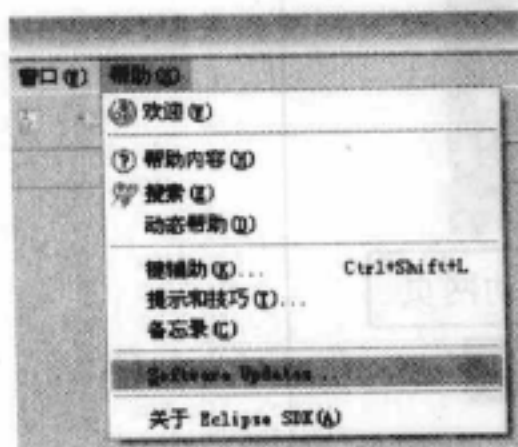


图 1.30 选择“帮助”/ Software Updates 命令

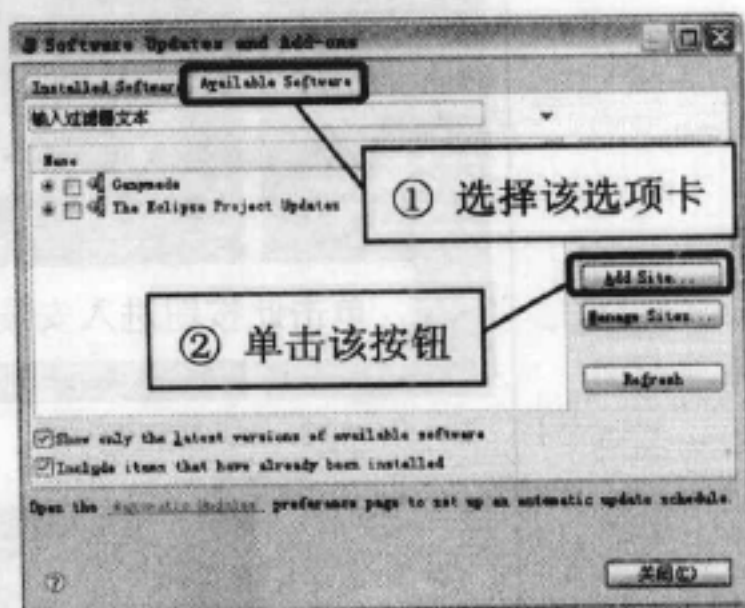


图 1.31 Software Updates and Add-ons (软件更新与追加) 对话框

- (3) 弹出 Add Site (添加站点地址) 对话框, 在 Location 文本框中粘贴从图 1.29 复制的 Update Site 的 URL 地址 (如果采用的是第一种方式, 即下载 MyEclipse 的整个插件压缩文档, 可以单击 Add Site 对话框中的 Archive 按钮, 选择压缩文档的文件位置), 单击“确定”按钮, 如图 1.32 所示。

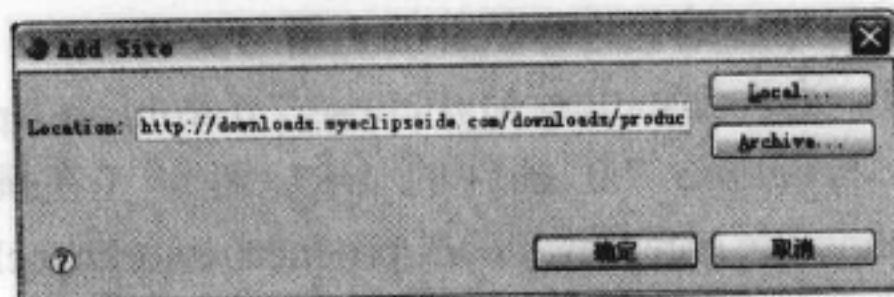


图 1.32 Add Site (添加站点地址) 对话框

- (4) 展开 MyEclipse 7.0 Update Site 树节点, 选中 MyEclipse Enterprise Workbench version 7.0 子节点, 单击 Install 按钮, 如图 1.33 所示。

(5) 弹出 Install 对话框，从中确认所选安装项目，单击“下一步”按钮，如图 1.34 所示。

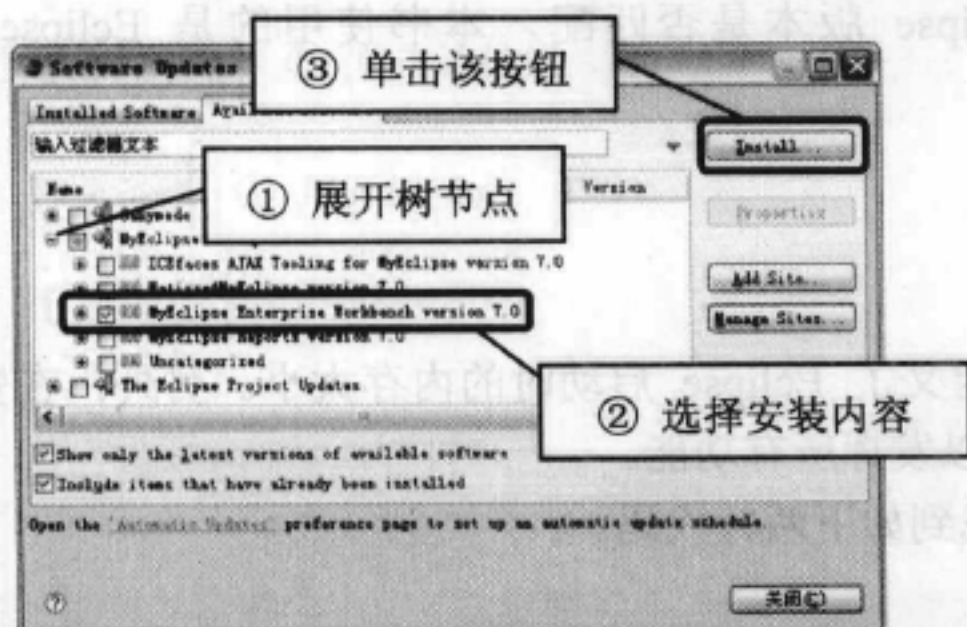


图 1.33 选择安装项目

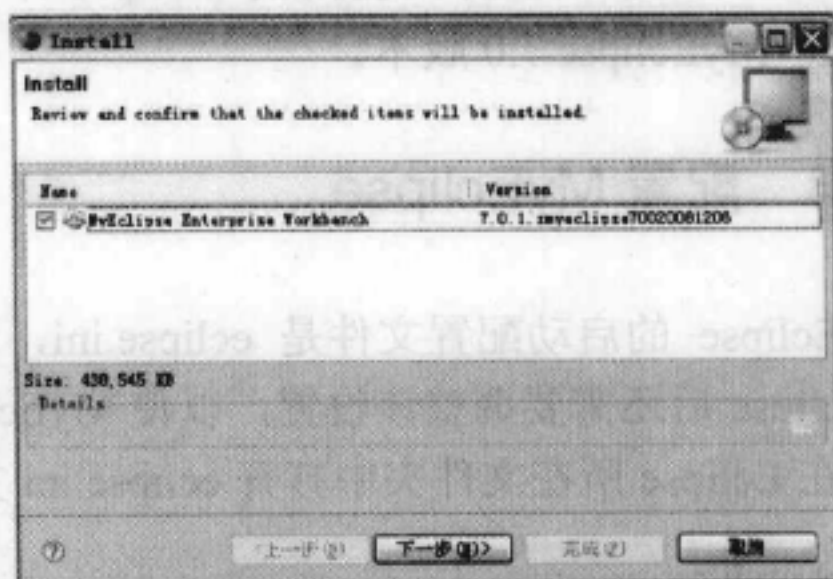


图 1.34 确认所选安装项目

(6) 弹出 Install—Review Licenses 对话框，从中选中 I accept the terms of the license agreements 单选按钮，即接受许可协议，然后单击“完成”按钮，如图 1.35 所示。

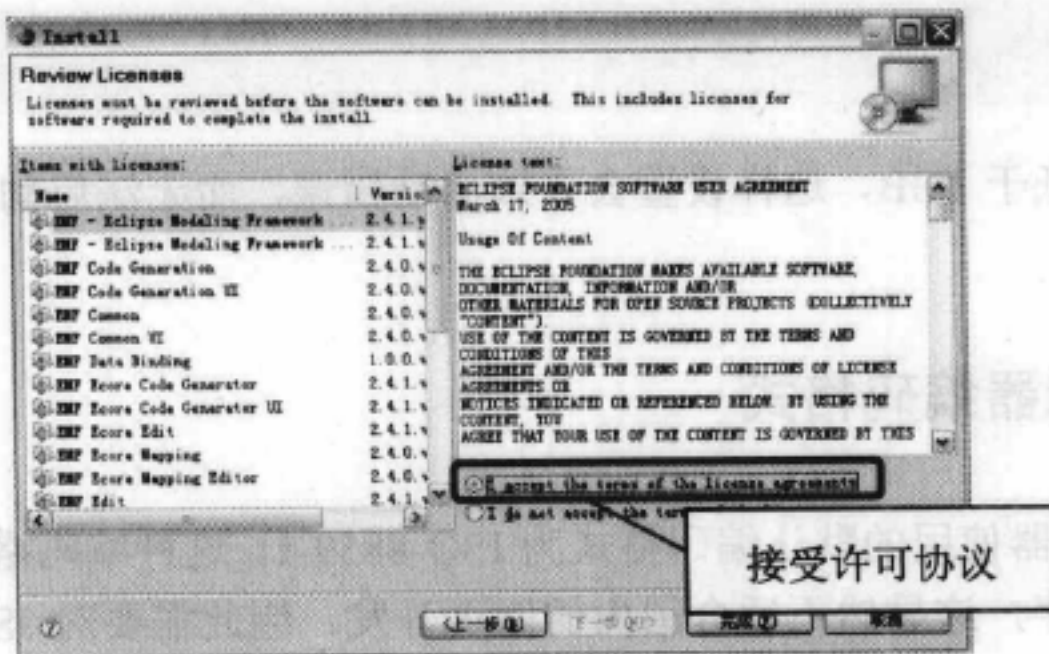


图 1.35 Install—Review Licenses 对话框

(7) 弹出安装进度提示对话框，从中用户可以了解安装任务的执行情况，如图 1.36 所示。如果使用的是下载到本地的 MyEclipse 插件压缩文档进行安装，那么该安装将会很快完成；而使用站点网址进行安装，则根据不同的网络带宽，速度会有所不同。

(8) 完成安装后，弹出如图 1.37 所示 Software Updates 对话框，提示必须重新启动 Eclipse 来应用刚刚添加到 Eclipse 中的功能插件。单击“是”按钮同意重新启动 Eclipse，单击“否”按钮不同意重新启动，单击 Apply Changes 按钮不重新启动而尝试应用新的功能插件（可能会功能不全）。单击“是”按钮，稍等片刻，Eclipse 会自动重启。

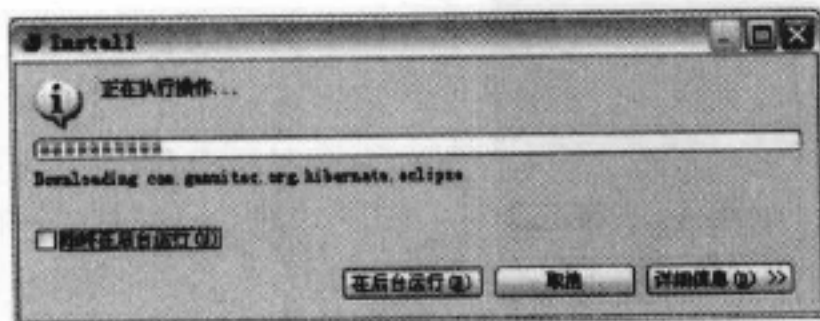


图 1.36 安装进度对话框

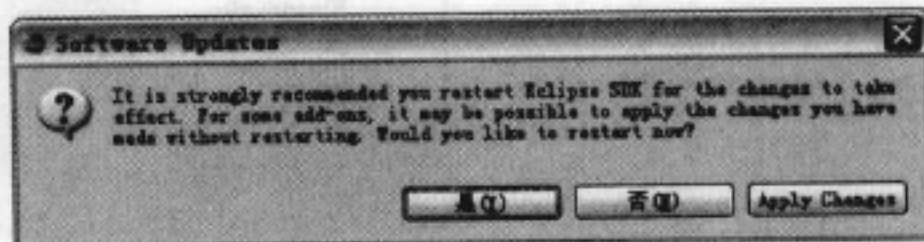


图 1.37 Software Updates 对话框

重新启动 Eclipse 之后,如果成功加载了 MyEclipse 插件,则在 Eclipse 的菜单栏中将出现 MyEclipse 菜单项,否则,应确认使用的 Eclipse 和 MyEclipse 版本是否匹配,本书使用的是 Eclipse 3.4 Classic+MyEclipse 7.0 版本。

