1 模块 1

Java Web 应用开发概述

知识目标

- 了解 Web 应用、Web 应用开发、动态网页及 B/S 结构等 Web 基础知识。
- 了解 JSP 的执行流程和技术优点。
- 掌握 Web 应用开发的流程和简单 JSP 程序的编写与运行方法。

技能目标

- 能够在 MyEclipse IDE(integrated development environment,集成开发环境)中配置 JRE。
- 能够在 MyEclipse IDE 中配置并运行 JSP 引擎 Tomcat。
- 能够在 MyEclipse IDE 中创建并部署 Java Web 项目。
- 能够在 MyEclipse IDE 中编写简单 JSP 程序并运行 JSP 程序。

1.1 Web 基础知识

Web 是存储在 Internet 上数量巨大的文档的集合,它是一种基于超文本和 HTTP 的、全球性的、动态交互的、跨平台的分布式图形信息系统。Web 同时也是一种建立在 Internet 上的网络服务,为浏览者在 Internet 上查找和浏览信息提供了图形化的、易于访问的直观界面,其中的文档及超级链接将 Internet 上的信息节点组织成一个互为关联的网状结构。

1.1.1 Web 应用及其开发

Web 应用(Web application)可以是一个构建并运行在 Web 上的完整网站,也可以仅仅是某个运行在 Web 上的应用程序。它是由一些 Web 网页和用来完成某些特定任务的相关资源整合在一起而形成的一个数据集合,通过服务器、客户端及网络等实现自身的功能。

Web 应用可采用不同的技术来构建,每种技术皆有其优劣势。当今的 Web 应用在其开发线路上主要包含两部分内容:Web 前端开发和 Web 后端开发。

Web 前端开发主要进行网站开发、优化和完善,包括 Web 页面的结构、Web 的外观视觉表现及 Web 层面的交互实现等工作。Web 前端开发在知识体系运用和开发方式上涵盖了许多传统的网站后台开发内容,不再局限于传统的以网页制作为主的网站前台开发。目前,必须掌握的 Web 前端开发技术主要有 HTML、CSS、JavaScript、DOM、JSON、JQuery、Ajax 等。

Web 后端开发主要进行与数据库交互作用所对应的业务逻辑设计,需要考虑功能的实现、数据的存取、平台的稳定性与性能等。目前,流行的 Web 后端开发技术主要有 Java EE、.NET、PHP 等。当然,Web 后端开发还涉及数据库管理系统、Web 应用服务器、设计模式等方面知识和技能的运用。

与过去在 Web 应用开发中所使用的“前台”和“后台”的概念相比,今天的“前端”和“后端”的划分则对技术准则进行了广泛的延伸及综合。随着大数据、云计算及人工智能化时代的逐步深入,如智能手机移动终端应用的大规模普及,今天的 Web 应用开发从宏观到微观,从内涵到外延,都与传统的 Web 应用开发大不相同了。

1.1.2 动态网页

在 Internet 发展初期,Web 应用只是一个静态网站。所谓静态,就是指网站的所有网页内容都基于静态的 HTML 页面;网站内容的修改只能通过修改静态的 HTML 页面来实现;Web 网站所能实现的任务仅仅是静态的信息显示,而不能与用户产生互动。

随着商业需求的不断增长和 Internet 技术的不断发展,出现了所谓的用动态网页技术开发的动态网站。动态网页是指在服务器端运行的程序或网页,它显示的内容随时间、用户及用户需求的改变而改变。例如,当我们作为普通用户登录某论坛网站时,只能看到帖子的浏览页面;而作为管理员登录时,同样的页面中还会出现“添加”“删除”“修改”等操作提示。此外,在不同的时间登录论坛,看到的帖子列表也是不同的。

动态网页的主要特点如下。

(1)交互性。网页会根据用户的要求和选择而动态改变与响应。例如,用户在网页中填写表单信息并提交,服务器将所提交信息处理后自动存储到后台数据库中,并转至相应的提示页面。采用动态网页技术的网站可以实现更多的与用户的交互功能,如用户注册、用户登录、信息查询、用户管理和在线操作等。

(2)自动更新。与静态网页修改后需重新上传以覆盖原先的页面相比,动态网页无须手动操作,便会自动生成更新的页面,大大节省了工作量。

(3)随机性。不同的人在不同的时间访问同一网址时会产生不同的页面效果。

当今的网站根据其功能需求和内容的多寡,往往采用“动静结合”的原则开发实现。在同一个网站上,动态网页内容和静态网页内容同时存在是很常见的事情。

动态网页的设计需要使用到服务器端脚本技术,如 JSP(Java server pages,Java 服务器页面)技术。本书将以 JSP 技术为核心,引导读者进行动态网站开发。使用动态网页技术开发的网站实际上是 B/S 结构的一种体现。因此,要想熟练掌握动态网页的开发技术,首先需要了解 B/S 结构的一些知识,这样才能更好地学习使用 JSP 技术开发动态网页。

1.1.3 B/S 结构

动态网站等 Web 应用普遍采用分布式计算模式构建,使用该模式进行 Web 应用开发时,往往需要编写大量的程序。这些程序被部署在不同的计算机上,在 Web 应用中承担着不同的职责。例如,有的程序显示用户界面,有的程序进行逻辑运算,有的程序则进行后台数据处理。

相应的 Web 应用在体系上分为多层结构,其中最典型的是 B/S/D(browser/server/database,浏览器/服务器/数据库)三层结构,如图 1-1 所示。

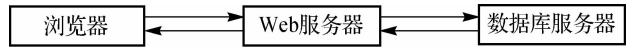

图 1-1 B/S/D 三层结构

第 1 层为浏览器。浏览器接收用户输入的数据,提交并发送至 Web 服务器;还会接收 Web 服务器返回的数据处理结果并显示。

第 2 层为 Web 服务器(安装了提供 Web 服务的应用程序及管理应用程序的计算机)。运行在 Web 服务器上的应用程序接收并处理用户数据,并根据流程需要与否在数据库中进行数据比对和处理。最后 Web 服务器综合各项数据处理结果,生成动态页面,通过浏览器呈现给用户。

第 3 层为数据库服务器。数据库服务器存储海量的用户信息,并提供数据处理和事务处理功能。

对于一些规模不是很大的动态网站等 Web 应用,会把 Web 服务器和数据库服务器安排在同一台计算机上。在这种情况下,第 2 层和第 3 层就合并为同一层,统称为服务器端,也就形成了所谓的 B/S(浏览器/服务器)结构。在 B/S 结构的 Web 应用中,浏览器端与服务器端采用请求/响应模式进行交互。

随着 Internet 技术的不断深入,作为 C/S(client/server,客户机/服务器)结构的一种改进,B/S 结构使得 Web 应用的维护和升级更为简单,用户访问范围更广,信息资源共享程度更高。

1.1.4 JSP 简介

JSP 技术是 Java EE(Java 企业版)中的一个关键技术,是用 Java 标准开发服务器端 Web 应用的主要技术。打开 JSP 文件,可以看到 JSP 文档是在 HTML 代码中嵌入 Java 脚本(添加了 JSP 标记的 Java 代码)编写而成的。JSP 文件以.jsp 为扩展名。

当我们通过浏览器访问一个 JSP 页面时,Web 服务器根据该请求加载被请求的 JSP 文件。然后,Web 服务器中的 JSP 引擎(用来统一管理和运行 Java Web 应用程序的软件,也可称为 JSP 容器,如 Tomcat、JRun、Resin 等)将被加载的 JSP 文件转译成 Servlet 源文件(以.java 为扩展名)。JSP 引擎再将生成的 Servlet 源文件编译成 Java 类文件(以.class 为扩展名)。最后,Web 服务器执行该 Java 类文件,并将结果在浏览器显示,如图 1-2 所示。

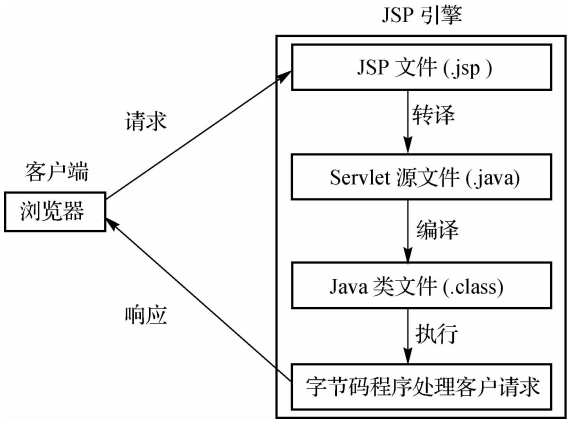

图 1-2 JSP 的执行流程

作为 Servlet(JSP 诞生之前的一种服务器端动态页面 Java 编写技术)的一种改进,JSP 页面负责将服务器端运算结果与 HTML 等静态内容结合后发送到浏览器,Servlet 和 JavaBean(一种 JSP 组件)则负责较为复杂的业务逻辑计算,生成相应的动态运算结果。

JSP 技术特征如下。

(1)JSP 分离了服务器端程序中的静态内容和动态内容。

(2)JSP 提供组件(如 JavaBean)、标准标签、自定义标签等可一次生成重复利用的实现方式,大大提高了程序开发效率。

(3)JSP 页面在被第一次请求时进行编译,如果在后续的请求中该页面没有改动,服务器可直接调用相应的已被编译好的代码,大大提高了访问速度。

(4)JSP 沿用了 Java 强大的 API 功能,只要服务器支持 JSP,就可以运行用 JSP 开发的 Web 应用,体现了 JSP 的跨平台优点。

1.2 创建第一个 Web 项目

使用 JSP 进行 Web 应用开发，需要准备以下开发工具：JDK、Web 服务器、JSP 集成开发环境、Web 浏览器及 Web 数据库等。下面逐一介绍它们的准备方法。

1.2.1 JSP 运行环境搭建

1. JDK

首先要安装 JDK（Java development kits，Java 开发包）。JDK 可到 Java 官方网站 https://www.oracle.com 进行下载，如图 1-3 所示，然后运行安装即可。本书建议采用 JDK 1.6 及以上版本。

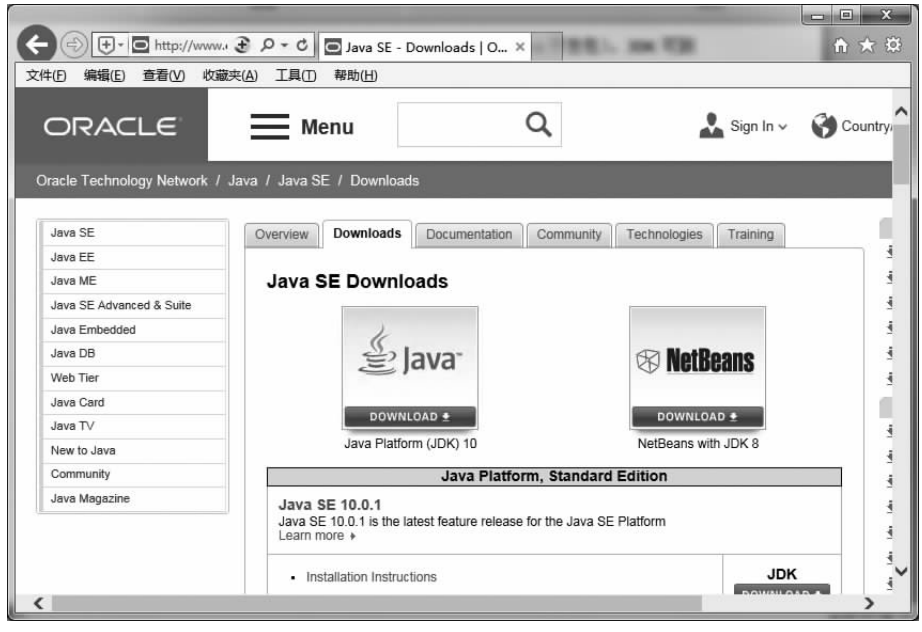

图 1-3 Java 官方网站

2. Web 服务器

本书选择 Tomcat 作为 Web 服务器中的 JSP 引擎。Tomcat 可到 Apache Tomcat 官方网站 http://tomcat.apache.org 进行下载，如图 1-4 所示，然后解压缩至某固定文件夹即可（也可下载相应的安装程序安装）。本书采用 Apache tomcat 6.0.33 版本。

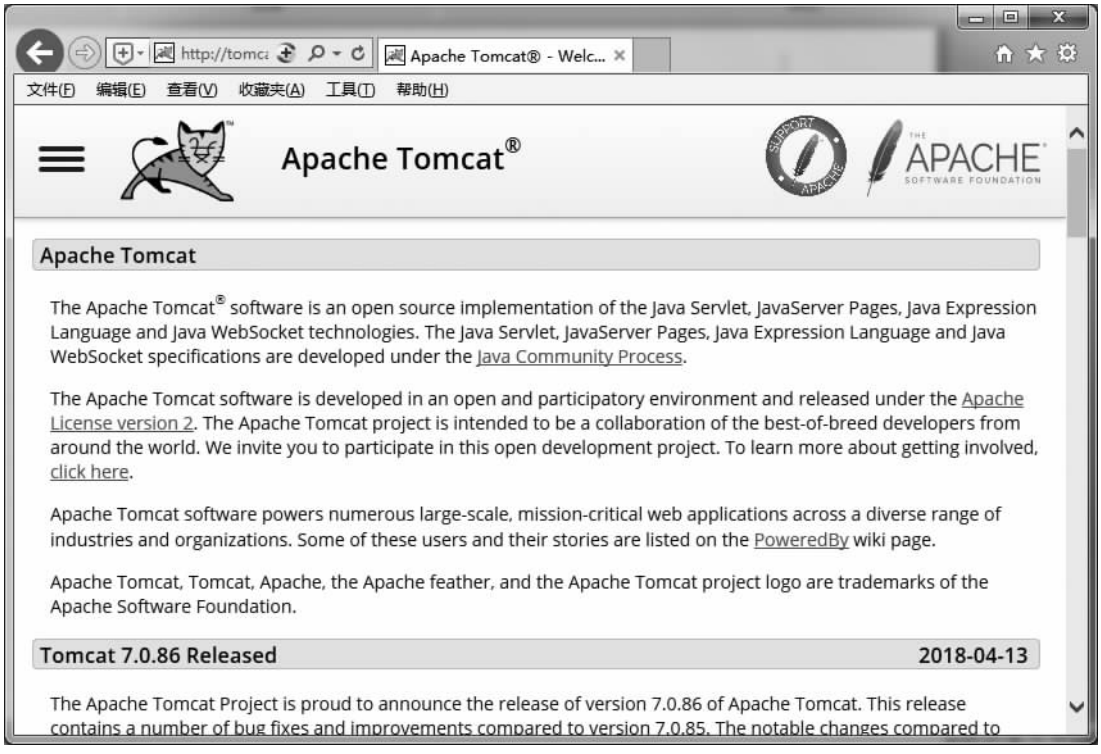

图 1-4　Apache Tomcat 官方网站

3. Web 浏览器

浏览器是用户访问 Web 应用的工具，JSP 开发的 Web 应用程序对浏览器没有特殊要求。本书采用 IE 11 作为 Web 浏览器。

4. Web 数据库

JSP 开发的 Web 应用程序通过 JDBC 驱动可访问多种数据库管理系统。在实验室学习环境下，数据存取规模不大，故本书选用 MySQL 数据库管理系统提供 Web 数据库服务。MySQL 数据库管理系统可到 MySQL 官方网站 https://www.mysql.com 下载，如图 1-5 所示。本书采用 MySQL 5.5 版本。