1.5.3 配置 MyEclipse

Eclipse 的启动配置文件是 eclipse.ini,该文件定义了 Eclipse 启动时的内存大小。另外,在安装 MyEclipse 后还需要调整该设置,以使 MyEclipse 可以发挥所有功能。

在 Eclipse 所在文件夹中打开 eclipse.ini 文件,找到如下两行代码。

```
-Xms40m
-Xmx256m
```

其中,Xms 是设置 Java 虚拟机使用的内存大小,而 Xmx 是设置虚拟机使用的最大内存大小。如果读者的计算机内存在 1GB 以上,建议将上述两行代码修改如下。

```
-Xms128m
-Xmx512m
```

如果读者计算机内存低于 1GB,这样设置会导致启动错误,而无法启动 Eclipse,这时可以适当调小 Xmx 的设置。

1.5.4 修改 JSP 编辑器编码格式

MyEclipse 的 JSP 编辑器使用的默认编码格式为 ISO-8859-1,这种编码格式是不支持中文的,带有中文的 JSP 文件将无法保存,这显然不适合国内的网页开发,因此需要对 JSP 编辑器的编码设置进行修改。其步骤如下:

(1) 在 Eclipse 的菜单栏中选择“窗口”/“首选项”命令。

(2) 在弹出的“首选项”窗口中选择 MyEclipse Enterprise Workbench/JSP 节点,在右侧窗格“创建文件”栏中打开“编码”下拉列表框,从中选择 Chinese, National Standard 选项,如图 1.38 所示。单击“应用”按钮,然后单击“确定”按钮关闭“首选项”窗口。这样,以后新建的所有 JSP 文件都会使用中文编码。

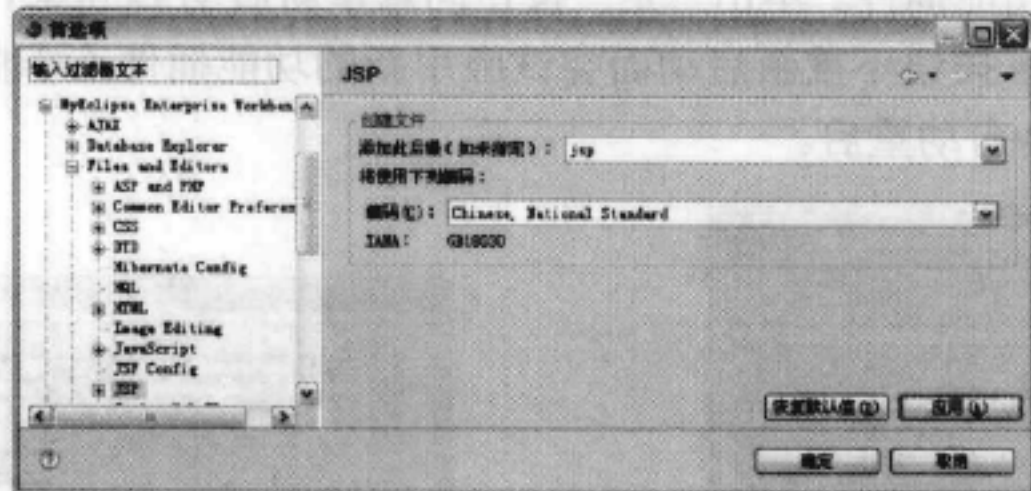


图 1.38 “首选项”窗口

1.5.5 设置 JSP 文件关联编辑器

MyEclipse 默认使用 JSP 设计器 MyEclipse Visual JSP Designer 来打开 JSP 文件。该设计器提供了页面设计功能，但是占用资源较高，有的计算机可能无法使用这个设计器。此时可以在 JSP 文件上单击鼠标右键，在弹出的快捷菜单中选择“打开方式”/MyEclipse JSP Editor 命令，这样即可使用 JSP 编辑器打开该文件。只不过它没有设计功能，但节约了很多系统资源。笔者推荐使用该编辑器，因为读者必须熟悉 HTML 和 JSP 语法。

利用上述方法虽然解决了 JSP 编辑器的问题，但每次都要在新的 JSP 文件上选择编辑器，非常麻烦。针对这个问题，本节就来介绍如何将 MyEclipse JSP Editor 设置为 JSP 的默认编辑器。其步骤如下：

(1) 在 Eclipse 的菜单栏中选择“窗口”/“首选项”命令。

(2) 在弹出的“首选项”窗口中选择“常规”/“编辑器”/“文件关联”节点，在右侧的“文件类型”列表框中选择“*.jsp”选项，然后在“相关联的编辑器”列表框中选择“MyEclipse JSP Editor (缺省值)”选项，单击“缺省值”按钮，即可将该编辑器设置为 JSP 文件默认的编辑器，如图 1.39 所示。最后单击“确定”按钮，关闭“首选项”窗口。

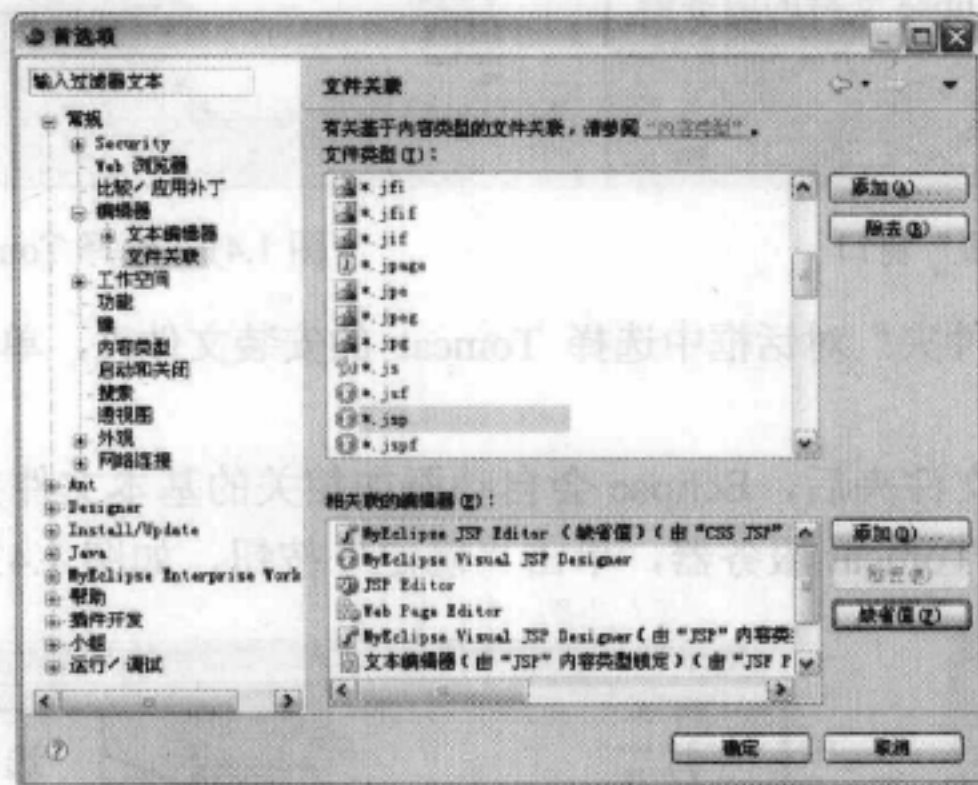


图 1.39 设置 JSP 文件关联编辑器

1.5.6 配置外置服务器

Tomcat 服务器是 Apache Jakarta 项目组开发的产品，它能够支持 Servlet 和 JSP 的最高版本，并且具有免费和跨平台等诸多特性，已经成为学习开发 Java Web 应用的首选，本书中的所有项目都使用了 Tomcat 作为 Web 服务器。MyEclipse 7.0 插件安装到 Eclipse 中后，自带了一个 Tomcat 6.0 服务器，我们可以直接使用这个 Tomcat 作为学习项目开发的容器。

由于每台计算机的配置不同，环境也不同，可能某些计算机无法运行这个内置的 Tomcat。这时就需要安装一个外置的 Web 服务器，也就是说，需要在 Eclipse 之外，在系统平台中安装一个服务器，然后在 MyEclipse 的“首选项”窗口中设置使用该服务器。

下面以 Tomcat 为例，介绍如何在 MyEclipse 中设置外部服务器。

Tomcat 可以从 <http://Tomcat.apache.org> 下载, 其安装方法非常简单, 采用默认安装方式即可, 这里不再多作介绍, 而是重点介绍在 MyEclipse 中如何设置外置 Web 服务器。其步骤如下:

(1) 启动 Eclipse, 选择“窗口”/“首选项”命令, 在弹出的“首选项”窗口中依次展开 MyEclipse Enterprise Workbench/Servers 节点, 其下的所有子节点均是 MyEclipse 7.0 支持的 Web 服务器, 如图 1.40 所示。

(2) 展开 Servers 节点下的 Tomcat 节点, 可以看到其中包含不同 Tomcat 版本的子节点(分别对应于不同的 Tomcat 版本), 在此选择 Tomcat 6.x 子节点来配置 Tomcat 6.0 版本的服务器, 然后单击右侧的“浏览”按钮选择 Tomcat 的安装文件夹, 如图 1.41 所示。