1.2.2　JSP 集成开发环境 MyEclipse

MyEclipse IDE 是一个成熟的用于 Java Web 应用开发的企业级平台。MyEclipse IDE 可到 MyEclipse 官方网站 https://www.genuitec.com 或 http://www.myeclipsecn.com 下载，如图 1-6 所示。本书采用 MyEclipse 8.5 版本。

图 1-5 MySQL 官方网站

图 1-6 MyEclipse 官方网站

在 Windows 操作系统中，执行“开始”→“所有程序”→“MyEclipse 8.5”命令，启动

MyEclipse 8.5，会出现图 1-7 所示的 MyEclipse 8.5 主窗口。

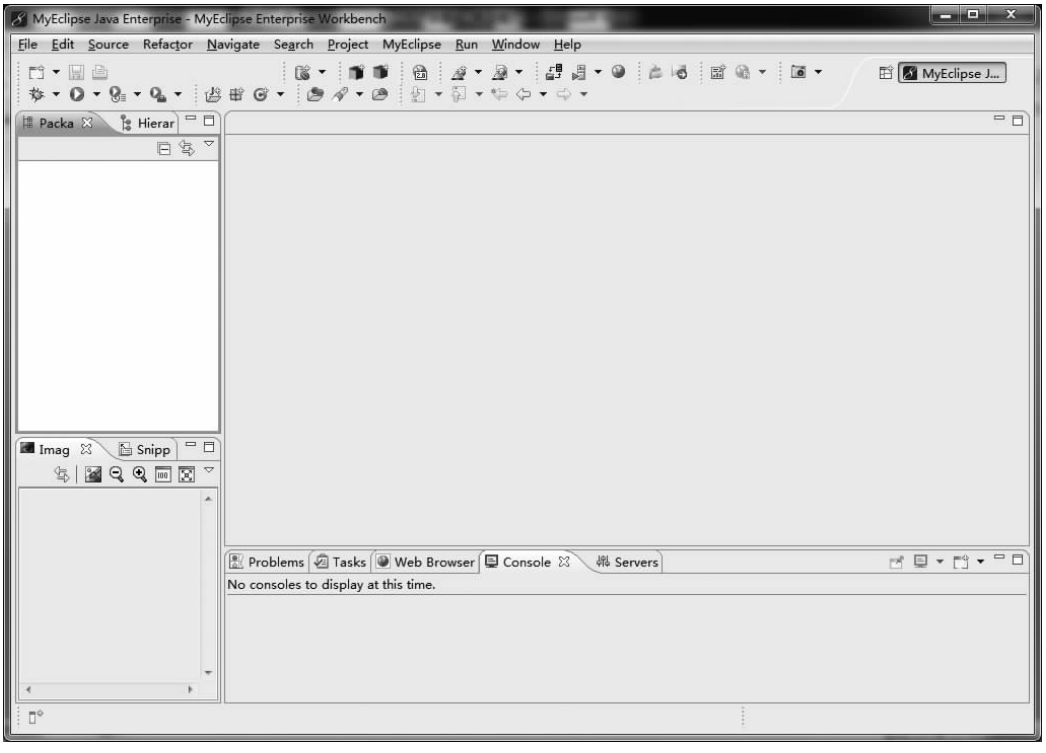

图 1-7　MyEclipse 8.5 主窗口

每次启动 MyEclipse 时，都会提示选择工作区，选定好且以后不打算更改，可选中左下方的复选框，再单击“OK”按钮即可，如图 1-8 所示。

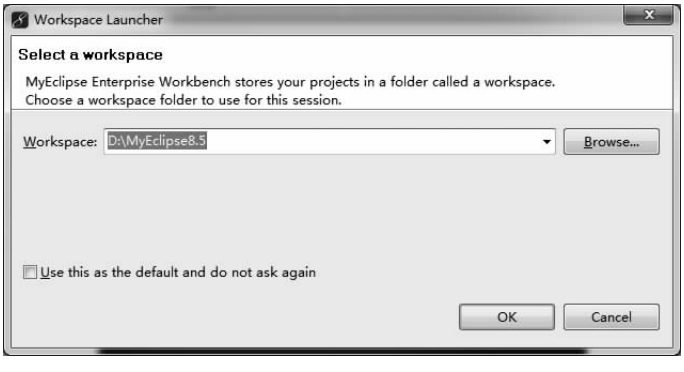

图 1-8　选择 MyEclipse 工作区

1.2.3　在 MyEclipse 8.5 下用 JSP 开发 Web 项目

在所有需要下载、安装的软件准备好之后，就可以在 MyEclipse 8.5 上创建 Web 项目，编写 JSP 页面了。但是，在 Web 项目创建前后，还需要按以下步骤做一些整合工作，以便所创建的 Web 项目能成功运行。

1. 配置 JRE

运行 Java 程序时需要 JRE(Java runtime environment,Java 运行时环境)的支持。可根据需要手动配置已下载的较高版本的 JRE。启动 MyEclipse 8.5,执行“Window”→“Preferences”命令,在打开的“Preferences”窗口中展开左侧目录树中的“Java”项,从中选择“Installed JREs”项,如图 1-9 所示。我们可以发现 MyEclipse 8.5 内嵌的 JDK 版本为 1.6.0。

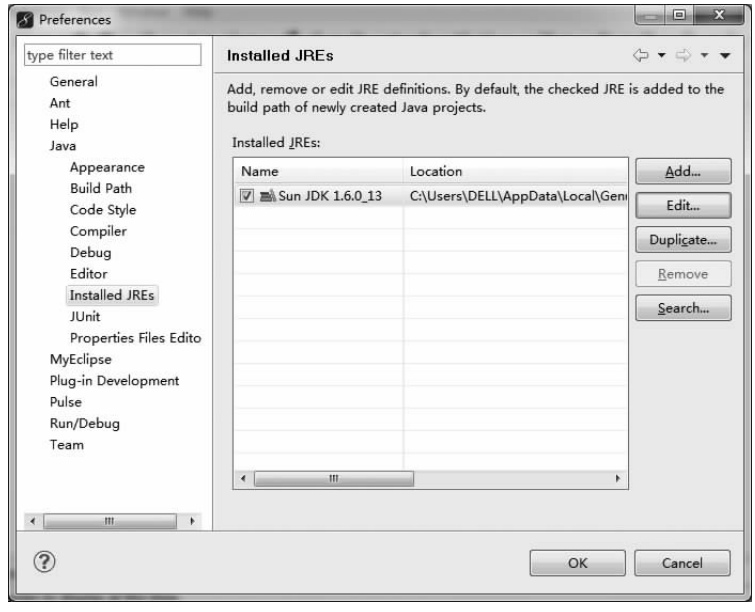

图 1-9 已安装的 JRE

如果需要更换为更高版本的 JRE,可单击“Add”按钮,在打开的“Add JRE”窗口中选择“Standard VM”,单击“Next”按钮,如图 1-10 所示。

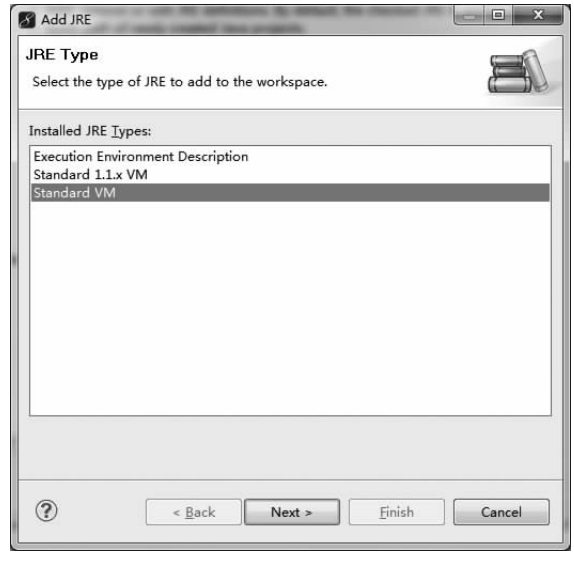

图 1-10 JRE 类型选择

在出现的“JRE Definition”对话框中单击“Directory”按钮，如图 1-11 所示。

图 1-11　JRE 定义

在打开的“浏览文件夹”对话框中选择已安装的较高版本的 JDK，单击“确定”按钮，如图 1-12 所示。

图 1-12　选择已安装的较高版本的 JDK

返回到添加好路径的“JRE Definition”对话框，单击“Finish”按钮完成 JRE 的配置。

注意：不要忘了在“Installed JREs”选项组中选中新添加的 JRE，使配置生效。

2. 配置 Tomcat

在 MyEclipse 8.5 主窗口中执行“Window”→“Preferences”命令，在打开的“Preferences”窗口中展开左侧目录树中的“MyEclipse”项，展开“Servers”项，再展开“Tomcat”项，然后选择“Tomcat 6.x”，在右侧界面上部“Tomcat server”选项组中选中

“Enable”单选按钮，在“Tomcat home directory”文本框中输入 Tomcat 6.0 所在的目录位置，单击“Apply”按钮，如图 1-13 所示。

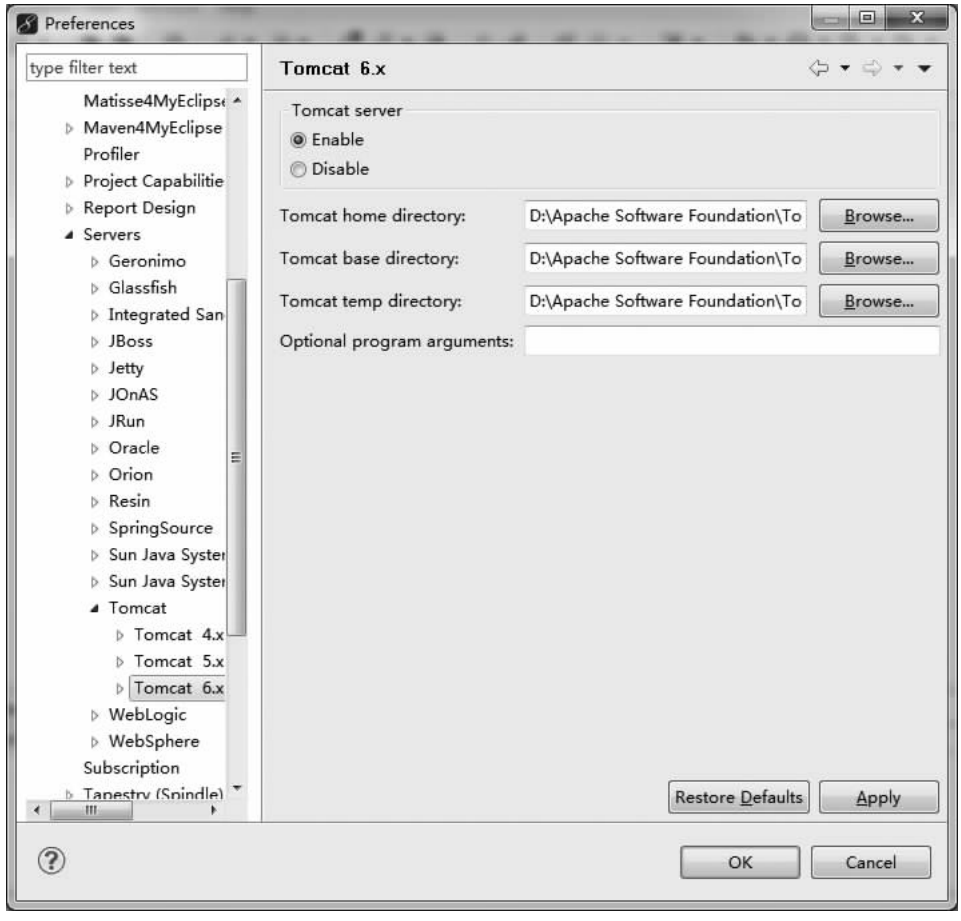

图 1-13　配置 Tomcat

展开左侧目录树中的“Tomcat 6.x”项，选择“JDK”项，如果发现默认 JDK 版本与前述配置的 JRE 版本不一致，可从“Tomcat JDK name”下拉列表框中选择，如图 1-14 所示；如果所需 JDK 版本不存在，则仍需单击“Add”按钮重新配置。以上操作完成后，单击“OK”按钮结束 Tomcat 配置。

3. 启动 Tomcat

启动 Tomcat 的方式有两种：一种是在 MyEclipse 8.5 主界面快捷图标栏单击“服务器”图标下拉箭头，在弹出的下拉菜单中选择“Tomcat 6.x” →“Start”命令，启动 Tomcat；另一种是在主窗体的“Servers”选项区中，选择“Tomcat 6.x”，右击，在弹出的快捷菜单中选择“Run Server”命令，如图 1-15 和图 1-16 所示。