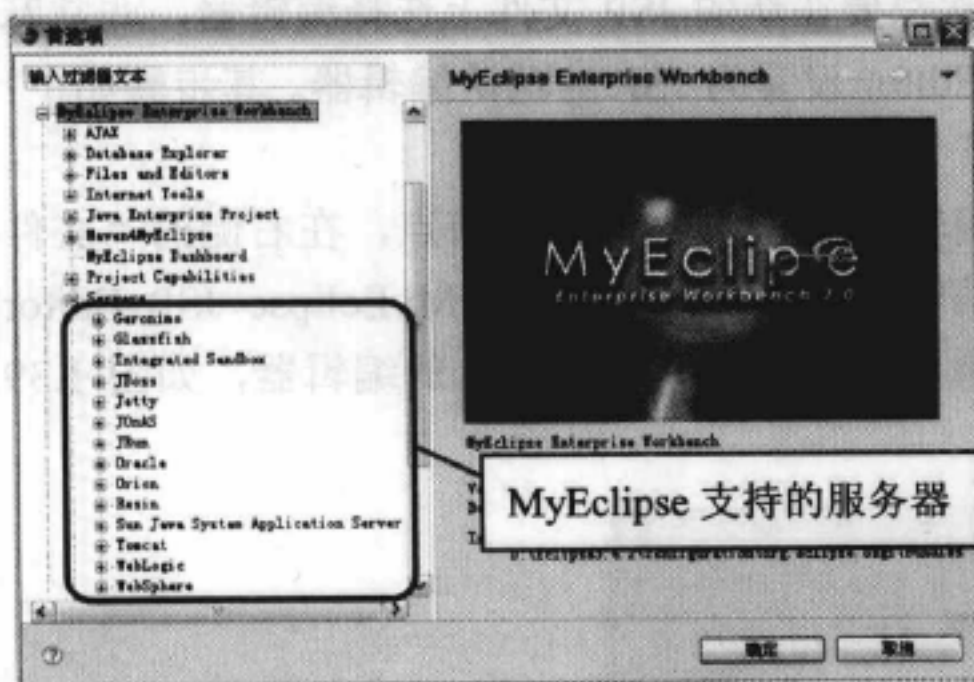


图 1.40 “首选项”窗口

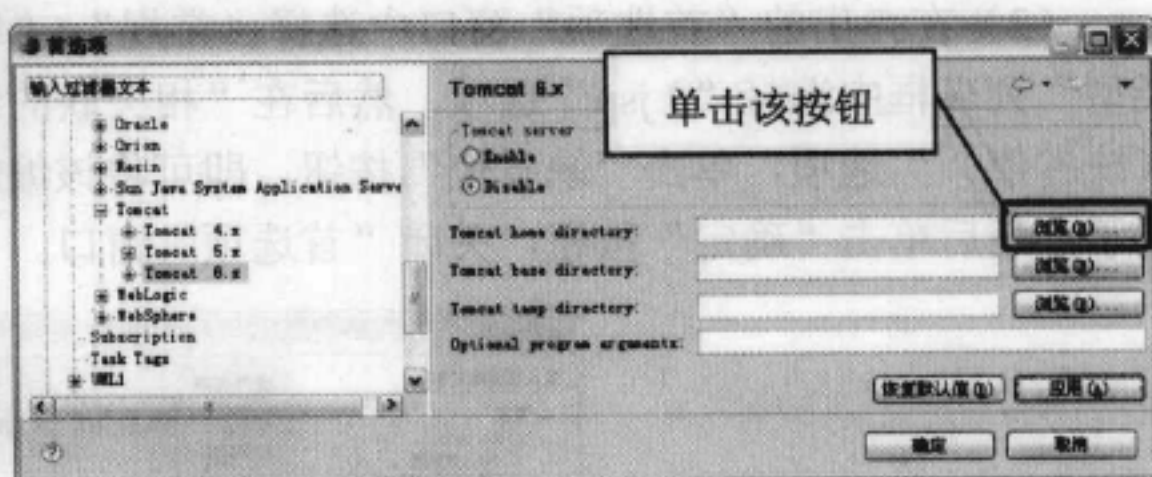


图 1.41 选择 Tomcat 6.0 并进行设置

(3) 在弹出的“浏览文件夹”对话框中选择 Tomcat 的安装文件夹, 单击“确定”按钮, 如图 1.42 所示。

(4) 选择 Tomcat 安装文件夹后, Eclipse 会自动添加相关的基本文件夹和临时文件夹位置, 然后选中 Enable 单选按钮激活该 Tomcat 服务器, 单击“确定”按钮, 如图 1.43 所示。



图 1.42 “浏览文件夹”对话框

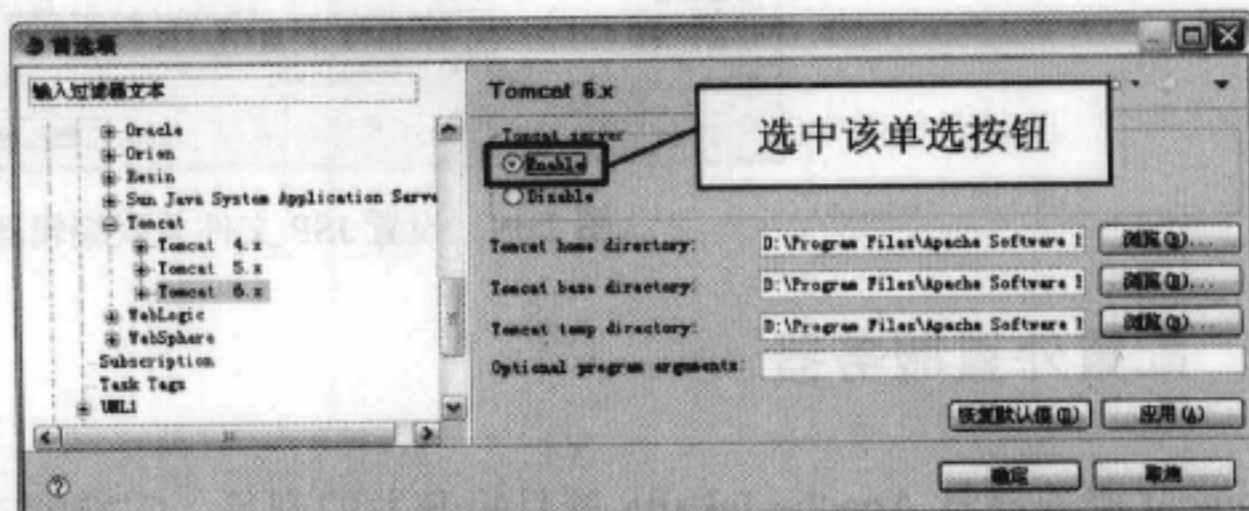


图 1.43 完成外部服务器的配置

1.6 开发第一个 JSP 程序

现在开发 Java Web 程序的环境已经搭建好了, 本节将介绍一个简单的 JSP 程序的开发过程(该 JSP 程序将在浏览器中输出“你好, 这是我的第一个 JSP 程序”), 让读者对 JSP 程序开发流程有一个基

本的认识。

1.6.1 编写 JSP 程序

例 1.01 使用向导创建一个简单的 JSP 程序。（实例位置：光盘\TM\Instances\1.01）

(1) 启动 Eclipse，在“包资源管理器”视图中，单击鼠标右键，在弹出的快捷菜单中选择“新建”/Web Project 命令，或者选择“文件”/“新建”/Web Project 命令。

(2) 在弹出的 New Web Project 对话框中的 Project Name 文本框中输入新建项目的名称，这里按照本书教学顺序，设置项目名称为 1.01，读者自己开发项目时可以任意设置项目名称。这时 Context root URL 文本框会自动使用和项目相同的名称。该文本框主要用于设置 Web 项目运行时在浏览器中的上下文名称，也就是 URL 地址路径中该项目的名称。因为这是第一个 JSP 程序，本实例为项目设置了一个有意义的上下文名称“/FirstJSP”。最后单击“完成”按钮，如图 1.44 所示。

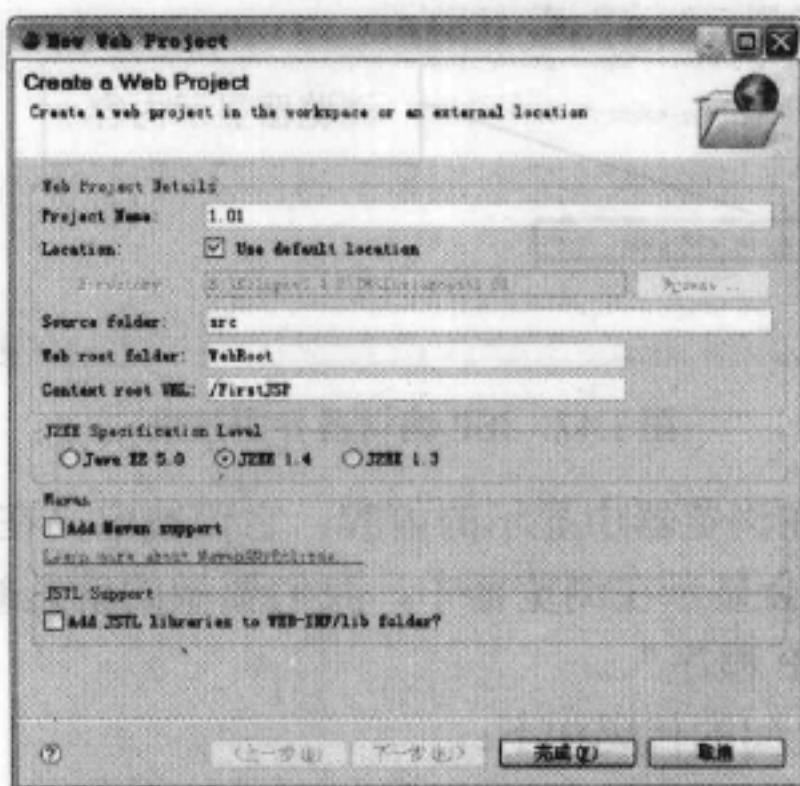


图 1.44 New Web Project 对话框

(3) 在新建 Web 项目时，会出现如图 1.45 所示的对话框，提示“项目的编译器级别将设置为 1.4，你当前默认的编译器级别是 6.0，是否要为该项目设置自定义的编译器级别？”。既然默认的编译器级别已经是 6.0，我们为什么要设置成 1.4 呢？那样会导致一些最新的语法不能够被使用。因此应单击“否”按钮，为项目设置 6.0 编译器级别。

(4) 在 Eclipse 启动之后，默认的透视图是“Java 透视图”，其中包含开发普通 Java 应用程序所需的视图。当我们创建一个 Web 项目时，将会需要有关 Web 开发相关的视图，所以 Eclipse 会弹出提示对话框，询问是否要切换到 MyEclipse Java Enterprise 透视图（它是开发 Web 项目的视图布局界面）。选中“记住我的决定”复选框，避免下次再出现该对话框，然后单击“是”按钮，如图 1.46 所示。

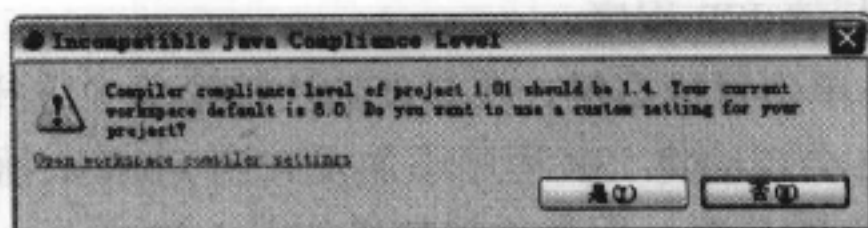


图 1.45 Incompatible Java Compliance Level 提示对话框

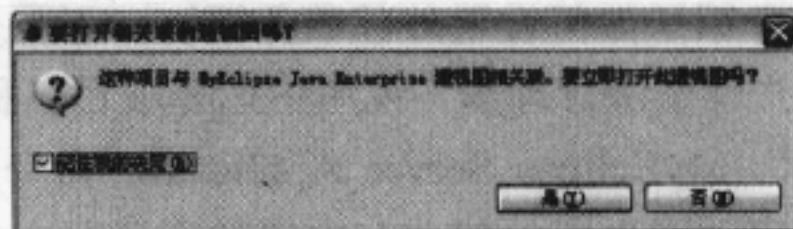


图 1.46 “要打开相关联的透视图吗？”提示对话框

(5) 展开项目的 WebRoot 文件夹，其中的 index.jsp 文件便是首页 JSP 文件，即 Web 项目默认首先被访问的页面文件。双击该文件，即可使用刚刚设置的 MyEclipse JSP Editor 编辑器打开该文件。该编辑器虽然没有设计功能，但是节约了很多系统资源。笔者推荐使用该编辑器，因为读者必须熟悉 HTML 和 JSP 语法。

打开 JSP 编辑器的界面效果如图 1.47 所示。如果读者的编辑器没有显示行号，可以在图中显示行号的位置单击鼠标右键，在弹出的快捷菜单中选择“显示行号”命令。



图 1.47 JSP 编辑器界面效果

index.jsp 文件的第 24 行是在浏览器中显示的内容，它们被定义在<body>和</body>标签中。由于所有定义在这对标签间的内容都会显示在浏览器中，我们要做的就是修改并编写自己的显示内容，例如：“你好，这是我的第一个 JSP 程序”。

修改<body>和</body>标签之间的代码如下：

```
</body>
    你好，这是我的第一个 JSP 程序<br>
    现在是时间：<%=new Date().toLocaleString() %>
</body>
```

上述代码还添加了当前时间作为网页的动态内容，以演示它与 HTML 静态页面的不同。

1.6.2 运行 JSP 程序

完成第一个 JSP 程序的编写后，还需要在浏览器中查看程序运行结果。运行一个 JSP 程序（也就是一个 JavaWeb 项目），需要有服务器的支持。MyEclipse 7.0 提供了一个内置的 Tomcat 服务器（前文中也介绍了如何设置外部服务器），下面介绍如何运行该 JSP 程序。

在“包资源管理器”视图中选择项目名称，即 1.01，单击鼠标右键，在弹出的快捷菜单中选择“运行方式”/MyEclipse Server Application 命令，MyEclipse 会自动将 JSP 程序添加发布到 Tomcat 服务器中，并启动服务器，然后在内置的浏览器中打开 JSP 程序的首页。程序运行效果如图 1.48 所示。