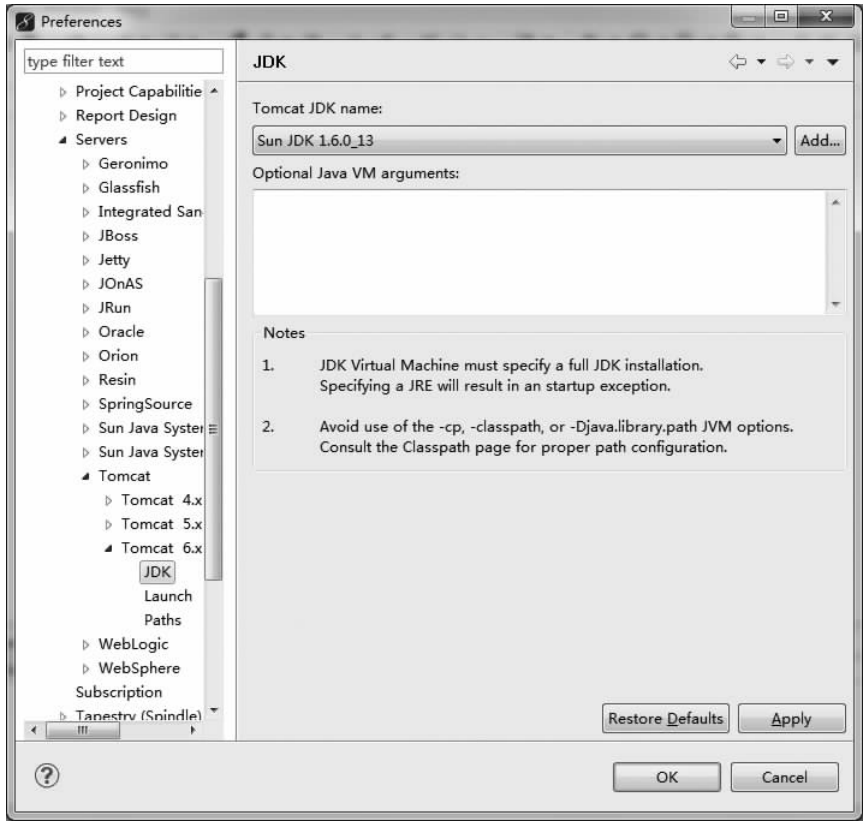

图 1-14　确认 JDK

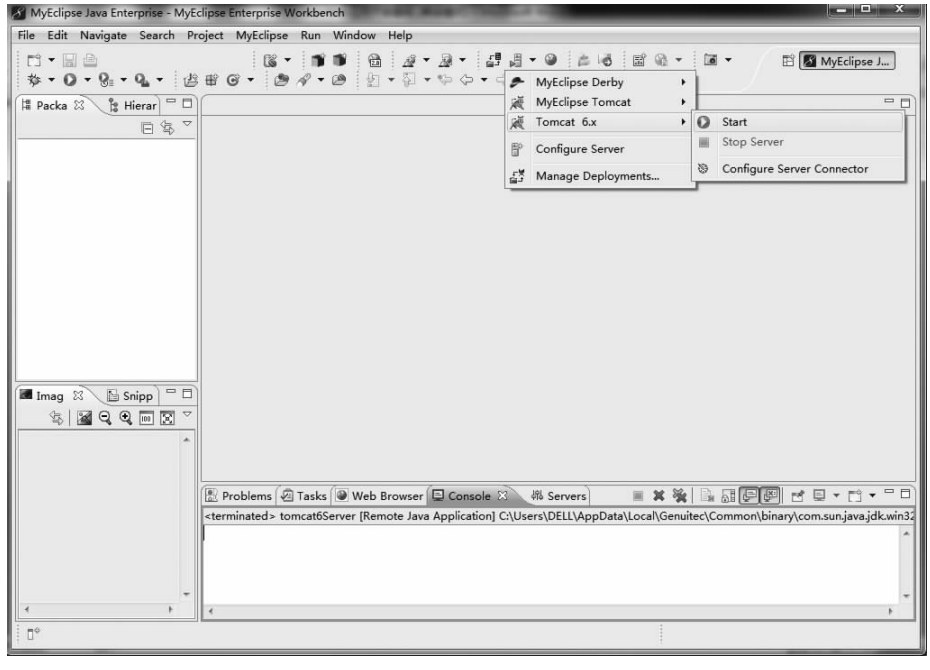

图 1-15　启动 Tomcat 的方式之一

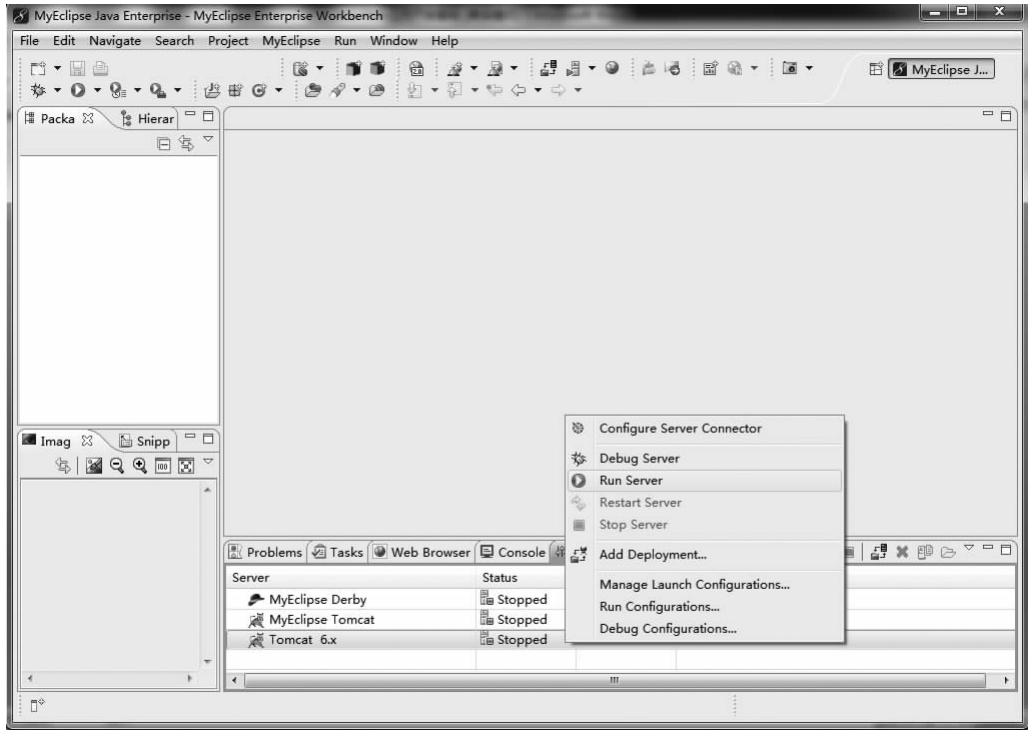

图 1-16　启动 Tomcat 的方式之二

上述命令执行后，会在控制台输出 Tomcat 的启动信息，如图 1-17 所示。

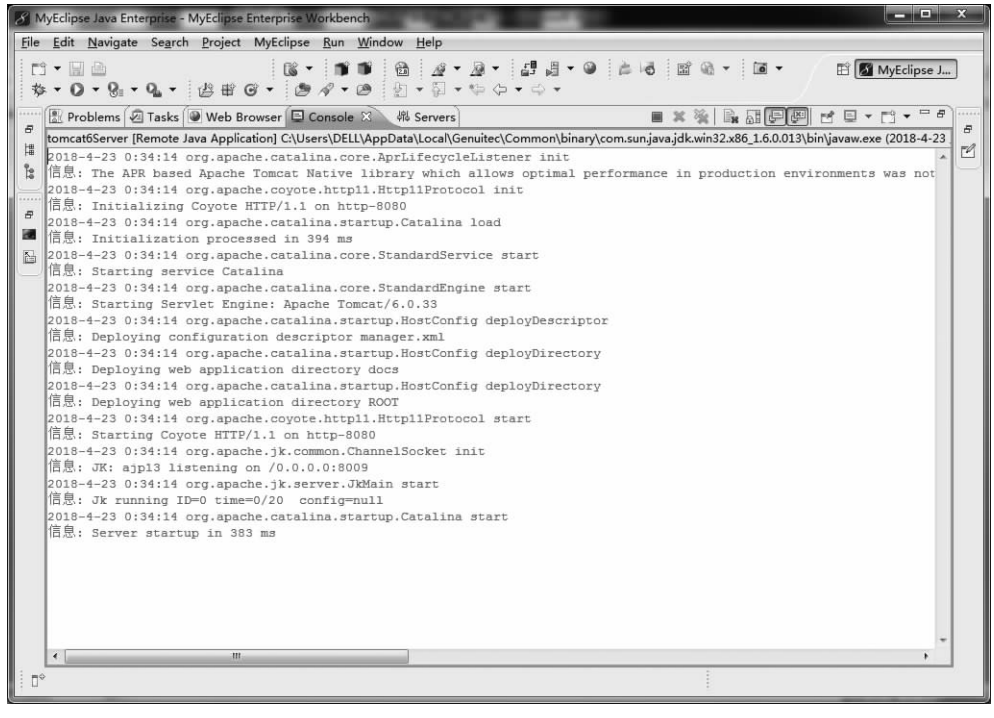

图 1-17　Tomcat 的启动信息

打开浏览器，在地址栏输入 http://localhost:8080，出现如图 1-18 所示的页面，表示 Tomcat 已正式启用。

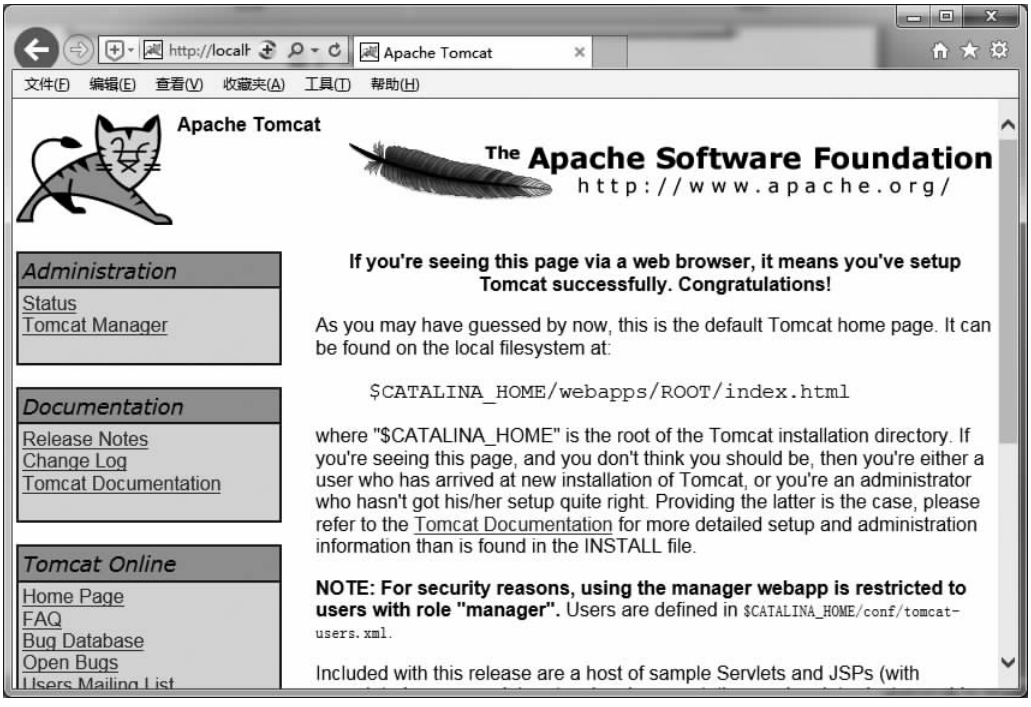

图 1-18　验证 Tomcat 启用

注意：使用超文本传输协议 http。在实验环境下，Web 服务器被安装在本地机上，所以其提供的 Web 服务名称为 localhost。8080 是 Tomcat 安装时默认的端口号，在图 1-17 所示 Tomcat 的启动信息 Initializing Coyote HTTP/1.1 on http-8080 中可以找到这个端口号。图 1-18 表示位于 Tomcat 文件夹../webapps/ROOT 中的 index.html 文件被正确调用。

4. 创建 Web 项目

在 MyEclipse 8.5 主窗口中，执行“File”→“New”→“Project...”命令，打开“New Project”窗口，创建 Java Web 项目向导，依次展开“MyEclipse”→“Java Enterprise Projects”项，从中选择“Web Project”，如图 1-19 所示，单击“Next”按钮，打开如图 1-20 所示的“New Web Project”对话框，在“Project Name”文本框中输入项目名称 myPro，其余选项保持默认设置，单击“Finish”窗口按钮，完成 Web 项目的创建。

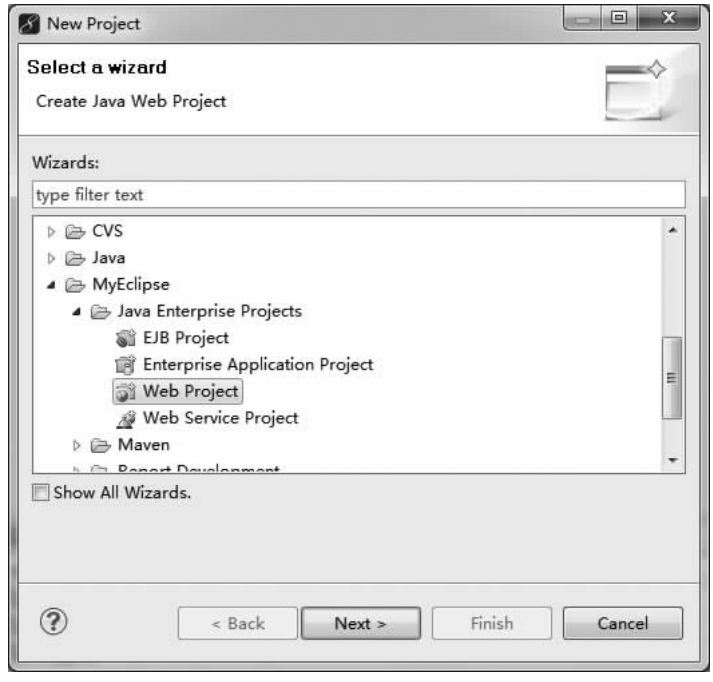

图 1-19　创建 Java Web 项目

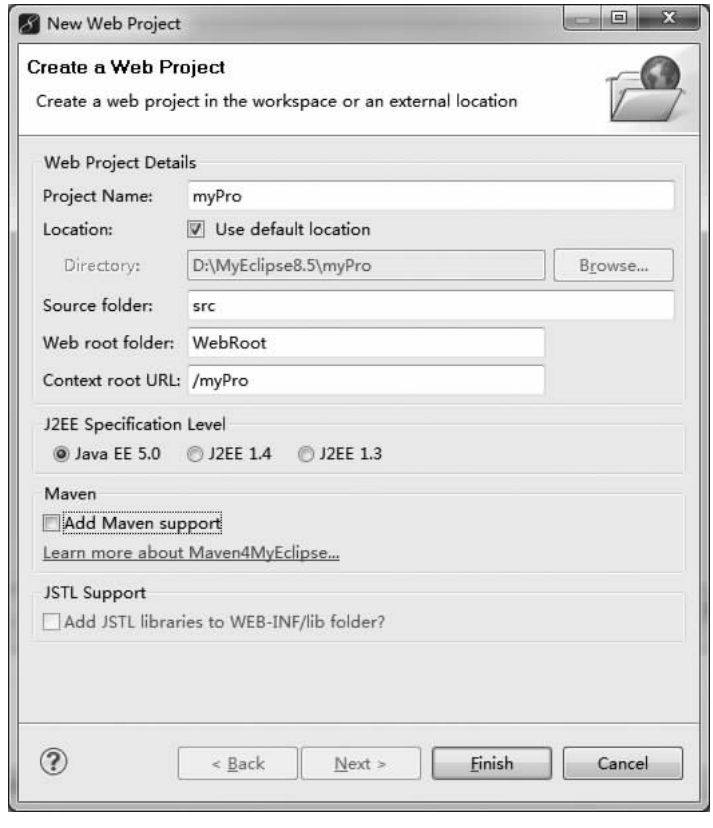

图 1-20　完成 Web 项目的创建

注意：单击“Finish”按钮，完成 Web 项目创建后可能会弹出如图 1-21 所示的对话框，提示所创建项目的编译器遵从级别为 5.0(图 1-20 中的“J2EE Specification Level”最高为 Java EE 5.0)，而当前工作区默认为 6.0(因为配置的 JDK 的版本为 1.6)。此时可使用自定义设置，直接单击“Yes”按钮，无须修改编译器遵从级别。

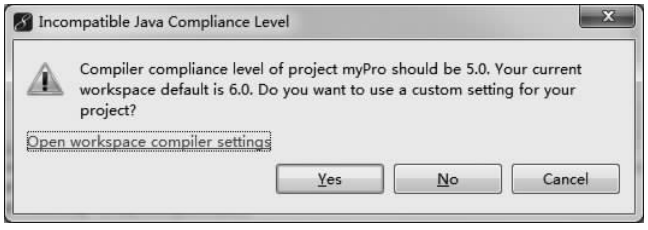

图 1-21　修改编译器遵从级别提示

5. 设计 Web 项目的目录结构

Web 项目要求按特定的目录结构组织文件，当在 MyEclipse 中创建好一个新的 Web 项目后，便可以在 MyEclipse 的“Package Exploxer”中看到该 Web 项目的目录结构，如图 1-22 所示。它是由 MyEclipse 自动生成的。

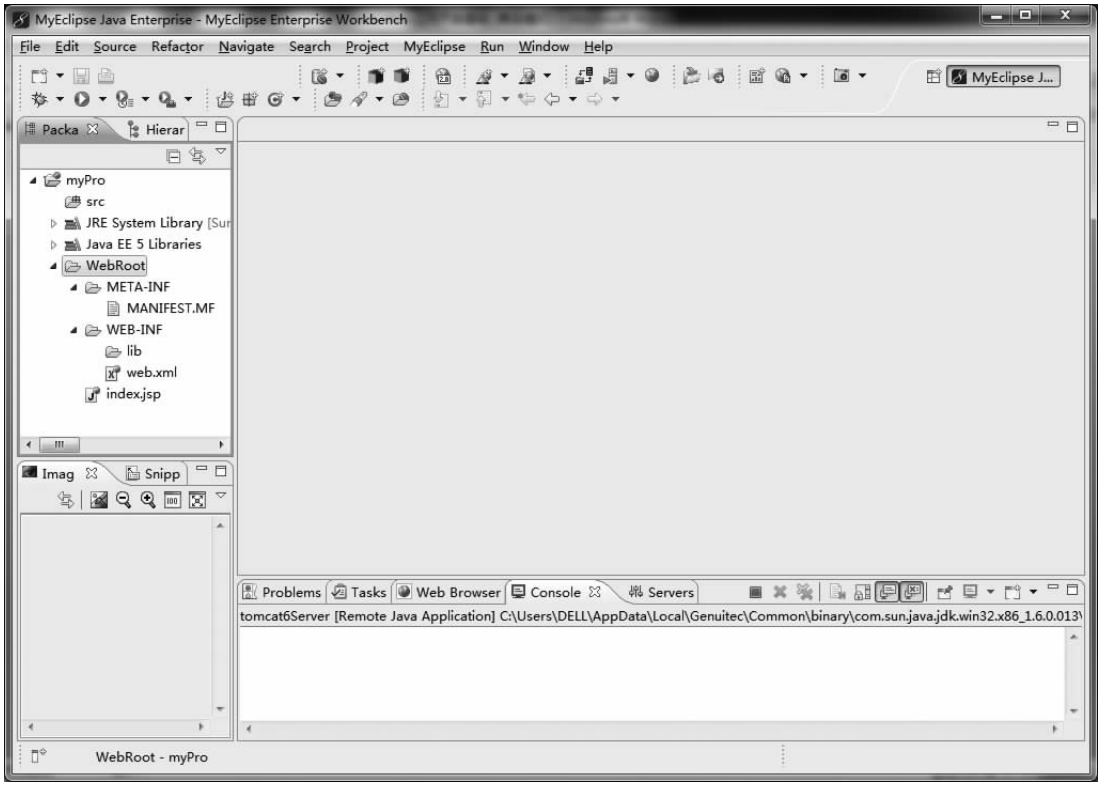

图 1-22　Web 项目的目录结构

下面来了解一下 Web 项目目录和文件的用途。

(1)src 目录:用来存放 Java 源文件。

(2)WebRoot 目录:Web 项目的顶层目录,由以下部分组成。

①META-INF 目录:系统自动生成,存放系统描述信息。

②WEB-INF 目录:无法被用户访问,由以下部分组成。

- lib 目录:它包含 Web 项目所依赖的.jar 或者.zip 文件。
- web.xml:Web 项目的初始化配置文件,不要将其删除或者随意修改。

③JSP 文件:Web 项目中所创建的 JSP 文件默认存放在此目录下。也可以在此目录下创建一些文件夹,根据开发需要把不同的 JSP 文件放在对应的文件夹中。index.jsp 为 Web 项目自动生成的网站主页,或者称为欢迎页面。

④静态文件:包括所有的 HTML 文件、CSS 文件、图像文件等,可按功能以文件夹形式分类存放。

6. 编写 Web 项目的代码

首先,创建一个 JSP 文件:右击 myPro 项目下的 WebRoot 目录,在弹出的快捷菜单中选择"New" →"JSP(Advanced Templates)"命令,利用 JSP 高级模板创建 JSP,如图 1-23 所示。

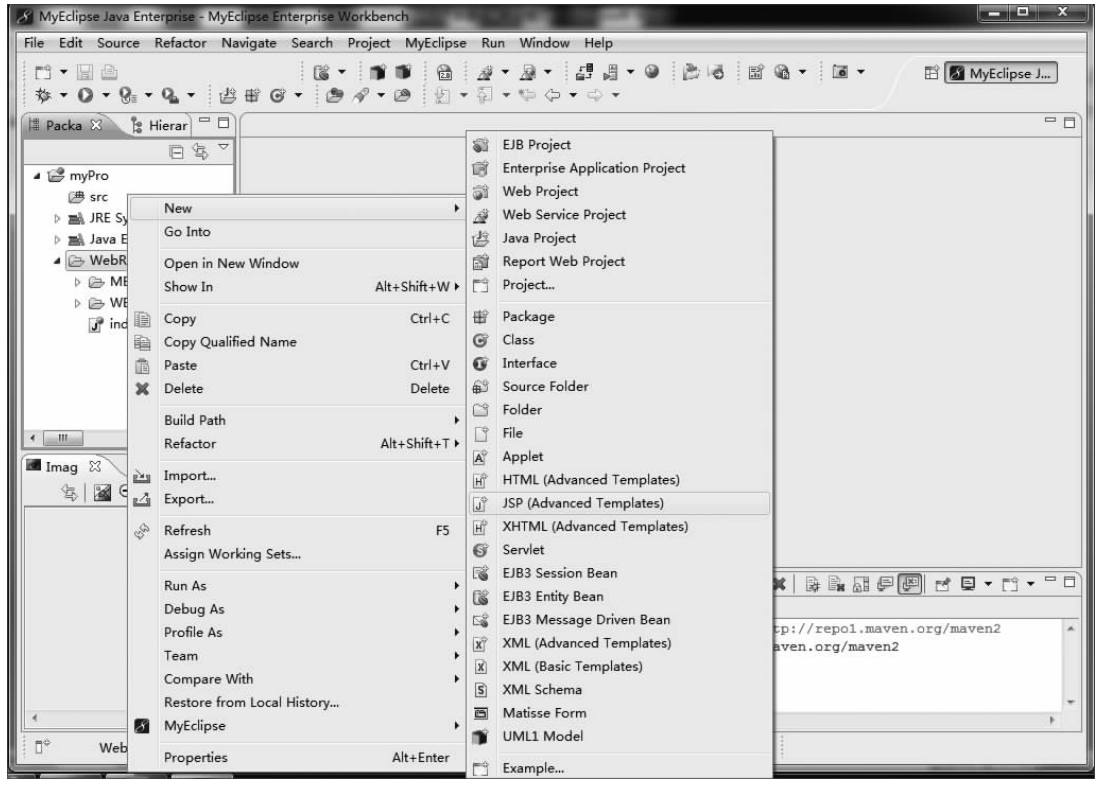

图 1-23 利用 JSP 高级模板创建 JSP

接着,在弹出的对话框中输入文件路径及文件名称,如图 1-24 所示。

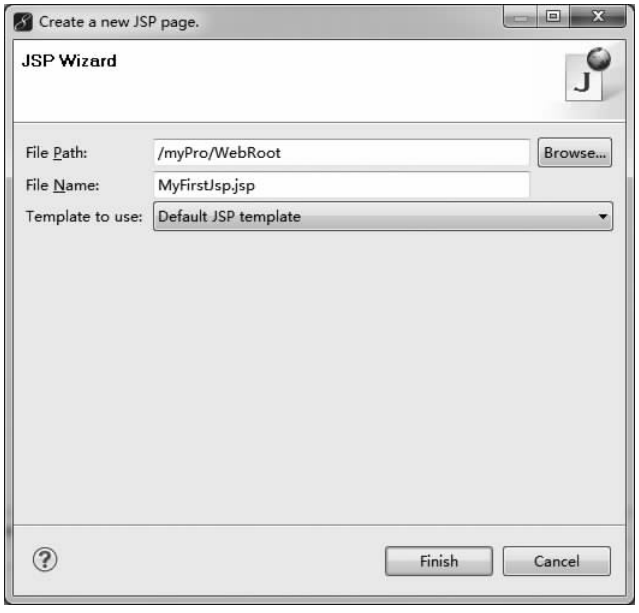

图 1-24　完成 JSP 页面创建

这里的文件名为 MyFirstJsp. jsp，直接放在/myPro/WebRoot 目录下。最后，单击“Finish”按钮，完成 JSP 页面的创建。MyFirstJsp. jsp 在 MyEclipse 主窗口的编辑区同时被打开。在 Design(设计)模式下，编辑区的上部为设计窗格，下部为代码窗格，如图 1-25 所示。

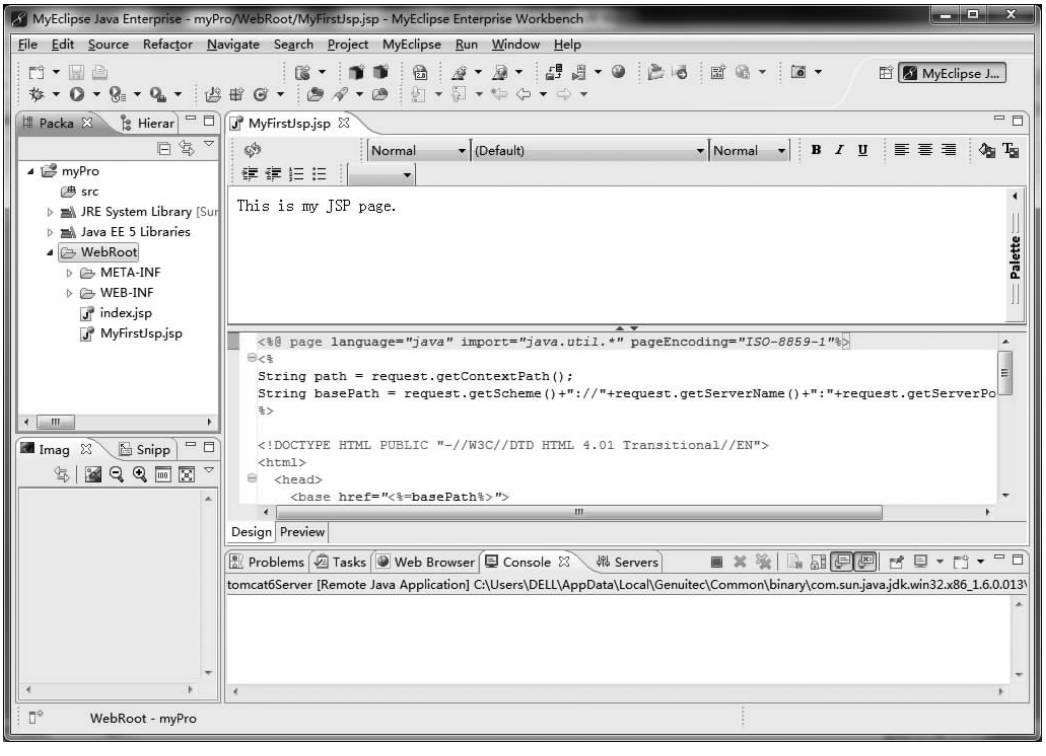

图 1-25　JSP 页面的编辑区

注意：所创建的 JSP 文件必须放在 WebRoot 根目录下或其下自行创建的各级子目录下，否则无法被访问。

7. 部署 Web 项目

部署 Web 项目的方式有两种：一种是在 MyEclipse 8.5 主界面快捷图标栏单击“项目部署”图标；另一种是右击 myPro，在弹出的快捷菜单中选择“MyEclipse” →“Add and Remove Project Deployments”命令，如图 1-26 所示。