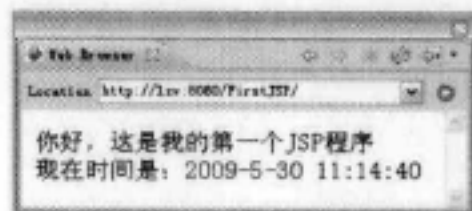


图 1.48 程序运行效果

1.7 JSP 常用资源

本节介绍一些常用的学习资源, 相信会对读者学习和开发项目有所帮助。

1.7.1 JSP 资源

JavaSE API 文档:	http://gceclub.sun.com.cn/java_docs/jdk6/html/zh_cn/api 。
JavaEE5API 文档:	http://java.sun.com/javaee/5/docs/api 。
JavaMail API 文档:	http://java.sun.com/products/javamail/javadocs 。
Struts 框架官方网站:	http://struts.apache.org 。
Hibernate 框架官方网站:	http://www.hibernate.org 。
Spring 框架官方网站:	http://www.springframework.org 。

1.7.2 Eclipse 资源

Eclipse 官方网站:	http://www.eclipse.org 。
Eclipse 插件中心:	http://www.eclipseplugincentral.com 。
SWT Designer 设计器插件:	http://www.swt-designer.com 。
MyEclipse 插件:	http://www.myeclipseide.com 。
Bable 多国语言包:	http://wwwlipse.org/babel 。

1.8 本章小结

本章首先简单描述了 JSP 技术以及 JSP 技术的特征; 然后介绍了 Java Web 开发常用的 Web 服务器、MySQL 数据库以及 GUI 工具的下载与安装、MyEclipse 的下载、安装与配置; 最后带领读者开发了一个简单的 JSP 程序, 使读者了解 Java Web 程序的开发流程, 同时还提供了学习与开发中常用的资源。通过本章的学习, 读者应该了解什么是 JSP、常用的 Web 服务器、如何安装数据库, 而最重要的是掌握搭建 Java Web 开发环境。正所谓“工欲善其事, 必先利其器”, 这是本章的重点, 读者应该熟练掌握, 为以后的学习和项目开发打下牢固的基础。



图 1.48 网络运行效果

1.7 JSP 常用资源

本节介绍一些常用的学习资源, 帮助读者学习和开发项目有侧帮助。

1.7.1 JSP 资源

- Java API 文档: <http://java.sun.com/docs/api/>
- Javadoc API 文档: <http://java.sun.com/javadoc/>
- JavaMail API 文档: <http://java.sun.com/products/javamail/>
- Struts 框架官方网站: <http://struts.apache.org>
- hibernate 框架官方网站: <http://www.hibernate.org>
- Spring 框架官方网站: <http://www.springframework.org>

1.7.2 Eclipse 资源


- Eclipse 官方网站: <http://www.eclipse.org>
- Eclipse 插件中心: <http://www.eclipseplugincentral.com>
- SWT Designer 设计器插件: <http://www.swt-designer.com>
- MyEclipse 插件: <http://www.myeclipse.com>
- Table 多国语言包: <http://www.table.org/pabel>

1.8 本章小结

本章首先简单介绍了 JSP 技术以及 JSP 技术的特征, 然后介绍了 Java Web 开发常用的 Web 服务器, MySQL 数据库以及 GUI 工具的下载与安装, MyEclipse 的下载、安装与配置; 最后介绍了开发了一个简单的 JSP 程序, 使读者了解 Java Web 程序的开发流程, 同时还提供了学习开发中常用的资源。通过本章的学习, 读者应该了解 JSP 常用的 Web 服务器, 如何安装数据库, 而最重要的就是搭建 Java Web 开发环境; 正所谓“工欲善其事, 必先利其器”, 这是本章的重点, 读者应该熟练掌握, 为以后的学习和项目开发打下坚实的基础。

第 2 章

掌握 JSP 语法

( 视频讲解：40 分钟)

几乎所有 Java Web 程序都是使用 JSP 技术作为显示层的。JSP 在 HTML 标记语言中嵌入各种 JSP 标签、注释和 Java 代码片段，能够动态生成 HTML 页面。本章将带领读者学习 JSP 技术的语法，奠定 Java Web 程序开发的基础。

通过阅读本章，您可以：

- » 了解 JSP 的基本构成
- » 了解指令标签
- » 了解脚本标签
- » 掌握 JSP 注释
- » 掌握 JSP 的动作标签

2.1 了解 JSP 的基本构成

例 2.01 了解 JSP 页面的基本构成。（实例位置：光盘\TM\Instances\2.01）

在开始学习 JSP 语法之前，不妨先来了解一下 JSP 页面的基本构成。JSP 页面主要由指令标签、HTML 标记语言、注释、嵌入 Java 代码、JSP 动作标签等 5 个元素组成，如图 2.1 所示。

```

1 <%@ page language="java" import="java.util.*" pageEncoding="GB18030"%>
2 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
3 <html>
4 <head>
5 <title>一个简单的JSP页面</title>
6 </head>
7 <body>
8 <!--HTML注释信息-->
9 <%
10     Date now = new Date();
11     String dateStr;
12     dateStr = String.format("%Y年%m月%d日", now, now, now);
13 >%
14     当前日期是: <%=dateStr%>
15 <br>
16 </body>
17 </html>
18

```

图 2.1 简单的 JSP 页面代码

☒ 指令标签

上述代码的第 1 行就是一个 JSP 的指令标签，它们通常位于文件的首位。

☒ HTML 标记语言

第 2~7 行、第 15~17 行都是 HTML 语言的代码，这些代码定义了网页内容的显示格式。

☒ 注释

第 8 行使用了 HTML 语言的注释格式，在 JSP 页面中还可以使用 JSP 的注释格式和嵌入 Java 代码的注释格式。

☒ 嵌入 Java 代码

在 JSP 页面中可以嵌入 Java 程序代码片段，这些 Java 代码被包含在<%>标签中，如上述的第 9~14 行就嵌入了 Java 代码片段。其中的代码可以看作是一个 Java 类的部分代码。

☒ JSP 动作标签

上述代码中没有编写动作标签。JSP 动作标签是 JSP 中标签的一种，它们都使用“JSP:”开头，例如“<jsp:forward>”标签可以将用户请求转发给另一个 JSP 页面或 Servlet 处理。在后面的章节中会对动作标签进行介绍。

2.2 指令标签

指令标签不会产生任何内容输出到网页中，主要用于定义整个 JSP 页面的相关信息，例如使用的语言、导入的类包、指定错误处理页面等。其语法格式如下：

```
<%@ directive attribute="value" attributeN="valueN" .....%>
```


- ☑ directive: 指令名称。
- ☑ attribute: 属性名称, 不同的指令包含不同的属性。
- ☑ value: 属性值, 为指定属性赋值的内容。

☞ 注意: 标签中的 `<%@` 和 `%>` 是完整的标记, 不能再添加空格, 但是标签中定义的各种属性之间以及与指令名之间可以有空格。

JSP 指令有 3 种, 分别是 `page`、`include` 和 `taglib`, 下面分别介绍。

2.2.1 page 指令

这是 JSP 页面最常用的指令, 用于定义整个 JSP 页面的相关属性, 这些属性在 JSP 被服务器解析成 Servlet 时会转换为相应的 Java 程序代码。page 指令的语法格式如下:

```
<%@ page attr1="value1" attr2="value2" .....%>
```

page 指令包含的属性有 15 个, 下面对一些常用的属性进行介绍。

1. language 属性

该属性用于设置 JSP 页面使用的语言, 目前只支持 Java 语言, 以后可能会支持其他语言, 如 C++、C# 等。该属性的默认值是 Java。

例如:

```
<%@ page language="java" %>
```

2. extends 属性

该属性用于设置 JSP 页面继承的 Java 类, 所有 JSP 页面在执行之前都会被服务器解析成 Servlet, 而 Servlet 是由 Java 类定义的, 所以 JSP 和 Servlet 都可以继承指定的父类。该属性并不常用, 而且有可能影响服务器的性能优化。

3. import 属性

该属性用于设置 JSP 导入的类包。JSP 页面可以嵌入 Java 代码片段, 这些 Java 代码在调用 API 时需要导入相应的类包。

例如:

```
<%@ page import="java.util.*" %>
```

4. pageEncoding 属性

该属性用于定义 JSP 页面的编码格式, 也就是指定文件编码。JSP 页面中的所有代码都使用该属性指定的字符集, 如果该属性值设置为 `iso-8859-1`, 那么这个 JSP 页面就不支持中文字符。通常我们设置编码格式为 `GBK`, 因为它可以显示简体中文和繁体中文, 而 MyEclipse 默认支持最新的 `GB18030` 编码格式, 并未提供 `GBK` 编码选项。通过第 1 章对 MyEclipse 环境的设置, 新创建的 JSP 文件都会使用“`GB18030`”编码格式。

例如：

```
<%@ page pageEncoding="GB18030"%>
```

5. contentType 属性

该属性用于设置 JSP 页面的 MIME 类型和字符编码，浏览器会据此显示网页内容。

例如：

```
<%@ page contentType="text/html; charset=ISO-8859-1"%>
```

如果将这个属性设置应用于 JSP 页面，那么浏览器在呈现该网页时会使用 ISO-8859-1 编码格式，而该编码格式不支持中文，所以页面会显示乱码，用户需要手动更改浏览器的显示编码才能看到中文内容，如图 2.2 所示。



图 2.2 错误的网页编码


6. session 属性

该属性指定 JSP 页面是否使用 HTTP 的 session 会话对象。其属性值是 boolean 类型，可选值为 true 和 false。默认值是 true，可以使用 session 会话对象；如果设置为 false，则当前 JSP 页面将无法使用 session 会话对象。

例如：

```
<%@ page session="false"%>
```

上述代码设置 JSP 页面不使用 session 对象，任何对 session 对象的引用都会发生错误。


 说明：session 是 JSP 的内置对象之一，在后面的章节将会介绍。

7. buffer 属性

该属性用于设置 JSP 的 out 输出对象使用的缓冲区大小，默认大小是 8KB，且单位只能使用 KB。建议程序开发人员使用 8 的倍数 16、32、64、128 等作为该属性的属性值。

例如：

```
<%@ page buffer="128kb"%>
```


 说明: out 对象是 JSP 的内置对象之一, 在后面的章节将会介绍。

8. autoFlush 属性

该属性用于设置 JSP 页面缓存满的时候, 是否自动刷新缓存。默认值为 true; 如果设置为 false, 则缓存被填满时将抛出异常。

例如:

```
<%@ page autoFlush="false"%>
```

上述代码取消了页面缓存的自动刷新。

9. info 属性

该属性用于设置 JSP 页面的相关信息, 该信息可以在 Servlet 接口的 `getServletInfo()` 方法中获取。

例如:

```
<%@ page info="这是一个登录页面, 是系统的入口"%>
```

10. isErrorPage 属性

通过该属性可以将当前 JSP 页面设置成错误处理页面来处理另一个 JSP 页面的错误, 也就是异常处理。这意味着当前 JSP 页面业务的改变。

例如:


```
<%@ page isELIgnored="true"%>
```

11. errorPage 属性

该属性用于指定处理当前 JSP 页面异常错误的另一个 JSP 页面, 指定的 JSP 错误处理页面必须设置 `isErrorPage` 属性为 true。errorPage 属性的属性值是一个 url 字符串。

例如:

```
<%@ page errorPage="error/loginErrorPage.jsp"%>
```

 注意: 如果设置该属性, 那么在 web.xml 文件中定义的任何错误页面都将被忽略, 而优先使用该属性定义的错误处理页面。

12. isELIgnored 属性

该属性用于定义 JSP 页面是否忽略 EL 表达式的使用。在 Servlet 2.4 版本中其默认值为 false, 即 JSP 支持 EL 表达式; 而在 Servlet 2.3 以前的版本中该属性的默认值为 true, 本书使用 Tomcat 6.0 服务器支持 Servlet 2.4, 所以默认值是 false, 可以直接使用 EL 表达式。