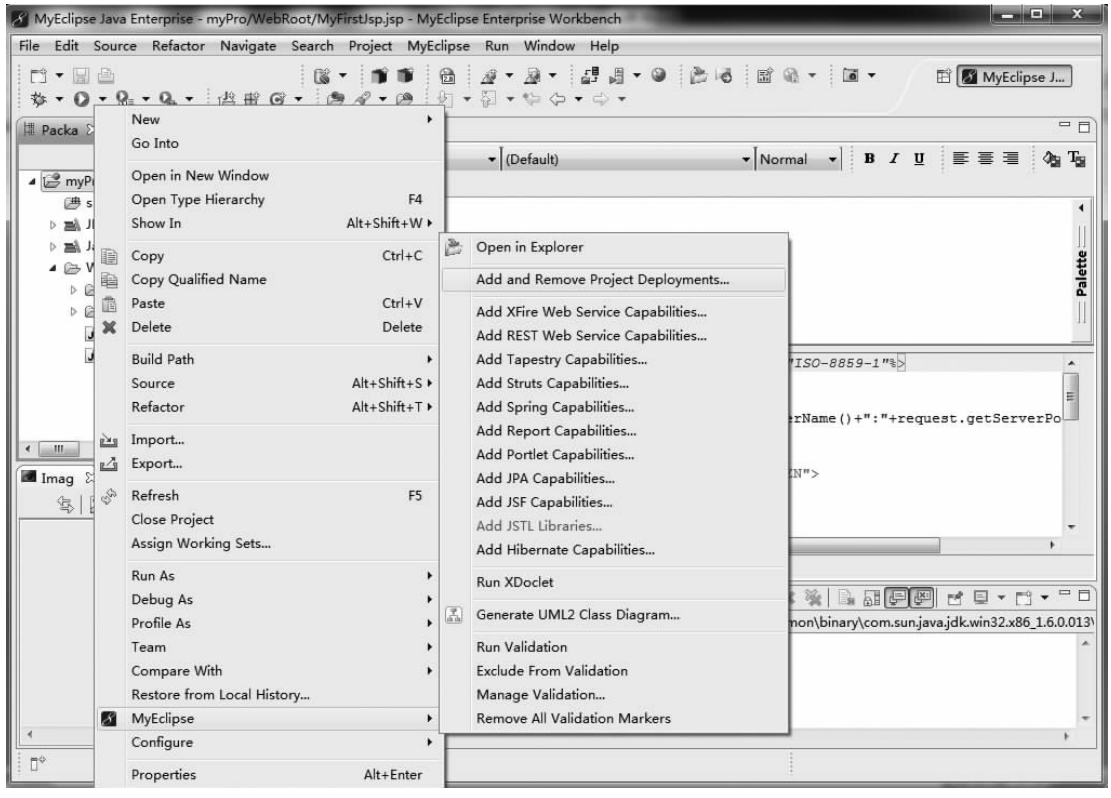

图 1-26　添加与删除项目部署

在弹出的对话框中选择需要部署的项目 myPro，然后单击“Add”按钮，如图 1-27 所示。

在弹出的对话框中选择 Server 为“Tomcat 6.x”，然后单击“Finish”按钮，如图 1-28 所示。

此时会返回到上一个对话框提示部署成功的信息，如图 1-29 所示，单击“OK”按钮关闭该对话框。

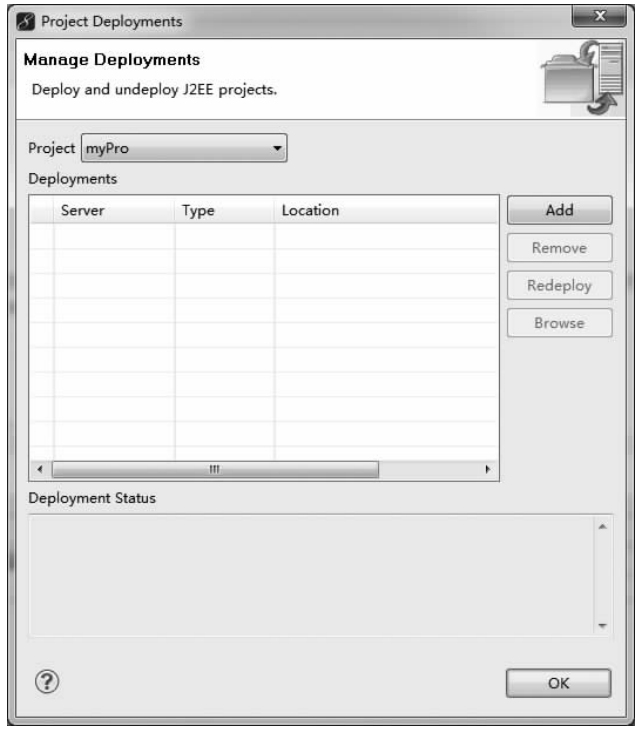

图 1-27　部署 Web 项目

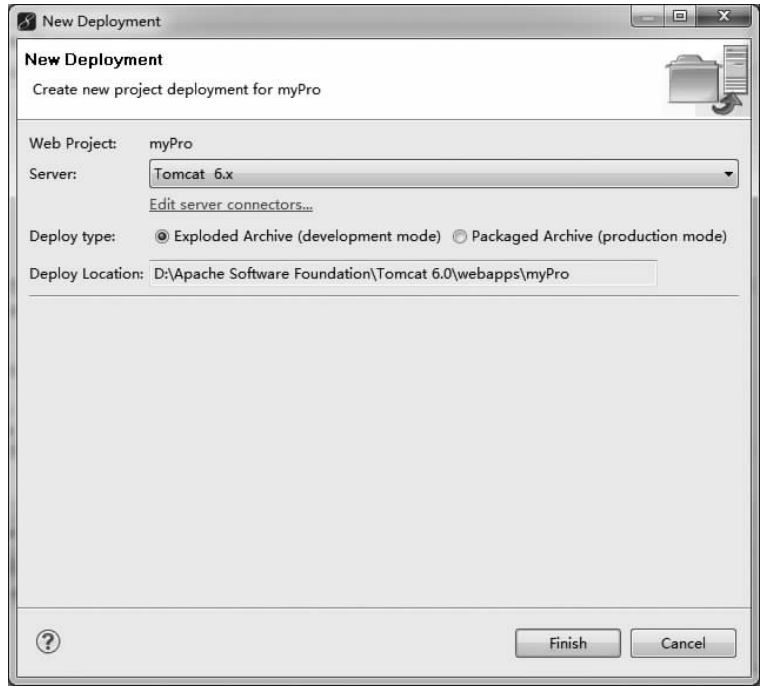

图 1-28　选择服务器

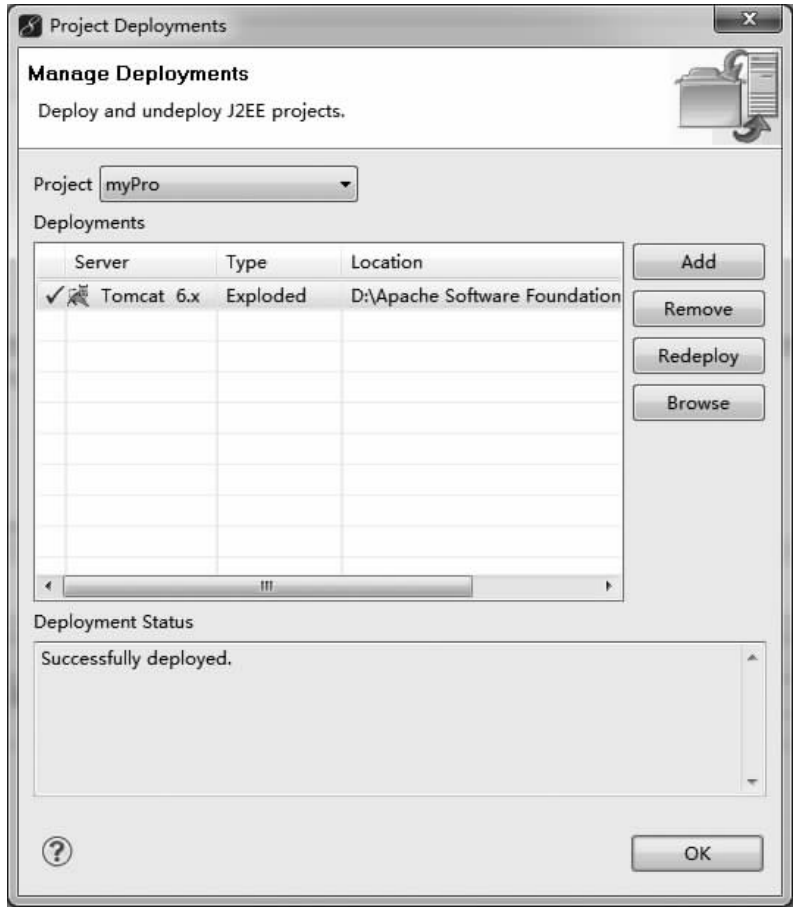

图 1-29 Web 项目部署成功

注意：Web 项目部署成功后才会在 Tomcat 的下一级目录 webapps 中出现所部署的 Web 项目。

8. 运行 Web 项目

打开 IE 浏览器，输入 URL http://localhost:8080/myPro/MyFirstJsp.jsp，并按回车键观看运行结果。在浏览器窗体中显示“This is my JSP page.”，表示 Web 项目 myPro 中的 MyFirstJsp.jsp 页面已经正确运行，如图 1-30 所示。

注意：URL 中的 myPro 是网站对外发布的虚拟上下文路径，它对应的实际路径是 Web 应用的文档根目录，即 WebRoot 文件夹。URL 中 Web 项目资源的访问路径是区分大小写的。

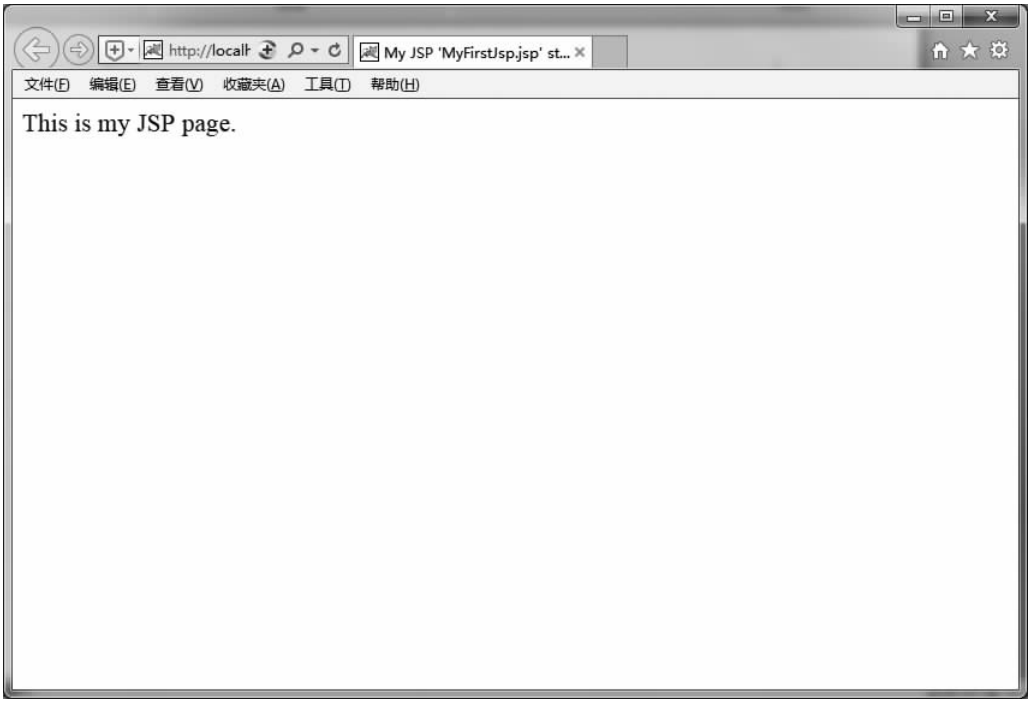

图 1-30　JSP 页面运行结果

1.3 小　结

Web 是存储在 Internet 上数量巨大的文档的集合。Web 应用是由一些 Web 网页和用来完成某些特定任务的相关资源捏合在一起而形成的一个数据集合，通过服务器、客户端及网络等提供服务功能。Web 应用开发包含 Web 前端开发和 Web 后端开发。采用动态网页技术开发的网站可以实现更强的交互性和智能化功能。B/S 结构是 C/S 结构的一种改进，使得 Web 应用的维护和升级更为简单，用户访问范围更广，信息资源共享程度更高。JSP 技术是用 Java 标准开发服务器端 Web 应用的主要技术。使用 JSP 进行 Web 应用开发需要 JDK、Web 服务器、JSP 集成开发环境、Web 浏览器及 Web 数据库等开发工具。在 MyEclipse 8.5 下用 JSP 开发 Web 项目的步骤为：配置 JRE—配置 Tomcat—启动 Tomcat—创建 Web 项目—设计 Web 项目的目录结构—编写 Web 项目的代码—部署 Web 项目—运行 Web 项目。

1.4 习　题

1. 用百度搜索 Web、Web 应用、Web 应用开发、动态网页、B/S、JSP 等关键词，收集相关信息进行学习和分析。

2. 回顾并总结在 MyEclipse IDE 下用 JSP 开发 Web 项目的软件安装及配置全过程，并在计算机上进行实践。

1.5 上 机 实 践

1. 在 MyEclipse 8.5 下编写一个 JSP 页面，输出“这是我的第一个 JSP 页面”。

2. index.jsp 为 Web 项目自动生成的网站主页或者称为欢迎页面。如何在浏览器中访问该页面？如何在 Web 项目中更改原先设置的欢迎页面？

2 模块 2

JSP 语法

知识目标

- 掌握小脚本、表达式和声明的基本用法。
- 掌握 page 指令和 include 指令的基本用法。
- 掌握添加 JSP 注释的方法。

技能目标

- 能够在 MyEclipse IDE 中编写小脚本、表达式和声明。
- 能够在 MyEclipse IDE 中编写 page 指令和 include 指令。
- 能够在 MyEclipse IDE 中编写并阅读 JSP 注释。

2.1 回顾和思考

在模块 1 中，我们学习了在 MyEclipse 8.5 IDE 下用 JSP 开发 Web 项目的详细流程。本模块我们将通过分析 JSP 页面代码的构成，学习并掌握基本的 JSP 语法。

【例 2-1】 尝试编写并运行一个 JSP 页面输出系统的当前时间。

程序实现代码如下。

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<%@ page import="java.text.*,java.util.*" %>
<html>
<body>
<%
```

```
    SimpleDateFormat formater = new SimpleDateFormat("yyyy 年 MM 月 dd 日 HH 时 mm 分 ss 秒");
    String strCurrentTime = formater.format(new Date());
%>
<center>
你好,交通信息工程学院! 现在是:
<%= strCurrentTime %>
</center>
</body>
</html>
```

尽管我们对这些代码还不是非常熟悉,但是可以看出 JSP 页面是由 HTML 标签描述的静态文本和 JSP 标记描述的动态内容穿插在一起而构成的。接下来就要对各类代码逐个分析。【例 2-1】中的文件名为 example2_1.jsp,运行结果如图 2-1 所示。

图 2-1 【例 2-1】的运行结果

2.2 JSP 指令元素

通过在模块 1 中用 JSP 高级模板创建 JSP 文件的实例和【例 2-1】,我们可以发现第一行

的代码总是包含在<%@ ... %>中，这就是 JSP 的一条指令。JSP 指令为转译阶段提供全局信息。JSP 指令不直接产生任何可见输出，而只是告诉 JSP 引擎如何处理所涉及的 JSP 文件。JSP 指令在整个 JSP 文件范围内都有效。JSP 指令元素主要有 page 指令和 include 指令。

2.2.1 page 指令

page 指令定义 JSP 文件中的全局属性，设定 JSP 页面的相关属性和相关功能。其语法格式如下。

```
<%@ page 属性 1="值 1" 属性 2="值 2" ... %>
```

page 指令中常见的属性有 language(定义所使用的脚本语言)、import(导入 Java API，可导入多个，中间用逗号隔开)、contentType(JSP 页面的文字类型和编码方式)、pageEncoding(JSP 页面使用的字符编码。如果没有设置该属性，就采用 contentType 属性指定的字符集)、isErrorPage(指定当前页面是否为另一个 JSP 页面的错误处理页面，默认为 false)、errorPage(指定的错误处理页面)。

【例 2-1】的 page 指令设置页面使用的脚本语言为 Java，页面的文字类型和编码方式为“text/html(html 文本)；字符设置为 GBK”，导入 java.util 和 java.text 两个包。

注意：在一个 JSP 页面中，page 指令可以出现多次，除 import 外，一般情况每一种属性只出现一次，后面重复出现的属性设置将覆盖先前的属性设置。习惯上把 page 指令作为 JSP 页面的第一行，但这不是硬性规定。

2.2.2 include 指令

include 指令是将指定位置的文件内容内嵌到当前页面。其语法格式如下。

```
<%@ include file = "文件名" %>
```

include 指令只有一个属性 file。include 指令包含的文件的路径名通常是指相对路径名，不需要端口、协议和域名。如果被包含文件与当前 JSP 文件处于同级目录，则“文件名”前不用指明路径；如果被包含文件处于当前 JSP 文件同级目录的下级目录，则需指明完整的下级目录，所有下级目录一直到被包含文件之间用“/”隔开；如果被包含文件与当前 JSP 文件分处于 Web 项目根目录下的不同级目录，则需指明被包含文件的绝对路径，以“//”开头，再加上其完整的目录路径。

include 指令是静态包含其他文件的，所包含的文件必须符合 JSP 语法，可以是 HTML 文件、JSP 文件、文本文件等。执行被包含文件后，主 JSP 文件的进程将会恢复，继续执行下一行。

注意：被包含文件中不能使用<html>、</html>、<body>或</body>等标记，否

则会影响在原 JSP 文件中同样的标记，有时会导致错误。

我们可以把一些共享性的内容写在一个单独的文件中，然后通过 include 指令引用该文件，从而缓解代码重复冗余问题，并且便于代码的修改和维护。例如，在网站的很多 JSP 页面中都要在顶部显示同样的图片。这时可写一个文件包含该图片及其显示方式，在需要显示该图片的 JSP 页面中用 include 指令嵌入该文件即可。

【例 2-2】 修改【例 2-1】，在其页面顶部增加显示一张图片。

操作步骤如下。

(1)编写一个名为 head. html 的文件，该文件包含一个名为 head. jpg 的图形文件。head. html 代码如下。

```
<html>
<body>
<table width="1300" border="0" align="center">
<tr>
<td width="1300" align="center">
<img src="image/head.jpg"/>
</td>
</tr>
</table>
</body>
</html>
```

(2)编写一个名为 example2_2. jsp 的文件，文件中使用 include 指令引用了 head. html 文件。example2_2. jsp 代码如下。

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<%@ page import="java.text.*,java.util.*" %>
<html>
<body>
<%@ include file = "head.html" %>
<br>
<%
  SimpleDateFormat formater = new SimpleDateFormat("yyyy 年 MM 月 dd 日 HH 时 mm 分 ss 秒");
  String strCurrentTime = formater.format(new Date());
%>
<center>
你好，交通信息工程学院！现在是：
<%= strCurrentTime %>
```

```
</center>
</body>
</html>
```

【例 2-2】的运行结果如图 2-2 所示。

图 2-2 【例 2-2】的运行结果

2.3 JSP 脚本元素

JSP 脚本元素包含小脚本、表达式和声明三种，用来嵌入 Java 代码，实现 JSP 页面的动态请求。

2.3.1 小脚本

小脚本是在 JSP 页面中嵌入的 Java 代码片段。其语法格式如下。

```
<%Java 代码%>
```

在【例 2-1】中，小脚本用两条语句来生成格式化时间字符串。

```
SimpleDateFormat formater = new SimpleDateFormat("yyyy 年 MM 月 dd 日 HH 时 mm 分 ss 秒");
String strCurrentTime = formater.format(new Date());
```

2.3.2 表达式

当 JSP 引擎遇到表达式时，先计算表达式或变量的值，然后把计算结果以字符串形式返回并插入页面的相应位置。其语法格式如下。

```
<%= 表达式 %>
```

在【例 2-1】中，通过表达式变量 strCurrentTime 在页面相应位置输出当前时间。

2.3.3 声明

声明用来定义变量和方法，作用范围为整个 JSP 页面。其语法格式如下。

```
<%! 声明 %>
```

问题：在【例 2-1】中，如果需要在同一个 JSP 页面中的多个位置输出不同格式的当前时间，该怎么办？

在需要输出当前时间的位置写上小脚本代码。

```
SimpleDateFormat formater = new SimpleDateFormat("时间格式");
String strCurrentTime = formater.format(new Date());
```

在构造方法 SimpleDateFormat("时间格式")的参数中设置所需的时间格式，在需要输出当前时间的位置插入表达式<%= strCurrentTime%>。虽然问题被解决了，但是上述小脚本代码在整个页面多次重复出现，显得异常笨拙。这时，把小脚本改成声明即可解决这个问题。

【例 2-3】 利用声明，对【例 2-1】进行更改。

文件名为 example2_3.jsp，其代码如下。

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<%@ page import="java.text.*,java.util.*" %>
<html>
<body>
<%!
  String formatDate(Date d, String format){
    SimpleDateFormat formater = new SimpleDateFormat(format);
    return formater.format(d);
  }
%>
<center>
你好，交通信息工程学院！现在是：
<%= formatDate(new Date(),"yyyy 年 MM 月 dd 日 HH 时 mm 分 ss 秒") %>
```

```
<br>
你好,运输管理工程学院! 现在是:
<%= formatDate(new Date(),"yyyy 年 MM 月 dd 日 hh 时 mm 分 ss 秒") %>
</center>
</body>
</html>
```

【例 2-3】运行结果如图 2-3 所示。

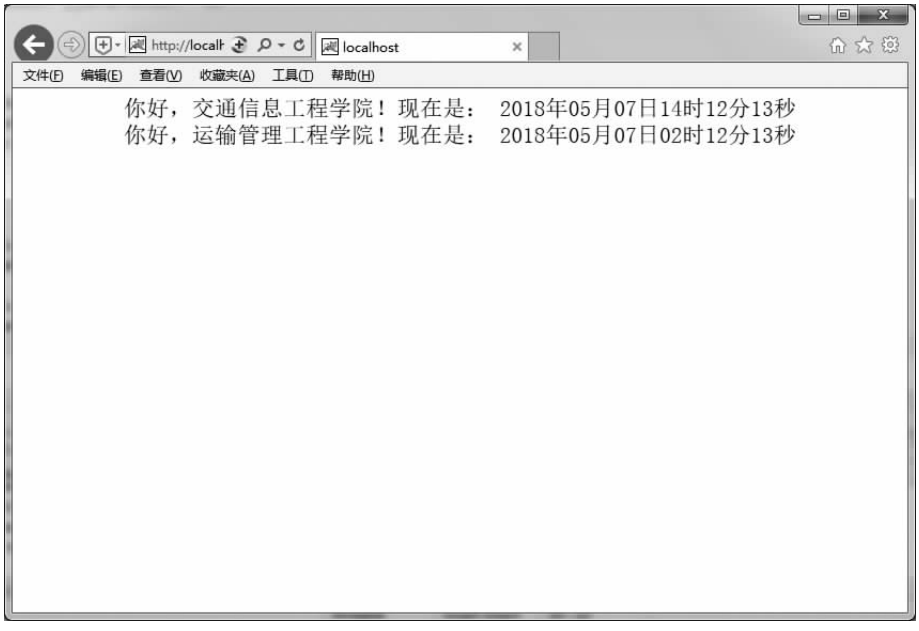

图 2-3 【例 2-3】的运行结果

注意：在 JSP 声明中定义的方法无须放在类体内，所以可以在页面中直接调用所定义的实例方法。无须创建某对象，再用该对象去调用所定义的实例方法。

2.4 JSP 动作元素简介

JSP 动作元素主要有 jsp:forward 动作、jsp:include 动作、jsp:param 动作、jsp:useBean 动作、jsp:setProperty 动作和 jsp:getProperty 动作等。其语法格式如下。

```
<jsp:动作名>...</jsp:动作名>
```

JSP 动作涉及页面之间控制权的转移，后述的内容将陆续介绍以上 JSP 动作。

2.5 JSP 程序中的注释

JSP 程序可以包含 HTML 注释和 JSP 注释，以及脚本元素中的注释。通过添加注释文字，可以提高 JSP 程序的可读性和可维护性。

2.5.1 HTML 注释

HTML 注释被原样发送到客户端。HTML 注释在浏览器中看不到，但是可以通过“查看源代码”看到。其语法格式如下。

```
<!-- 注释内容 -->
```

HTML 注释内如果包含 JSP 脚本元素，这些小脚本、表达式和声明都会被执行且在客户端源代码中显示相应的结果。

2.5.2 JSP 注释

JSP 注释又称隐藏注释，在客户端通过查看源码也看不到。JSP 引擎不会编译这些注释。其语法格式如下。

```
<%-- 注释内容 --%>
```

2.5.3 脚本元素中的注释

脚本元素中包含 Java 代码，因此 Java 中的注释规则在小脚本和声明中同样适用。其语法格式如下。

```
//单行注释
/* 多行注释 */
/** Java DOC 注释 */
```

【例 2-4】 向【例 2-3】中添加一些注释。

文件名为 example2_4.jsp，其代码如下。

```
<!-- example2_3.jsp,输出系统的当前时间,客户端能看到 -->
<%-- example2_3.jsp,输出系统的当前时间,客户端不能看到 --%>
<%@ page language="java" contentType="text/html;charset=GBK" %>
<%@ page import=" java.text.*,java.util.*" %>
<html>
<body>
```

```
<%!
  /*
    格式化输出当前系统时间
    输入参数(Date d, String format)
    返回一个时间字符串
  */
  String formatDate(Date d, String format){
    //调用 SimpleDateFormat(String s)类构造方法
    SimpleDateFormat formater = new SimpleDateFormat(format);
    return formater.format(d);
  }
%>
<center>
你好,交通信息工程学院! 现在是:
<% = formatDate(new Date(),"yyyy 年 MM 月 dd 日 HH 时 mm 分 ss 秒") %>
<br>
你好,运输管理工程学院! 现在是:
<% = formatDate(new Date(),"yyyy 年 MM 月 dd 日 hh 时 mm 分 ss 秒") %>
</center>
</body>
</html>
```

在 IE 浏览器中运行后,执行“查看”→“源”命令,可以看到 HTML 注释,但是看不到 JSP 注释及 Java 注释,如图 2-4 所示。

图 2-4　在客户端看到的【例 2-4】源码

2.6 小　结

JSP 页面由 HTML 静态文本、JSP 指令元素、JSP 脚本元素、JSP 动作元素及注释等内容按相应的语法规定交织构成，以实现所需的语义逻辑。JSP 指令元素主要包括 page 指令和 include 指令。JSP 脚本元素包含小脚本、表达式和声明。JSP 文件中的注释包含 HTML 注释和 JSP 注释，以及小脚本和声明中的 Java 注释。

2.7 习　题

1. 在【例 2-3】中我们学会了使用 JSP 声明来简化不必要的页面代码重复。通过学习，我们知道还有一种办法同样可以简化页面代码，即把重复使用的代码写在一个可共享的文件中，再用 include 指令把该文件嵌入当前页面。编写这个共享文件，并修改例【2-3】，使之达成相同的效果。

2. 试编写一个 JSP 页面，输出表达式 1＋2＋3＋…＋100 的和，并在语句中添加适当注释。

3. 推算以下 JSP 代码片断的运行结果，并思考一下原因。

(1)

```
<%!
int a = 10;
int add(int a){
  a++;
  return a;
}
%>
<%
int a = 100;
%>
<%= add(20) %>
<%= a %>
```

(2)

```
<%!
int a = 10;
int add(int a){
  a++;
  return a;
}
%>
<%
int a = 100;
%>
<%= add(20) %>
<%= this.a %>
```

(3)

```
<%!
int a = 10;
int add(int a){
  this.a++;
  return this.a;
}
%>
<%
int a = 100;
%>
<%= add(20) %>
<%= a %>
```

(4)

```
<%!
int a = 10;
int add(int a){
  this.a++;
  return this.a;
}
%>
<%
```

```
this.a = 100;
%>
<%= add(20) %>
<%= a %>
```

4. 试编写一个 JSP 页面，实时显示当前的系统时间。提示：添加以下 HTML 代码，以实现页面的自动刷新功能。

```
<meta http-equiv="refresh" content="1;url=exercise2_4.jsp">
```

refresh 表示要刷新页面，1 表示刷新的时间间隔是 1 秒，exercise2_4.jsp 指要刷新的页面。

2.8 上机实践

1. 试编写一个 JSP 页面，输出 100 以内的所有素数，并在语句中添加适当的注释。

2. 试编写一个 JSP 页面，产生一个异常，然后转到另一个页面处理该异常。提示：定义 page 指令中的 isErrorPage 属性和 errorPage 属性。

3. 试编写一个 JSP 页面，统计该网页的访问量。提示：在声明的方法中使用 synchronized 关键字。

4. 试编写一个JSP 页面，输入你的 18 位身份证号，从中分析出你的生日。其结果如图 2-5 所示。

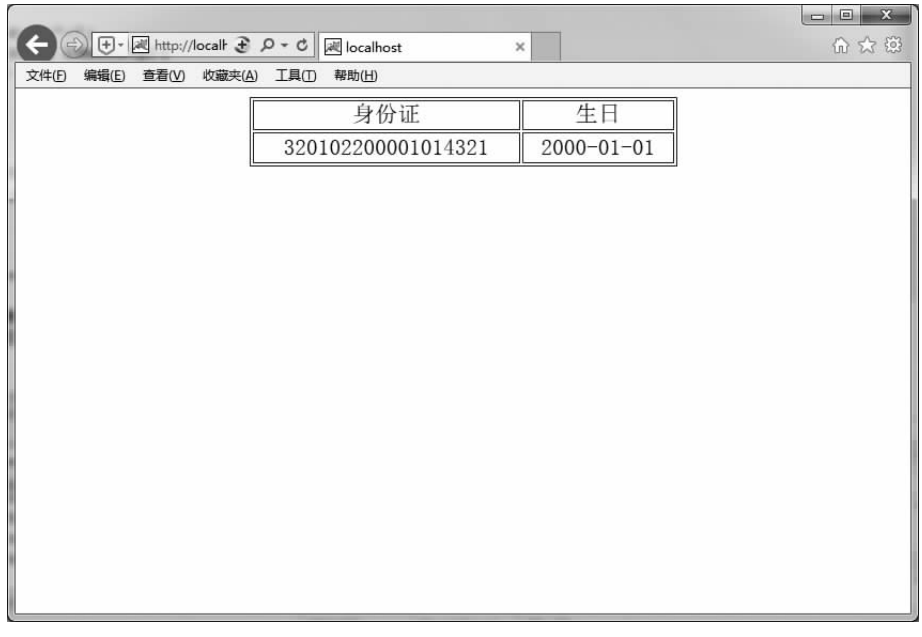

图 2-5 上机实践 4 的输出结果

3 模块 3

JSP 内置对象

知识目标

- 掌握 request 对象、response 对象、out 对象、session 对象、application 对象、pageContext 对象和 exception 对象的基本用法。
- 了解 page 对象和 config 对象。
- 掌握 Cookie 类的基本用法。
- 掌握 param 动作、include 动作和 forward 动作的基本用法。

技能目标

- 能够在 MyEclipse IDE 中用 request 对象、response 对象、out 对象、session 对象、application 对象、pageContext 对象和 exception 对象编写 JSP 页面并运行。
- 能够在 MyEclipse IDE 中用 Cookie 类编写 JSP 页面并运行。
- 能够在 MyEclipse IDE 中用 param 动作、include 动作和 forward 动作编写 JSP 页面并运行。

3.1 回顾和思考

在模块 2 中我们学习并分析了 JSP 页面代码的构成，能够设计编写简单的 JSP 页面。本模块我们将系统地学习 JSP 内置对象及部分 JSP 动作元素，从而使用 JSP 处理客户端请求，并与用户进行信息交互。

【例 3-1】 下面是网页制作课程中设计的一个用户登录表单示例。

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<html>
<head>
  <title>example3_1.jsp</title>
</head>
<body>
  <form id="form" name="form" method="post" action="example3_2.jsp">
    <center><p>用户名:<input type="text" name="user" maxlength="12"/></p>
</center>
    <center><p>密   码:<input type="password" name="pwd" maxlength="8"/>
</p></center>
    <center><p>     
  <input type="submit" name="button" value="提交"/>  
  <input type="reset" name="button" value="取消"/></p></center>
  </form>
</body>
</html>
```