例如:

```
<%@ page isELIgnored="false"%>
```

2.2.2 include 指令

include 指令用于文件包含。该指令可以在 JSP 页面中包含另一个文件的内容，但是它仅支持静态包含，也就是说被包含文件中的所有内容都被原样包含到该 JSP 页面中；如果被包含文件中有代码，将不被执行。被包含的文件可以是一段 Java 代码、HTML 代码或者是另一个 JSP 页面。

例如：

```
<%@include file="validate.jsp" %>
```

上述代码将当前 JSP 文件中相同位置的 `validate.jsp` 文件包含进来。其中，`file` 属性用于指定被包含的文件，其值是当前 JSP 页面文件的相对 URL 路径。

下面举例演示 **include** 指令的应用。在当前 JSP 页面中包含 `date.jsp` 文件，而这个被包含的文件中定义了获取当前日期的 Java 代码，从而组成了当前页面显示日期的功能。这个实例主要用于演示 **include** 指令。

例 2.02 在当前页面中包含另一个 JSP 文件来显示当前日期。(实例位置：光盘\TM\Instances\2.02)

(1) 首先编辑 `date.jsp` 文件，程序代码如下。

```
<%@page pageEncoding="GB18030" %>
<%@page import="java.util.Date"%>
<%
    Date now = new Date();
    String dateStr;
    dateStr = String.format("%tY 年%tm 月%td 日", now, now, now);
%>
<%=dateStr%>
```

(2) 编辑 `index.jsp` 文件，它是本实例的首页文件，其中使用了 **include** 指令包含 `date.jsp` 文件到当前页面。被包含的 `date.jsp` 文件中的 Java 代码以静态方式导入到 `index.jsp` 文件，然后才被服务器编译执行。程序代码如下：

```
<%@ page language="java" import="java.util.*"
    contentType="text/html; charset=GB18030" pageEncoding="GB18030"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
        <title>include 指令演示</title>
    </head>
    <body>
        <!--HTML 注释信息-->
        当前日期是：
        <%@include file="date.jsp"%>
        <br>
    </body>
</html>
```


程序运行结果如图 2.3 所示（可以将地址栏中的访问地址复制到 IE 或其他浏览器中访问）。

date.jsp 文件将被包含在 index.jsp 文件中，所以文件中的 page 指令代码可以省略，在被包含到 index.jsp 文件中后会直接使用 index.jsp 文件的设置，但是为了在 Eclipse 编辑器中避免编译错误提示，本文添加了相关代码。

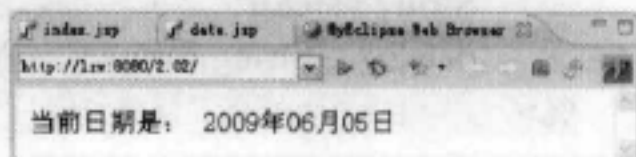


图 2.3 程序运行结果

说明：程序运行方法参见第 1 章中实例的运行步骤。

注意：被 include 指令包含的 JSP 页面中不要使用<html>和<body>标签，它们是 HTML 语言的结构标签，被包含进其他 JSP 页面会破坏页面格式。另外还要注意源文件和被包含文件中的变量和方法的名称不要冲突，因为它们最终会生成一个文件，重名将导致错误发生。

2.2.3 taglib 指令

该指令用于加载用户自定义标签，自定义标签将在后面章节进行讲解。使用该指令加载后的标签可以直接在 JSP 页面中使用。其语法格式如下：

```
<%@taglib prefix="fix" uri="tagUriorDir" %>
```

- ☒ prefix 属性：用于设置加载自定义标签的前缀。
- ☒ uri 属性：用于指定自定义标签的描述符文件位置。

例如：

```
<%@taglib prefix="view" uri="/WEB-INF/tags/view.tld" %>
```

2.3 嵌入 Java 代码

在 JSP 页面中可以嵌入 Java 的代码片段来完成业务处理，比如之前的实例在页面中输出当前日期，就是通过嵌入 Java 代码片段实现的。本节将介绍 JSP 嵌入 Java 代码的几种格式和用法。

2.3.1 代码片段

所谓代码片段就是在 JSP 页面中嵌入的 Java 代码，也有称为脚本段或脚本代码的。代码片段将在页面请求的处理期间被执行，可以通过 JSP 内置对象在页面输出内容、访问 session 会话、编写流程控制语句等。其语法格式如下：

```
<% 编写 Java 代码 %>
```

Java 代码片段被包含在“<%”和“%>”标记之间。可以编写单行或多行的 Java 代码，语句以“;”结尾，其编写格式与 Java 类代码格式相同。

例如：

```

<%
    Date now = new Date();
    String dateStr;
    dateStr = String.format("%tY 年%tm 月%td 日", now, now, now);
%>

```

上述代码在代码片段中创建 Date 对象，并生成格式化的日期字符串。

例 2.03 在代码片段中编写循环输出九九乘法表。（实例位置：光盘\TM\Instances\2.03）

```

<%@ page language="java" import="java.util.*" pageEncoding="GB18030"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
        <title>JSP 的代码片段</title>
    </head>
    <body>
        <%
            long startTime = System.nanoTime(); // 记录开始时间，单位纳秒
        %>
        输出九九乘法表
        <br>
        <%
            for (int i = 1; i <= 9; i++) { // 第一层循环
                for (int j = 1; j <= i; j++) { // 第二层循环
                    String str = j + "*" + i + "=" + j * i;
                    out.print(str + "&nbsp;"); // 使用空格格式化输出
                }
                out.println("<br>"); // HTML 换行
            }
            long time = System.nanoTime() - startTime;
        %>
        生成九九乘法表用时
        <%
            out.println(time / 1000); // 输出用时多少毫秒
        %>
        毫秒。
    </body>
</html>

```

程序运行结果如图 2.4 所示。

```

输出九九乘法表
1*1=1
1*2=2 2*2=4
1*3=3 2*3=6 3*3=9
1*4=4 2*4=8 3*4=12 4*4=16
1*5=5 2*5=10 3*5=15 4*5=20 5*5=25
1*6=6 2*6=12 3*6=18 4*6=24 5*6=30 6*6=36
1*7=7 2*7=14 3*7=21 4*7=28 5*7=35 6*7=42 7*7=49
1*8=8 2*8=16 3*8=24 4*8=32 5*8=40 6*8=48 7*8=56 8*8=64
1*9=9 2*9=18 3*9=27 4*9=36 5*9=45 6*9=54 7*9=63 8*9=72 9*9=81
生成九九乘法表用时 109 毫秒。

```

图 2.4 JSP 页面输出乘法表

2.3.2 声明

声明脚本用于在 JSP 页面中定义全局的（即整个 JSP 页面都需要引用的）成员变量或方法，它们可以被整个 JSP 页面访问，服务器执行时会把 JSP 页面转换为 Servlet 类，在该类中会把使用 JSP 声明脚本定义的变量和方法定义为类的成员。

（1）定义全局变量

例如：

```
<%! long startTime = System.nanoTime();%>
```

上述代码在 JSP 页面定义了全局变量 `startTime`，该全局变量可以在整个 JSP 页面使用。

（2）定义全局方法

例如：

```
<%!  
    int getMax(int a, int b) {  
        int max = a > b ? a : b;  
        return max;  
    }  
%>
```

2.3.3 JSP 表达式

JSP 表达式可以直接把 Java 的表达式结果输出到 JSP 页面中。表达式的最终运算结果将被转换为字符串类型，因为在网页中显示的文字都是字符串。JSP 表达式的语法格式如下：

```
<%= 表达式 %>
```

其中，表达式可以是任何 Java 语言的完整表达式。

例如：

```
圆周率是：<%=Math.PI %>
```

2.4 注 释

由于 JSP 页面由 HTML、JSP、Java 脚本等组成，所以在其中可以使用多种注释格式，本节将对这些注释的语法进行讲解。

2.4.1 HTML 注释

HTML 语言的注释不会被显示在网页中，但是在浏览器中选择查看网页源代码时，还是能够看到

注释信息的。

语法:

```
<!-- 注释文本 -->
```

例如:

```
<!-- 显示数据报表的表格 -->  
<table>  
.....  
</table>
```

上述代码为 HTML 的一个表格添加了注释信息,其他程序开发人员可以直接从注释中了解表格的用途,无须重新分析代码。在浏览器中查看网页代码时,上述代码将完整地显示,包括注释信息。

2.4.2 JSP 注释

程序注释通常用于帮助程序开发人员理解代码的用途,使用 HTML 注释可以为页面代码添加说明性的注释,但是在浏览器中查看网页源代码时将暴露这些注释信息;而如果使用 JSP 注释就不用担心出现这种情况了,因为 JSP 注释是被服务器编译执行的,不会发送到客户端。

语法:

```
<%-- 注释文本 --%>
```

例如:

```
<%-- 显示数据报表的表格 --%>  
<table>  
.....  
</table>
```

上述代码的注释信息不会被发送到客户端,那么在浏览器中查看网页源码时也就看不到注释内容。

2.4.3 动态注释

由于 HTML 注释对 JSP 嵌入的代码不起作用,因此可以利用它们的组合构成动态的 HTML 注释文本。

例如:

```
<!-- <%=new Date()%> -->
```

上述代码将当前日期和时间作为 HTML 注释文本。

2.4.4 代码注释

JSP 页面支持嵌入的 Java 代码,这些 Java 代码的语法和注释方法都和 Java 类的代码相同,因此也

就可以使用 Java 的代码注释格式。

例如：

```
<%//单行注释%>
<%/ *
    多行注释
    */
%>
<%/ **JavaDoc 注释, 用于成员注释 */ %>
```

2.5 JSP 动作标签

在 JSP 2.0 规范中提供了 20 个标准的使用 XML 语法写成的动作标签, 这些标签可用来实现特殊的功能, 例如转发用户请求、操作 JavaBean、包含其他文件等。

动作标签是在请求处理阶段按照在页面中出现的顺序被执行的。JSP 动作标签的优先级低于指令标签, 在 JSP 页面被执行时将首先进入翻译阶段, 程序会先查找页面中的指令标签, 将它们转换成 Servlet, 从而设置整个 JSP 页面。

动作标签遵循 XML 语法, 包括开始标签和结束标签。其通用的语法格式如下:

```
<标签名 属性 1="值 1" 属性 2="值 2".../>
```

或者

```
<标签名 属性 1="值 1" 属性 2="值 2"...>
    标签内容
</标签名>
```

本节将介绍 JSP 项目开发中常用的 JSP 动作标签。

2.5.1 <jsp:include>

这个动作标签可以将另外一个文件的内容包含到当前 JSP 页面中。被包含的文件内容可以是静态文本, 也可以是动态代码。其语法格式如下:

语法:

```
<jsp:include page="url" flush="false|true" />
```

或者:

```
<jsp:include page="url" flush="false|true">
    子标签
</jsp:include>
```

- ☑ **page**: 该属性用于指定被包含文件的相对路径。例如, “validate.jsp” 是将与当前 JSP 文件在同一文件夹中的 validate 文件包含到当前 JSP 页面中。

- ☑ **flush**: 可选参数, 用于设置是否刷新缓冲区。默认值为 `false`; 如果设置为 `true`, 则在当前页面输出使用了缓冲区的情况下, 将先刷新缓冲区, 然后再执行包含工作。

例如:

```
<jsp:include page="validate.jsp"/>
```

上述代码将 `validate.jsp` 文件内容包含到当前页面中。

- 🔊 **注意**: 被包含的 JSP 页面中不要使用 `<html>` 和 `<body>` 标签, 它们是 HTML 语言的结构标签, 被包含进其他 JSP 页面会破坏页面格式。另外要注意的一点是, 源文件和被包含文件中的变量和方法的名称不要冲突, 因为它们最终会生成一个文件, 重名会导致错误发生。

下面我们再来看看 `<jsp:include>` 与 `include` 指令的区别。

`<jsp:include>` 标签与 `include` 指令都拥有包含其他文件内容到当前 JSP 页面中的能力, 但是它们存在一定的区别, 具体体现在如下几点。