【例 3-1】被命名为 example3_1.jsp，运行结果如图 3-1 所示。回忆一下在网页制作课程中，HTML 标签 form 的属性 action 是暂不赋值的，因为那时不需要对表单的内容进行处理；而在例【3-1】中，属性 action 被赋值为“example3_2.jsp”。这就告诉我们 example3_1.jsp 的处理页面为 example3_2.jsp，在 example3_1.jsp 页面中填写的信息将由 example3_2.jsp 页面提取并进行相应的处理。example3_2.jsp 页面该如何设计？其核心代码涉及的相关 JSP 内置对象需在本模块进行学习。

图 3-1 【例 3-1】运行结果

3.2 JSP 内置对象

JSP 内置对象是 JSP 容器加载的一组类的实例，它不需要通过“new”关键字去获取实例，可以直接在 JSP 页面使用。JSP 内置对象共有 9 个，分别是 request、response、out、session、application、pageContext、page、config 和 exception。

3.2.1 request 对象

【例 3-2】 用户信息输出。

首先编写 example3_2.jsp 页面，代码如下。

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<%
  request.setCharacterEncoding("GBK");
  String name = request.getParameter("user");
  String pwd = request.getParameter("pwd");
 %>
<html>
<head>
  <title> example3_2.jsp</title>
</head>
<body>
  <div align="center">你输入的身份信息
  <table width="200" border="1" align="center">
  <tr>
    <td align="left">用户名:<% = name %></td>
  </tr>
  <tr>
    <td align="left">密   码:<% = pwd %></td>
  </tr>
  </table>
  </div>
</body>
</html>
```

在本例中出现了内置对象 request。request 内置对象用于处理客户端的请求，包含所有

来自浏览器的请求信息。在 JSP 页面中通过 request 内置对象的方法来获取请求的相关数据。在本例中出现的 request 对象方法 void setCharacterEncoding(String charset)是指定请求的数据编码为中文,以防中文乱码;String getParameter(String name)则根据页面表单组件名称获取请求页面提交数据。

【例 3-2】中的 example3_2.jsp 页面通过 getParameter 方法获取 example3_1.jsp 页面表单组件 text 和 password 中用户输入的值,并用 JSP 表达式在页面相关位置输出。【例 3-2】运行结果如图 3-2 所示。

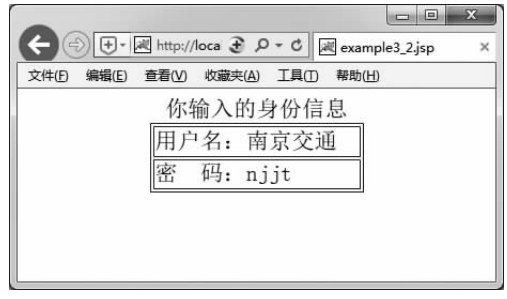

图 3-2 【例 3-2】运行结果

注意:【例 3-1】中表单信息的发送采用了 post 方式。post 方式将表单的内容通过 http 发送,在地址栏中看不到表单的提交内容。使用 post 方式发送信息没有字符长度的限制。表单信息发送的另一种方式为 get 方式。get 方式将表单的内容编码后,通过 url 发送,在地址栏能看到表单的提交内容。使用 get 方式发送信息有 255 个字符长度的限制。

在【例 3-2】中,通过 request 内置对象的 getParameter 方法获取的表单组件的值是单个的。如果遇到能提供多个值的表单组件(如复选框),该怎么办?

【例 3-3】 用户信息提交。

表单页面 example3_3.jsp 的代码如下。

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<html>
<head>
  <title>example3_3.jsp</title>
</head>
<body>
  <form id="form" name="form" method="post" action="example3_4.jsp">
    <center><p>爱好:<input type="checkbox" name="ah" value="音乐"/>音乐
    <input type="checkbox" name="ah" value="棋艺"/>棋艺
    <input type="checkbox" name="ah" value="书法"/>书法
    <input type="checkbox" name="ah" value="美术"/>美术</p></center>
```

```
    <center><p><input type="submit" name="button" value="提交"/>
    <input type="reset" name="button" value="取消"/></p></center>
  </form>
</body>
</html>
```

【例 3-3】运行结果如图 3-3 所示(复选框全部选中)。该表单只有一个复选框组件,单击“提交”按钮后,页面的处理权交给了 example3_4.jsp。那么该如何编写 example3_4.jsp 页面呢?

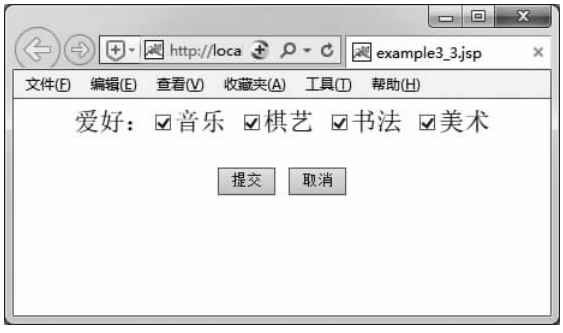

图 3-3 【例 3-3】运行结果

【例 3-4】 用户信息输出。

编写 example3_4.jsp 页面,代码如下。

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<%
  request.setCharacterEncoding("GBK");
  String[] ah = request.getParameterValues("ah");
  String s = "";
    if(ah != null)
      for(int i = 0;i < ah.length;i++)
        s = s + ah[i] + " ";
%>
<html>
<head>
  <title>example3_4.jsp</title>
</head>
<body>
  <div align="center">爱好
```

```
    <table width="240" border="1" align="center">
    <tr>
      <td align="center"><% = s %></td>
    </tr>
    </table>
    </div>
</body>
</html>
```

在【例 3-4】中，request 内置对象调用其方法 String[] getParameterValues(String name)来获取请求的复选框组件数据。该方法返还一个字符串数组，从而解决了多值表单组件的数据请求处理问题。【例 3-4】运行结果如图 3-4 所示。

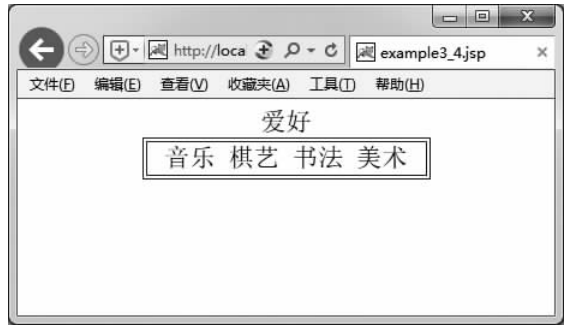

图 3-4 【例 3-4】运行结果

可以发现通常情况下登录后常用的页面处理流程是：用户进行登录，如果用户名和密码都正确，则通过验证，进入登录成功页面；反之，则进入登录失败页面。要实现上述功能，需要掌握利用 request 内置对象实现页面转发的方法。

继续使用【例 3-1】的用户登录界面，属性 action 的值换为“example3_5.jsp”。

【例 3-5】 用户登录后页面转发。

编写 example3_5.jsp 页面，其代码如下。

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<%
  request.setCharacterEncoding("GBK");
  String name = request.getParameter("user");
  String pwd = request.getParameter("pwd");
  if(name.equals("南京交通") && pwd.equals("njjt"))
  request.getRequestDispatcher("example3_5_1.jsp").forward(request,response);//登录成功
  else request.getRequestDispatcher("example3_5_2.jsp").forward(request,response);//登录失败
 %>
```

将登录成功页面命名为 example3_5_1.jsp，其代码如下。

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<html>
<head>
  <title>example3_5_1.jsp</title>
</head>
<body>
  <h1 align="center"><font size="4">登录结果</font></h1>
  <p align="center"><font color="#FF0000">登录成功！</font></p>
</body>
</html>
```

将登录失败页面命名为 example3_5_2.jsp，其代码如下。

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<html>
<head>
  <title>example3_5_2.jsp</title>
</head>
<body>
  <h1 align="center"><font size="4">登录结果</font></h1>
  <p align="center"><font color="#FF0000">登录失败！</font></p>
</body>
</html>
```

在【例 3-5】中，request 内置对象首先调用其 RequestDispatcher getRequestDispatcher("url") 方法生成一个 RequestDispatcher（request 转发器）对象，再调用该 RequestDispatcher 对象的 void forward(ServletRequest arg0，ServletResponse arg1)方法完成页面的转发。【例 3-5】运行结果如图 3-5 和图 3-6 所示。

图 3-5 【例 3-5】登录成功运行结果

图 3-6 【例 3-5】登录失败运行结果

注意: JSP 中的内置对象有很多方法。在 MyEclipse 8.5 的代码编辑窗口中输入内置对象名称后,再输入点操作符就可以查看到对应内置对象所有方法的 API 帮助。

3.2.2 response 对象

response 内置对象与 request 内置对象相对应,用于响应客户端请求并向客户端输出信息。

在【例 3-5】中使用 request 内置对象实现了页面的转发。同样可以使用 response 内置对象实现页面的跳转或重定向(这里不使用“转发”术语,而使用“跳转或重定向”术语,为什么?)。继续使用【例 3-1】的用户登录界面,将表单属性 action 的值更换为“example3_6.jsp”。

【例 3-6】 用户登录后页面跳转。

example3_6.jsp 页面代码如下。

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<%
  request.setCharacterEncoding("GBK");
  String name = request.getParameter("user");
  String pwd = request.getParameter("pwd");
  if(name.equals("南京交通") && pwd.equals("njjt"))
    response.sendRedirect("example3_6_1.jsp");
  else
    response.sendRedirect("example3_6_2.jsp");
%>
```

example3_6_1.jsp 和 example3_6_2.jsp 可分别参考 example3_5_1.jsp 和 example3_5_2.jsp,此处代码略去。【例 3-6】运行结果可参考图 3-5。

注意: response 内置对象的 sendRedirect("url")方法与 request 转发器的 void forward(request,response)方法的差别在于:使用 sendRedirect("url")方法实现页面跳转后,客户端重新建立了链接,url 地址发生了改变;使用 void forward(request,response)方法实现页

面转发后，客户端并没有建立新的链接，url 地址没有改变。后者亦可称为派遣访问。

【例 3-7】 可以利用 response 内置对象的 void setHeader(String name, String value) 方法在网页中实时显示系统时间。该方法的第一个形参为 http 响应报头名，第二个形参为 http 响应报头值。本例将使用 Refresh 响应报头，响应报头值设为 1 秒。文件名为 example3_7.jsp，其代码如下。

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<%@ page import="java.text.*,java.util.*" %>
<html>
<body>
<%!
  String formatDate(Date d, String format){
    SimpleDateFormat formater = new SimpleDateFormat(format);
    return formater.format(d);
  }
%>
<center>
你好，交通信息工程学院！现在是：
<%= formatDate(new Date(),"yyyy 年 MM 月 dd 日 HH 时 mm 分 ss 秒") %>
<%
  response.setHeader("Refresh","1");
%>
</center>
</body>
</html>
```

【例 3-7】运行结果如图 3-7 所示。通过每隔 1 秒的页面刷新，实时显示了系统时间。

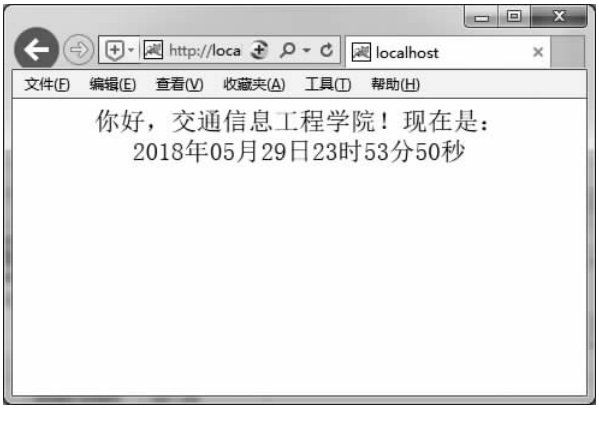

图 3-7 【例 3-7】运行结果

3.2.3 out 对象

out 内置对象是 JSP 编程中使用最为频繁的内置对象。out 内置对象用于向客户端输出数据。out 内置对象常用的方法有 void print(String output)、void println(String output)和 void write(String output)。这三个方法都用于向页面相应位置输出数据。

【例 3-8】 out 内置对象应用实例。

编写 example3_8.jsp,其代码如下。

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<%@ page import="java.text.*,java.util.*" %>
<html>
<body>
<%!
  String formatDate(Date d, String format){
    SimpleDateFormat formater = new SimpleDateFormat(format);
    return formater.format(d);
  }
%>
<center>
你好,交通信息工程学院! 现在是:
<%
  out.print(formatDate(new Date(),"yyyy 年 MM 月 dd 日 HH 时 mm 分 ss 秒"));
  out.print("<br>");//换行
%>
你好,运输管理工程学院! 现在是:
<%
  out.write(formatDate(new Date(),"yyyy 年 MM 月 dd 日 hh 时 mm 分 ss 秒"));
%>
</center>
</body>
</html>
```

在 example3_8.jsp 的代码中,用 out 内置对象代替了 JSP 表达式,收到了同样的效果。【例 3-8】运行结果如图 3-8 所示。

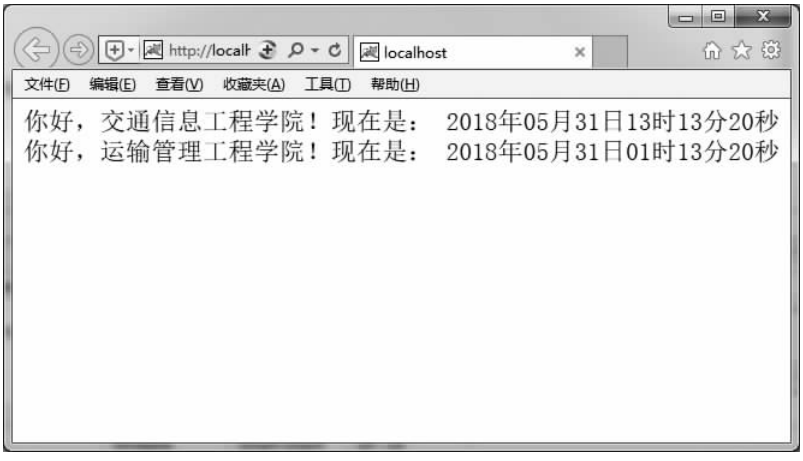

图 3-8 【例 3-8】运行结果

3.2.4 session 对象

session 内置对象用于存储用户会话的所有信息，以此识别不同的用户。session 内置对象在页面发生跳转时，能保存并跟踪用户的会话信息。session 内置对象最常用的方法是 void setAttribute(String name, Object value)和 Object getAttribute(String name)。前者设定指定名称的属性及其值，并把该属性连同它的值存储在 session 内置对象中；后者根据属性的名称获取其存储在 session 内置对象中的值。

【例 3-9】 使用 session 内置对象显示会话计数。

编写 example3_9.jsp，其代码如下。

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<%
  if(session.getAttribute("count") == null)
    session.setAttribute("count",new Integer(0));
  Integer count = (Integer)session.getAttribute("count");
  session.setAttribute("count",new Integer(count.intValue() + 1));
%>
<html>
<body>
  <div align="center">使用 session 内置对象显示会话计数</div>
  <table width="400" border="1" align="center">
    <tr>
      <td width="200" align="center"><b>会话计数</b></td>
      <td width="200" align="center"><% = session.getAttribute("count") %></td>
```

```
    </tr>
  </table>
</body>
</html>
```