☑ 相对路径

`include` 指令使用 `file` 属性指定被包含的文件, 该属性值使用文件的相对路径指定被包含文件的位置, 而 `<jsp:include>` 标签以页面的相对路径来指定被包含的资源。

☑ 包含资源

`include` 指令包含的资源为静态, 例如 HTML、TXT 等; 如果将 JSP 的动态内容用 `include` 指令包含的话, 也会被当作静态资源包含到当前页面; 被包含资源与当前 JSP 页面是一个整体, 资源相对路径的解析在 JSP 页面转换为 Servlet 时发生, 如图 2.5 所示。

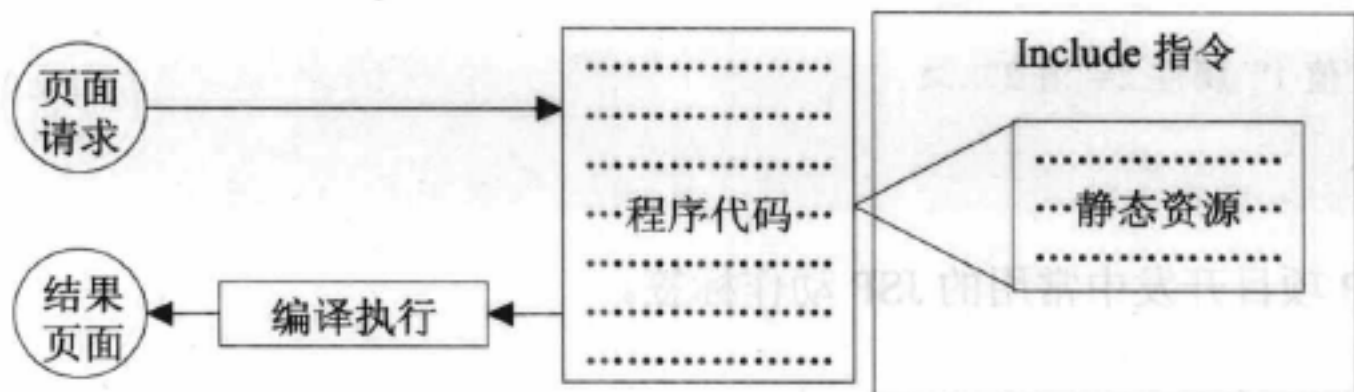


图 2.5 include 指令工作流程

`<jsp:include>` 标签包含 JSP 动态资源时, 资源相对路径的解析在请求处理时发生。当前页面和被包含的资源是两个独立的实体, 被包含的页面会对包含它的 JSP 页面中的请求对象进行处理, 然后将处理结果作为当前 JSP 页面的包含内容, 与当前页面内容一起发送到客户端, 如图 2.6 所示。

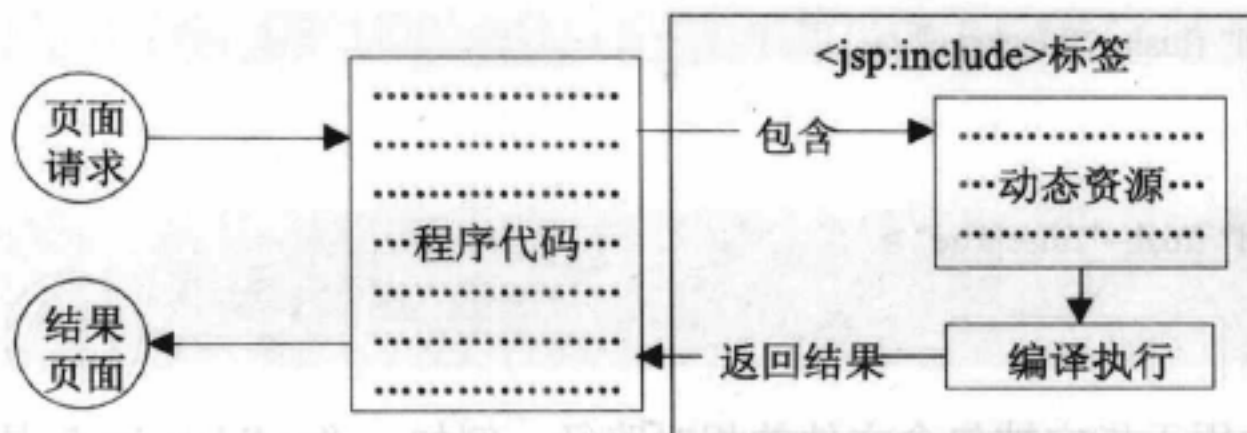
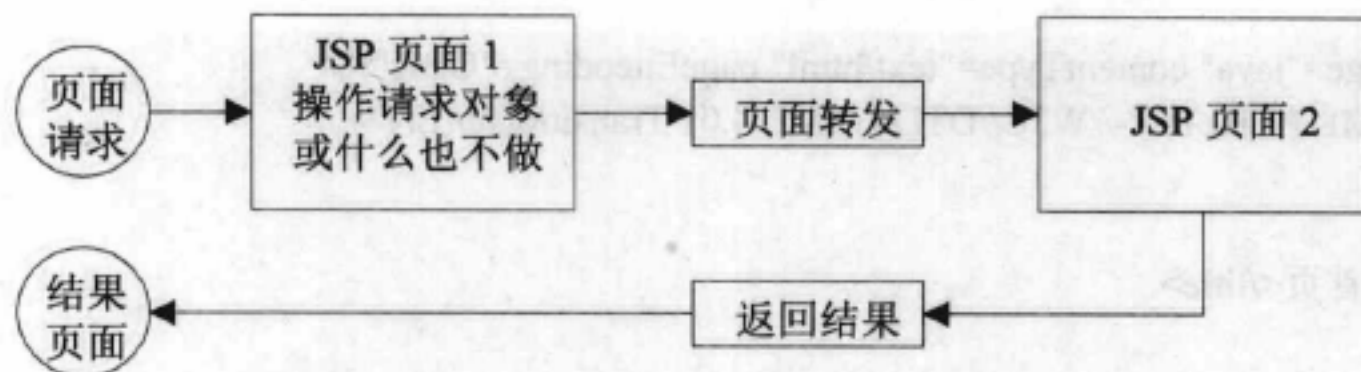


图 2.6 `<jsp:include>` 标签工作流程

2.5.2 <jsp:forward>

<jsp:forward>是请求转发标签。该标签可以将当前页面的请求转发给其他 Web 资源，例如另一个 JSP 页面、HTML 页面、Servlet 等；而当前页面可以不对请求进行处理，或者做些验证性的工作和其他工作。其工作原理如图 2.7 所示。



例 2.04 将首页请求转发到用户添加页面。（实例位置：光盘\TM\Instances\2.04）

（1）首先编写 addUser.jsp 文件，它是添加用户的页面。

```

<%@ page language="java" import="java.util.*" pageEncoding="GB18030"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>JSP 的 include 动作标签</title>
  </head>
  <body>
    <form action="index.jsp" method="post">
      <table align="center">
        <tr>
          <td align="center" colspan="2">
            <h3>添加用户</h3>
          </td>
        </tr>
        <tr>
          <td>姓名: </td>
          <td><input name="name" type="text"></td>
        </tr>
        <tr>
          <td>性别: </td>
          <td>
            <input name="sex" type="radio" value="男" checked="checked">
            <input name="sex" type="radio" value="女">
          </td>
        </tr>
        <tr>
          <td align="center" colspan="2">
            <input type="submit" value="添加">
            <input type="reset" value="重置">
          </td>
        </tr>
      </table>
    </form>
  </body>
</html>
  
```

```

        </tr>
      </table>
    </form>
  </body>
</html>

```

(2) 服务器默认运行的是 index.jsp 文件，它是 Web 程序的首页。在该文件中将请求转发给 adduser.jsp 页面文件，从而使 adduser.jsp 作为首先被访问的页面。

```

<%@ page language="java" contentType="text/html" pageEncoding="GBK"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>首页</title>
  </head>
  <body>
    <jsp:forward page="addUser.jsp"/>
  </body>
</html>

```

实例运行结果如图 2.8 所示。

图 2.8 实例运行结果

2.5.3 <jsp:param>

该标签可以作为其他标签的子标签，为其他标签传递参数。其语法格式如下：

语法：

```
<jsp:param name="paramName" value="paramValue" />
```

☑ name 属性：用于指定参数名称。

☑ value 属性：用于设置对应的参数值。

例如：

```

<jsp:forward page="addUser.jsp">
  <jsp:param name="userName" value="mingri"/>
</jsp:forward>

```

上述代码在转发请求到 adduser.jsp 页面的同时，传递了参数 userName，其参数值为“mingri”。

2.5.4 操作 JavaBean 的动作标签

<jsp:useBean>、<jsp:setProperty>和<jsp:getProperty>这 3 个动作标签用于操作 JavaBean 对象，有关

JavaBean 的知识将在第 5 章中进行讲解,所以这 3 个操作 JavaBean 的标签也一同放在第 5 章进行介绍。

2.6 实 战

既然 JSP 页面可以嵌入 Java 代码片段(或称为脚本),那么在 JSP 页面中就可以通过 JDBC 实现数据库连接与操作。本节将举例说明如何在 JSP 页面中通过 JDBC 读取数据库并显示到页面中。

例 2.05 在 JSP 页面中通过 JDBC 连接数据库并将数据显示在页面表格中。(实例位置:光盘 \TM\Instances\2.05)

```
<%@ page language="java" pageEncoding="GB18030"%>
<%@page import="java.sql.Connection"%>
<%@page import="java.sql.*"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>JSP 读取数据库</title>
  </head>
  <body>
    <table border="1" align="center">
      <tr>
        <th>书号</th>
        <th>书名</th>
        <th>作者</th>
        <th>出版社</th>
        <th>单价</th>
        <th>出版日期</th>
      </tr>
      <%
        String driverClass = "com.mysql.jdbc.Driver";
        String url = "jdbc:mysql://localhost:3306/db_Database02";
        String user = "root";
        String password = "111";
        Connection conn;
        try {
          Class.forName(driverClass).newInstance();
          conn = DriverManager.getConnection(url, user, password);
          Statement stmt = conn.createStatement();
          String sql = "SELECT * FROM tb_books";
          ResultSet rs = stmt.executeQuery(sql);
          while (rs.next()) {
            %>
            <tr>
              <td><%=rs.getString("ISBN")%></td>
              <td><%=rs.getString("bookName")%></td>
              <td><%=rs.getString("publishing")%></td>
              <td><%=rs.getString("writer")%></td>
```



```

        <td><%=rs.getString("price")%></td>
        <td><%=rs.getString("date")%></td>
    </tr>
    <%
    }
    } catch (Exception ex) {
        ex.printStackTrace();
    }
    %>
</table>
</body>
</html>

```

实例运行结果如图 2.9 所示。

书号	书名	作者	出版社	单价	出版日期
7-115-16380-6	Spring应用开发完全手册	李钟尉	人民邮电出版社	59	2007-09-01
7-111-15984-5	JSP工程应用与项目实践	白伟明	机械工业出版社	38	2005-02-01
7-111-16490-4	Visual Basic 信息系统开发实例精选	高春艳	机械工业出版社	44	2005-07-01
7-111-16617-5	ASP 信息系统开发实例精选	王国辉	机械工业出版社	45	2005-07-01
7-115-14545-8	Visual Basic 数据库系统开发完全手册	明日科技	人民邮电出版社	52	2006-03-01
7-115-14564-4	Visual C++ 数据库系统开发完全手册	明日科技	人民邮电出版社	52	2006-03-01
7-115-14873-2	ASP程序开发范例宝典	孙明丽	人民邮电出版社	82	2006-07-01

图 2.9 实例运行结果

2.7 本章小结


本章带领读者了解了 JSP 的基本构成,并详细介绍了构成 JSP 页面的各个部分——指令标签、HTML 代码、嵌入 Java 代码、注释和 JSP 动作标签(其中 HTML 代码不在本书讲解范围内,没有介绍)。通过本章的学习,读者应该对 JSP 页面的内容结构有所了解,配合后面章节介绍的 JSP 内置对象,可以开发完整的 JSP 应用。

2.8 实战练习

1. 在 JSP 页面中输出字符串。(答案位置:光盘\TM\Instances\2.06)
2. 在 JSP 页面中输出完整的时间,格式为:“年 月 日 时:分:秒”。(答案位置:光盘\TM\Instances\2.07)
3. 计算 5 的阶乘并在 JSP 页面中输出。(答案位置:光盘\TM\Instances\2.08)
4. 在 JSP 页面中输出字符“*”组成的金字塔。(答案位置:光盘\TM\Instances\2.09)

第 3 章

JSP 内置对象

( 视频讲解：82 分钟)

为了简化 Web 程序的开发过程，JSP 提供了由容器实现和管理的内置对象，也可以将其称之为固有对象、隐含对象。这些内置对象在所有的 JSP 页面中都可以直接使用，不需要 JSP 页面编写者来实例化。JSP 页面的内置对象被广泛地用于 JSP 的各种操作中，例如应用 request 对象来处理请求、应用 out 对象向页面输出信息、应用 session 对象来保存数据等。熟练地掌握和应用这些内置对象，对于 Java Web 程序开发人员来说是至关重要的。本章将向读者详细地介绍 JSP 内置对象。

通过阅读本章，您可以：

- » 掌握 request 对象的应用及常用方法
- » 掌握 response 对象的应用及常用方法
- » 掌握 session 对象的应用及常用方法
- » 掌握 application 对象的应用及常用方法
- » 掌握 out 对象的应用及常用方法
- » 了解 pageContext、config、page、exception 对象的应用

3.1 JSP 内置对象的概述

JSP 中采用 Java 语言作为脚本编程语言, 这样一来使系统具有了强大的对象处理能力, 并且可以动态创建 Web 页面内容。但 Java 语法在使用一个对象之前都要先将这个对象进行实例化, 比较繁琐。为了简化开发 JSP 提供了一些内置对象, 这也是 JSP 语法结构中的独特语句变量, 又被称为 JSP 预定义变量。它们都是由系统容器实现和管理的, 在 JSP 页面中不需要定义, 可以直接使用。

在 JSP 中一共预先定义了 9 个这样的对象, 分别为 request、response、session、application、out、pageContext、config、page、exception。本章将分别介绍这些内置对象及其常用方法。

3.2 request 对象

 视频讲解: 光盘\TM\Video\3\request 对象.exe

request 对象是 javax.servlet.http.HttpServletRequest 类型的对象。该对象代表了客户端的请求信息, 主要用于接收通过 HTTP 协议传送到服务器端的数据 (包括头信息、系统信息、请求方式以及请求参数等)。request 对象的作用域为一次请求。

3.2.1 获取请求参数值

在一个请求中, 可以通过使用 “?” 的方式来传递参数, 然后通过 request 对象的 getParameter() 方法来获取参数的值。例如:

```
String id = request.getParameter("id");
```

上面的代码使用 getParameter() 方法从 request 对象中获取参数 id 的值, 如果 request 对象中不存在此参数, 那么该方法将返回 null。

例 3.01 使用 request 对象获取请求参数值。(实例位置: 光盘\TM\Instances\3.01)

首先在 Web 项目中创建 index.jsp 页面, 在其中加入一个超链接按钮用来请求 show.jsp 页面, 并在请求后增加一个参数 id。关键代码如下:

```
<body>
  <a href="show.jsp?id=001">获取请求参数的值</a>
</body>
```

然后新建 show.jsp 页面, 在其中通过 getParameter() 方法来获取 id 参数与 name 参数的值, 并将其输出到页面中。关键代码如下:

```
<body>
  id 参数的值为: <%=request.getParameter("id") %><br>
```



```
name 参数的值为: <%=request.getParameter("name") %>
</body>
```

在上面的代码中我们同时将 id 参数与 name 参数的值显示在页面中,但是在请求中只传递了 id 参数,并没有传递 name 参数,所以 id 参数的值被正常显示出来,而 name 参数的值则显示为 null,运行结果如图 3.1 所示。

```
id参数的值为: 001
name参数的值为: null
```

图 3.1 程序运行结果

3.2.2 解决中文乱码

在上面的代码中我们为 id 参数传递了一个字符串类型的值“001”,如果将这个参数的值更改为中文,则在 show.jsp 就会发生大家都不愿意看到的问题——在显示参数值时中文内容变成了乱码。这是因为请求参数的文字编码方式与页面中的不一致所造成的,所有的 request 请求都是 iso-8859-1 的,而在此页面采用的是 GBK 的编码方式。要解决此问题,只要将获取到的数据通过 String 的构造方法使用指定的编码类型重新构造一个 String 对象即可正确地显示出中文信息。

例 3.02 解决中文乱码。(实例位置: 光盘\TM\Instances\3.02)

创建 index.jsp 页面,在其中加入一个超链接,并在该超链接中传递两个参数,分别为 name 与 sex,其值全部为中文。关键代码如下:

```
<body>
  <a href="show.jsp?name=张三&sex=男">解决中文乱码</a>
</body>
```

接下来创建 show.jsp 页面,在其中将第一个参数 name 的值进行编码转换,将第二个参数 sex 的值直接显示在页面中,从而比较一下效果。关键代码如下:

```
<body>
  name 参数的值为: <%=new String(request.getParameter("name").getBytes("iso-8859-1"),"gbk") %>
  sex 参数的值为: <%=request.getParameter("sex") %>
</body>
```

运行本实例后,可以发现 name 参数的值被正常显示出来,而 sex 参数的值则被显示成了乱码,如图 3.2 所示。

```
name参数的值为: 张三
sex参数的值为: ??
```

图 3.2 实例运行结果

3.2.3 获取 Form 表单的信息

除了获取请求参数中传递的值之外,我们还可以使用 request 对象获取从表单中提交过来的信息。

在一个表单中会有不同的标签元素,对于文本元素、单选按钮、单选下拉列表框都可以使用 `getParameter()` 方法来获取其具体的值,但对于复选框以及多选列表框被选定的内容就要使用 `getParameterValues()` 方法来获取了,该方法会返回一个字符串数组,通过循环遍历这个数组就可以得到用户选定的所有内容。

例 3.03 获取 Form 表单信息。(实例位置: 光盘\TM\Instances\3.03)

创建 `index.jsp` 页面文件,在该页面中创建一个 form 表单,在表单中分别加入文本框、下拉列表框、单选按钮和复选框。关键代码如下:

```
<form action="show.jsp" method="post">
  <ul style="list-style: none; line-height: 30px">
    <li>输入用户姓名: <input type="text" name="name" /><br /></li>
    <li>选择性别:
      <input name="sex" type="radio" value="男" />男
      <input name="sex" type="radio" value="女" />女
    </li>
    <li>
      选择密码提示问题:
      <select name="question">
        <option value="母亲生日">母亲生日</option>
        <option value="宠物名称">宠物名称</option>
        <option value="电脑配置">电脑配置</option>
      </select>
    </li>
    <li>请输入问题答案: <input type="text" name="key" /></li>
    <li>
      请选择个人爱好:
      <div style="width: 400px">
        <input name="like" type="checkbox" value="唱歌跳舞" />唱歌跳舞
        <input name="like" type="checkbox" value="上网冲浪" />上网冲浪
        <input name="like" type="checkbox" value="户外登山" />户外登山<br />
        <input name="like" type="checkbox" value="体育运动" />体育运动
        <input name="like" type="checkbox" value="读书看报" />读书看报
        <input name="like" type="checkbox" value="欣赏电影" />欣赏电影
      </div>
    </li>
    <li><input type="submit" value="提交" /></li>
  </ul>
</form>
```

页面运行结果如图 3.3 所示。

输入用户姓名:

选择性别: ☐ 男 ☐ 女

选择密码提示问题: 母亲生日

请输入问题答案:

请选择个人爱好:

☐ 唱歌跳舞 ☐ 上网冲浪 ☐ 户外登山

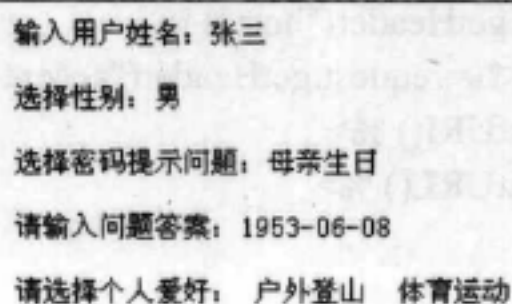
☐ 体育运动 ☐ 读书看报 ☐ 欣赏电影

图 3.3 页面运行结果

接下来编写 show.jsp 页面文件, 该页面是用来处理请求的, 在其中分别使用 `getParameter()` 方法与 `getParameterValues()` 方法将用户提交的表单信息显示在页面中。关键代码如下:


```
<ul style="list-style:none; line-height:30px">
  <li>输入用户姓名: <%=new String(request.getParameter("name").getBytes("ISO8859_1"),"GBK") %></li>
  <li>选择性别: <%=new String(request.getParameter("sex").getBytes("ISO8859_1"),"GBK") %></li>
  <li>选择密码提示问题: <%=new String(request.getParameter("question").getBytes("ISO8859_1"),"GBK") %>
</li>
  <li>请输入问题答案: <%=new String(request.getParameter("key").getBytes("ISO8859_1"),"GBK") %></li>
  <li>
    请选择个人爱好:
    <%
      String[] like =request.getParameterValues("like");
      for(int i =0;i<like.length;i++){
    %>
    <%= new String(like[i].getBytes("ISO8859_1"),"GBK")+ "&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;" %>
    <% }
    %>
  </li>
</ul>
```

show.jsp 页面运行结果如图 3.4 所示。



```
输入用户姓名: 张三
选择性别: 男
选择密码提示问题: 母亲生日
请输入问题答案: 1953-06-08
请选择个人爱好: 户外登山 体育运动
```

图 3.4 show.jsp 页面运行结果

 说明: 如果想要获得所有的参数名称可以使用 `getParameterNames()` 方法, 该方法返回一个 `Enumeration` 类型值。

3.2.4 获取请求客户端信息

在 `request` 对象中通过相应的方法 (如表 3.1 所示) 还可以获取到客户端的相关信息, 如 HTTP 报头信息、客户信息提交方式、客户端主机 IP 地址、端口号等。

表 3.1 request 获取客户端信息方法说明

方 法	返 回 值	说 明
<code>getHeader(String name)</code>	<code>String</code>	返回指定名称的 HTTP 头信息
<code>getMethod()</code>	<code>String</code>	获取客户端向服务器发送请求的方法
<code>getContextPath()</code>	<code>String</code>	返回请求路径
<code>getProtocol()</code>	<code>String</code>	返回请求使用的协议
<code>getRemoteAddr()</code>	<code>String</code>	返回客户端 IP 地址

续表

方 法	返 回 值	说 明
getRemoteHost()	String	返回客户端主机名称
getRemotePort()	int	返回客户端发出请求的端口号
getServletPath()	String	返回接受客户提交信息的页面
getRequestURI()	String	返回部分客户端请求的地址, 不包括请求的参数
getRequestURL()	StringBuffer	返回客户端请求地址

例 3.04 获取请求信息。(实例位置: 光盘\TM\Instances\3.04)

本实例通过上面介绍的方法演示如何使用 request 对象获取请求客户端信息。关键代码如下:

```
<ul style="line-height:24px">
  <li>客户使用的协议: <%=request.getProtocol() %>
  <li>客户端发送请求的方法: <%=request.getMethod() %>
  <li>客户端请求路径: <%=request.getContextPath() %>
  <li>客户机 IP 地址: <%=request.getRemoteAddr() %>
  <li>客户机名称: <%=request.getRemoteHost() %>
  <li>客户机请求端口号: <%=request.getRemotePort() %>
  <li>接收客户信息的页面: <%=request.getServletPath() %>
  <li>获取报头中 User-Agent 值: <%=request.getHeader("user-agent") %>
  <li>获取报头中 accept 值: <%=request.getHeader("accept") %>
  <li>获取报头中 Host 值: <%=request.getHeader("host") %>
  <li>获取报头中 accept-encoding 值: <%=request.getHeader("accept-encoding") %>
  <li>获取 URI: <%= request.getRequestURI() %>
  <li>获取 URL: <%=request.getRequestURL() %>
</ul>
```

上面的代码运行结果如图 3.5 所示, 可以看到请求客户端的信息以及报头中的部分信息都已经被显示在页面上了。

```
• 客户使用的协议: HTTP/1.1
• 客户端发送请求的方法: GET
• 客户端请求路径: /3.04
• 客户机IP地址: 192.168.1.42
• 客户机名称: 192.168.1.42
• 客户机请求端口号: 1047
• 接收客户信息的页面: /index.jsp
• 获取报头中User-Agent值: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; SV1:
.NET CLR 1.1.4322; .NET CLR 2.0.50727; InfoPath.2)
• 获取报头中accept值: */*
• 获取报头中Host值: 192.168.1.42:8080
• 获取报头中accept-encoding值: gzip, deflate
• 获取URI: /3.04/
• 获取URL: http://192.168.1.42:8080/3.04/
```

图 3.5 客户端信息

3.2.5 在作用域中管理属性

通过使用 setAttribute()方法可以在 request 对象的属性列表中添加一个属性, 然后在 request 对象的

作用域范围内通过使用 `getAttribute()` 方法将其属性取出；此外，还可使用 `removeAttribute()` 方法将一个属性删除掉。

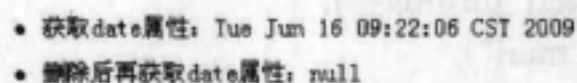
例 3.05 管理 request 对象属性。（实例位置：光盘\TM\Instances\3.05）

本实例首先将 `date` 属性加入到 `request` 属性列表中，然后输出这个属性的值；接下来使用 `removeAttribute()` 方法将 `date` 属性删除，最后再次输出 `date` 属性。关键代码如下：

```
<%
    request.setAttribute("date",new Date()); //添加一个属性
%>
<ul style="line-height: 24px;">
    <li>获取 date 属性: <%=request.getAttribute("date") %></li>
    <!-- 将属性删除 -->
    <%=request.removeAttribute("date"); %>
    <li>删除后再获取 date 属性: <%=request.getAttribute("date") %></li>
</ul>
```

注意：request 对象的作用域为一次请求，超出作用域后属性列表中的属性即会失效。

程序运行结果如图 3.6 所示，第一次正确输出了 `date` 的值；在将 `date` 属性删除以后，再次输出时 `date` 的值为 `null`。



- 获取date属性: Tue Jun 16 09:22:06 CST 2009
- 删除后再获取date属性: null

图 3.6 管理属性

3.2.6 cookie 管理

cookie 是小段的文本信息，通过使用 cookie 可以标识用户身份、记录用户名及密码、跟踪重复用户。cookie 在服务器端生成并发送给浏览器，浏览器将 cookie 的 `key/value` 保存到某个指定的目录中，服务器的名称与值可以由服务器端定义。

通过 cookie 的 `getCookies()` 方法可以获取到所有的 cookie 对象集合，然后通过 cookie 对象的 `getName()` 方法获取到指定名称的 cookie，再通过 `getValue()` 方法即可获取到 cookie 对象的值。另外，将一个 cookie 对象发送到客户端使用了 `response` 对象的 `addCookie()` 方法。

例 3.06 管理 cookie。（实例位置：光盘\TM\Instances\3.06）

首先创建 `index.jsp` 页面文件，在其中创建 form 表单，用于让用户输入信息；并且从 `request` 对象中获取 cookie，判断是否含有此服务器发送过的 cookie。如果没有，则说明该用户第一次访问本站；如果有，则直接将值读取出来，并赋给对应的表单。关键代码如下：

```
<%
    String welcome = "第一次访问";
    String[] info = new String[]{"","",""};
    Cookie[] cook = request.getCookies();
    if(cook!=null){
        for(int i=0;i<cook.length;i++){
```



```
<%  
    String name = request.getParameter("name");  
    String birthday = request.getParameter("birthday");  
    String mail = request.getParameter("mail");  
    Cookie myCook = new Cookie("mrCookInfo",name+"#"+"birthday"+"#"+"mail");  
    myCook.setMaxAge(60*60*24*365);           //设置 cookie 有效期  
    response.addCookie(myCook);  
%>  
表单提交成功  
<ul style="line-height: 24px">  
    <li>姓名: <%= name %>  
    <li>出生日期: <%= birthday %>  
    <li>电子邮箱: <%= mail %>  
    <li><a href="index.jsp">返回</a>  
</ul>
```

第一次访问

- 姓 名:
- 出生日期:
- 邮籍地址:
-

zhangsan: 欢迎回来!


- 姓 名: zhangsan
- 出生日期: 1954-06-08
- 邮箱地址: zhang@mr***.com
-

46

3.2.7 获取浏览器使用的语言

在一个支持国际化的站点中，一般都是根据其浏览器设定的语言来显示对应内容。我们只需通过 `getLocale()` 方法就可以很轻松地获取到客户端浏览器的语言类型。

3.3 response 对象

 视频讲解：光盘\TM\Video\3\response 对象.exe

`response` 代表的是对客户端的响应，主要是将 JSP 容器处理过的对象传回到客户端。`response` 对象也具有作用域，它只在 JSP 页面内有效。`response` 对象的常用方法如表 3.2 所示。

表 3.2 response 对象的常用方法


方 法	返 回 值	说 明
<code>addHeader(String name,String value)</code>	<code>void</code>	添加 HTTP 文件头，如果同名的头存在，则覆盖
<code>setHeader(String name,String value)</code>	<code>void</code>	设定指定名称的文件并头的值，如果存在则覆盖
<code>addCookie(Cookie cookie)</code>	<code>void</code>	向客户端添加一个 cookie 对象
<code>sendError(int sc,String msg)</code>	<code>void</code>	向客户端发送错误信息。例如：404 网页找不到
<code>sendRedirect(String location)</code>	<code>void</code>	发送请求到另一个指定位置
<code>getOutputStream()</code>	<code>ServletOutputStream</code>	获取客户端输出流对象
<code>setBufferSize(int size)</code>	<code>void</code>	设置缓冲区大小

3.3.1 重定向网页

重定向是通过使用 `sendRedirect()` 方法，将响应发送到另一个指定的位置进行处理。重定向可以将地址重新定向到不同的主机上，在客户端浏览器上将会得到跳转的地址，并重新发送请求链接。用户可以从浏览器的地址栏中看到跳转后的地址。进行重定向操作后，`request` 中的属性全部失效，并且进入一个新的 `request` 对象的作用域。

例如，使用该方法重定向到明日图书网。

```
response.sendRedirect("www.mingribook.com");
```

 注意：在 JSP 页面中使用该方法的时候前面不要有 HTML 代码，并且在重定向操作之后紧跟一个 `return`，因为重定向之后下面的代码已经没有意义了，并且还可能产生错误。

3.3.2 处理 HTTP 文件头


`setHeader()` 方法通过两个参数——头名称与参数值的方式来设置 HTTP 文件头。

例如，设置网页每 5 秒自动刷新一次。

```
response.setHeader("refresh","5");
```

例如，设置 2 秒钟后自动跳转至指定的页面。

```
response.setHeader("refresh","2;URL=welcome.jsp");
```

 注意：refresh 参数并不是 HTTP 1.1 规范中的标准参数，但 IE 与 Netscape 浏览器都支持该参数。

例如，设置响应类型。

```
response.setContentType("text/html");
```

3.3.3 设置输出缓冲

通常情况下，服务器要输出到客户端的内容不会直接写到客户端，而是先写到一个输出缓冲区；只有在以下的 3 种情况下，才会把缓冲区的内容写到客户端。

- ☒ JSP 页面的输出信息已经全部写入到了缓冲区。
- ☒ 缓冲区已满。
- ☒ 在 JSP 页面中调用了 flushbuffer() 方法或 out 对象的 flush() 方法。

使用 response 对象的 setBufferSize() 方法可以设置缓冲区的大小。例如，设置缓冲区大小为 0KB，即不缓冲。

```
response.setBufferSize(0);
```

还可以使用 isCommitted() 方法来检测服务器端是否已经把数据写入到了客户端。

3.4 session 对象

 视频讲解：光盘\TM\Video\3\session 对象.exe

session 对象是由服务器自动创建的与用户请求相关的对象。服务器为每个用户都生成一个 session 对象，用于保存该用户的信息，跟踪用户的操作状态。session 对象内部使用 Map 类来保存数据，因此保存数据的格式为“key/value”。session 对象的 value 可以是复杂的对象类型，而不仅仅局限于字符串类型。session 中的常用方法如表 3.3 所示。

表 3.3 session 对象常用方法

方 法	返 回 值	说 明
getAttribute(String name)	Object	获得指定名字的属性
getAttributeNames()	Enumeration	获得 session 中所有属性对象
getCreationTime()	long	获得 session 对象创建时间
getId()	String	获得 session 对象唯一编号

续表

方 法	返 回 值	说 明
getLastAccessedTime()	long	获得 session 对象最后一次被操作的时间
getMaxInactiveInterval()	int	获得 session 对象有效时间
isNew()	boolean	判断 session 对象是否为新创建的
removeAttribute(String name)	void	删除 session 对象中指定名称的属性
invalidate()	void	销毁 session 对象
setMaxInactiveInterval(int interval)	void	设置 session 对象的最大有效时间
setAttribute(String key,Object obj)	void	将 obj 以 key 名称保存在 session 中

3.4.1 创建及获取 session 信息

session 是与请求有关的会话对象，是 `java.servlet.http.HttpSession` 对象，用于保存和存储页面的请求信息。session 对象的 `setAttribute()` 方法可实现将信息保存在 session 范围内，而通过 `getAttribute()` 方法可以获取保存在 session 范围内的信息。

`setAttribute()` 方法的语法格式如下：

```
setAttribute(String key,Object obj)
```

- ☑ key: 保存在 session 范围内的关键字。
- ☑ obj: 保存在 session 范围内的对象。

`getAttribute()` 方法的语法格式如下：

```
getAttribute(String key)
```

key: 指定保存在 session 范围内的关键字。

例 3.07 创建和获取 session 信息。（实例位置：光盘\TM\Instances\3.07）

(1) 在 `index.jsp` 页面中，实现将文字信息保存在 session 范围内。

```
<body>
  <%
    String sessionMessage = "session 练习";
    session.setAttribute("message",sessionMessage);
    out.print("保存在 session 范围内的对象为: "+sessionMessage);
  %>
</body>
```

运行结果如图 3.9 所示。

保存在session范围内的对象为: session练习

图 3.9 index.jsp 页面运行结果

(2) 在 `default.jsp` 页面中，获取保存在 session 范围内的信息并在页面中显示。

```

<body>
<%
    String message = (String)session.getAttribute("message");
    out.print("保存在 session 范围内的值为: "+message);
%>
</body>

```

运行结果如图 3.10 所示。

保存在session范围内的值为: session练习

图 3.10 default.jsp 页面运行结果

⚠ 注意: session 默认在服务器上的存储时间为 30 分钟, 当客户端停止操作 30 分钟后, session 中存储的信息会自动失效。此时调用 `getAttribute()` 等方法, 将出现异常。

3.4.2 从会话中移除指定的绑定对象

对于存储在 session 会话中的对象, 如果想将其从 session 会话中移除, 可以使用 session 对象的 `removeAttribute()` 方法。

语法格式如下:

```
removeAttribute(String key)
```

key: 保存在 session 范围内的关键字。

例如, 将保存在 session 会话中的对象移除。

```
session.removeAttribute("message");
```

3.4.3 销毁 session

当调用 session 对象的 `invalidate()` 方法后, 表示 session 对象被删除, 即不可以再使用 session 对象。语法格式如下:

```
session.invalidate();
```

如果调用了 session 对象的 `invalidate()` 方法, 之后在调用 session 对象的任何其他方法时, 都将报出 `Session already invalidated` 异常。

3.4.4 会话超时的管理

在应用 session 对象时应该注意 session 的生命周期。一般来说, session 的生命周期在 20~30 分钟之间。当用户首次访问时将产生一个新的会话, 以后服务器就可以记住这个会话状态, 当会话生命周期超时, 或者服务器端强制使会话失效时, 这个 session 就不能使用了。在开发程序时应该考虑到用