【例 3-9】运行结果如图 3-9 所示。会话计数从 1 开始。选择刷新页面、新建选项卡、新建窗口等操作，会话计数增加 1 次。

图 3-9 【例 3-9】运行结果

3.2.5 application 对象

服务器中的 JSP 引擎启动后，application 内置对象就产生了。不同用户浏览网站的不同页面时，application 内置对象都是同一个，即所有客户端共享此 application 内置对象。直至 JSP 引擎关闭，此 application 内置对象才消亡。

application 内置对象最常用的方法是 void setAttribute(String name, Object value)和 Object getAttribute(String name)。前者设定指定名称的属性及其值，并把该属性连同它的值存储在 application 内置对象中；后者根据属性的名称获取其存储在 application 内置对象中的值。

【例 3-10】 使用 application 内置对象显示应用程序计数。

编写 example3_10.jsp，其代码如下。

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<%
  if(application.getAttribute("count") == null)
    application.setAttribute("count",new Integer(0));
  Integer count = (Integer)application.getAttribute("count");
  application.setAttribute("count",new Integer(count.intValue() + 1));
%>
```

```
<html>
<body>
  <div align="center">使用 application 内置对象显示应用程序计数</div>
  <table width="400" border="1" align="center">
    <tr>
      <td width="200" align="center"><b>应用程序计数</b></td>
      <td width="200" align="center"><% = application.getAttribute("count") %></td>
    </tr>
  </table>
</body>
</html>
```

【例 3-10】运行结果如图 3-10 所示。应用程序计数从 1 开始。选择刷新页面、新建选项卡、新建窗口、新建会话、重新打开浏览器等操作，应用程序计数增加 1 次。关闭 Tomcat，再重启 Tomcat，应用程序计数则又从 1 开始。

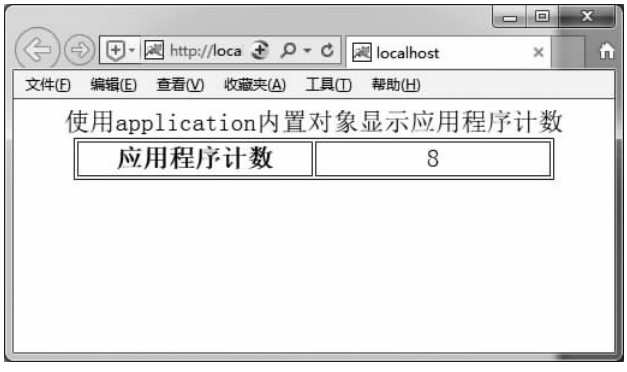

图 3-10 【例 3-10】运行结果

3.2.6 pageContext 对象

用户使用 pageContext 内置对象可以访问本页面中其他所有内置对象。pageContext 内置对象的作用范围仅限于页面内。pageContext 内置对象最常用的方法是 void setAttribute(String name, Object value)和 Object getAttribute(String name)。前者设定指定名称的属性及其值，并把该属性连同它的值存储在 pageContext 内置对象中；后者根据属性的名称获取其存储在 pageContext 内置对象中的值。

【例 3-11】 使用 pageContext 内置对象显示页面计数。

编写 example3_11.jsp，其代码如下。

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<%
  if(pageContext.getAttribute("count") == null)
    pageContext.setAttribute("count",new Integer(0));
  Integer count = (Integer)pageContext.getAttribute("count");
  pageContext.setAttribute("count",new Integer(count.intValue() + 1));
%>
<html>
<body>
  <div align="center">使用 pageContext 内置对象显示页面计数</div>
  <table width="400" border="1" align="center">
    <tr>
      <td width="200" align="center"><b>会话计数</b></td>
      <td width="200" align="center"><% = pageContext.getAttribute("count") %></td>
    </tr>
  </table>
</body>
</html>
```

【例 3-11】运行结果如图 3-11 所示。页面计数从 1 开始。选择刷新页面、新建选项卡、新建窗口、新建会话、重新打开浏览器等操作，页面计数始终为 1。

图 3-11 【例 3-11】运行结果

3.2.7 page 对象

page 内置对象提供对页面上定义的所有对象的访问。page 内置对象表示 JSP 页面本身。在 JSP 中很少使用 page 内置对象，一般使用 page 指令替代它。

3.2.8 config 对象

config 内置对象存储 Servlet 的一些初始信息，如读取 web.xml 配置信息等。与 page

内置对象一样，在 JSP 中很少被使用。

3.2.9 exception 对象

我们回顾模块 2 上机实践的第 2 题。首先编写一个产生异常的 JSP 页面 lab2_2.jsp，其代码如下。

```
<%@ page language="java" contentType="text/html;charset=GBK" errorPage="error.jsp"%>
<html>
<body>
<%
  int i = 0;//除数为 0
  int j = 10 / i;
  out.print("j=" + j);
%>
</body>
</html>
```

此页面在执行时会发生除数为 0 的错误，其后代码的执行会终止，程序会转至 page 指令中 errorPage 属性指定的当前页面的错误处理页面 error.jsp。error.jsp 页面代码如下。

```
<%@ page language="java" contentType="text/html;charset=GBK" isErrorPage="true"%>
<html>
<body>
<%
  out.print("出错原因是" + exception.getMessage());
%>
</body>
</html>
```

该页面中 page 指令的 isErrorPage 属性值为 true，即表示当前页面为错误处理页面。可在此页面上使用 exception 内置对象，以捕捉导致控制权转至错误处理页面的错误。String getMessage()方法返回异常消息字符串。

3.3 Cookie

Cookie 是 Web 服务器暂存在客户端浏览器内(有效期短)或硬盘中(有效期长)的少量

文本数据。当用户再次访问某个 Web 站点时，Web 服务器要求客户端浏览器查找并返回先前发送的 Cookie，从而达到令 Web 服务器快速识别用户及其请求的目的。

Cookie 类位于 javax.servlet.http 包中。可以通过创建 Cookie 类的实例，然后将其与 request 内置对象和 response 内置对象结合使用，以实现将 Cookie 信息传送到客户端或获取客户端 Cookie 信息等功能。

【例 3-12】 使用 Cookie 实现 30 秒内保留登录的用户名和密码功能。

编写 example3_12.jsp，其代码如下。

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<%@ page import="java.io.*,java.net.*" %>
<%
  request.setCharacterEncoding("GBK");
  String name = request.getParameter("user");//获取用户名
  String pwd = request.getParameter("pwd");//获取密码
  Cookie cookies[] = request.getCookies();//获取所有的 Cookie
  if(name != null){
    try{
      name = URLEncoder.encode(name,"UTF-8");//用户名中文编码
    }catch(UnsupportedEncodingException e){}
    Cookie c = new Cookie("user",name);//以上次登录的用户名为新建 Cookie
    c.setMaxAge(30);//Cookie 的有效期为 30 秒
    response.addCookie(c);//写入 Cookie
  }
  else if(cookies != null)//如果 Cookie 在有效期内
    for(int i = 0; i < cookies.length; i++)
      if(cookies[i].getName().equals("user"))
        try{
          name = URLDecoder.decode(cookies[i].getValue(),"UTF-8");}//用户名中文解码
        catch(UnsupportedEncodingException e){}
  if(pwd != null){
    Cookie c = new Cookie("pwd",pwd);
    c.setMaxAge(30);
    response.addCookie(c);
  }
  else if(cookies != null)
    for(int i = 0; i < cookies.length; i++)
      if(cookies[i].getName().equals("pwd"))
```

```
            pwd = cookies[i].getValue();
    %>
    <html>
    <head>
      <title>example3_12.jsp</title>
    </head>
    <body>
      <h2 align="center">使用 Cookie 保存信息</h2>
      <hr>
      <form method="post" action="">
        <center><p>用户名:
          <input type="text" size="20" name="user" value="<% if(name != null){try{
          name = URLDecoder.decode(name,"UTF-8");
        }catch(UnsupportedEncodingException e){}out.println(name);} %>">
        </p></center>
        <center><p>密   码:
          <input type="password" size="20" name="pwd" value="<% if(pwd != null)out.
println(pwd); %>">
        </p></center>
        <center><p>
          <input type="submit" value="提交">
        <input type="reset" value="取消">
        </p></center>
      </form>
    </body>
    </html>
```

运行 example3_12.jsp 后输入用户名和密码，单击“提交”按钮，随后关闭浏览器。再次运行该页面，存储在 Cookie 中的用户名和密码显示如图 3-12 所示。

图 3-12 【例 3-12】运行结果

3.4 JSP 动作元素

JSP 动作元素也称 JSP 标准动作，简称 JSP 动作。利用 JSP 动作可实现页面之间控制权的灵活转移。与 JSP 指令在编译时即被 JSP 引擎执行不同，JSP 动作元素在客户端发生请求时才被执行。JSP 动作元素包括 jsp:param、jsp:include、jsp:forward、jsp:useBean、jsp:setProperty、jsp:getProperty 等。此处将讲述 jsp:param 动作、jsp:include 动作和 jsp:forward 动作。对 jsp:useBean、jsp:setProperty 动作和 jsp:getProperty 动作，将在下一模块与 JavaBean 结合起来学习。

3.4.1 jsp:param 动作

param 动作的语法格式如下。

```
<jsp:param name="参数名" value="参数值">...</jsp:param>
```

如果起始标记和结束标记之间没有内容，则上述语法格式可简化为

```
<jsp:param name="参数名" value="参数值"/>
```

jsp:param 动作通常与 jsp:include 动作、jsp:forward 动作和 jsp:setProperty 动作等一起使用。

3.4.2 jsp:include 动作

include 动作用于将其他 HTML 文件或 JSP 文件合并到当前页面文件。include 动作的语法格式如下。

```
<jsp:include page="文件名" flush="true">...</jsp:include >
```

其中，page 属性的值规定了所嵌入文件的相对路径，flush 属性的作用是在嵌入其他响应前清空存储在缓冲区中的数据，一般设为 true。

如果起始标记和结束标记之间没有参数内容，则上述语法格式可简化为

```
<jsp:include page="文件名" flush="true"/>
```

【例 3-13】 在页面中嵌入 example2_3.jsp 页面。

文件名为 example3_13.jsp，其代码如下。

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<html>
<head>
```

```
    <title>jsp:include 动作不带参数</title>
</head>
<body>
<jsp:include page="//ch2/example/example2_3.jsp" flush="true"/>
</body>
</html>
```

因为嵌入的是 example2_3.jsp 页面，所以 Page 属性的值为 example2_3.jsp 所处的绝对路径。【例 3-13】运行结果如图 3-13 所示。

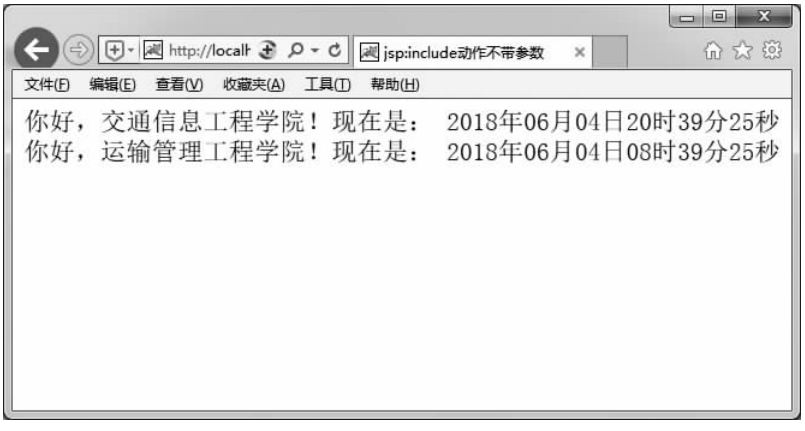

图 3-13 【例 3-13】运行结果

再看一个带参数的 include 动作的例子。

【例 3-14】 在页面中嵌入 example3_2.jsp 页面。

文件名为 example3_14.jsp，其代码如下。

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<html>
<head>
  <title>jsp:include 动作带参数</title>
</head>
<body>
  <%
    request.setCharacterEncoding("GBK");
  %>
  <jsp:include page="example3_2.jsp" flush="true">
  <jsp:param value="南京交通" name="user"/>
  <jsp:param value="njjt" name="pwd"/>
  </jsp:include>
```

```
</body>
</html>
```

与【例 3-2】非常相似,【例 3-14】运行结果如图 3-14 所示。

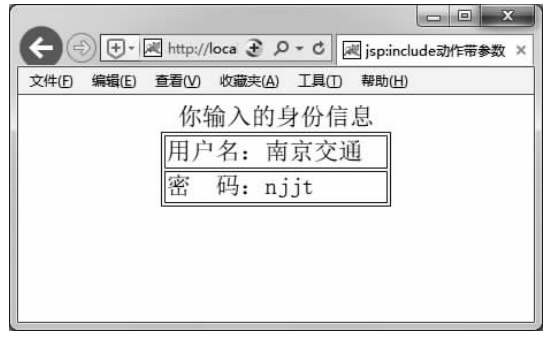

图 3-14 【例 3-14】运行结果

3.4.3 jsp:forward 动作

forward 动作用于将用户的请求重定向到另一个 HTML 页面、JSP 页面或 Servlet。forward 动作的语法格式如下。

```
<jsp:forward page="url">...</jsp:forward>
```

其中,page 属性指定目标页面的网址。

如果起始标记和结束标记之间没有参数内容,则上述语法格式可简化为

```
<jsp:forward page="url"/>
```

【例 3-15】 用 forward 动作改写【例 3-5】中 example3_5.jsp 页面的相应代码,以达到相似的运行效果。

文件名为 example3_15.jsp,其代码如下。

```
<%-- jsp:forward 动作 --%>
<%@ page language="java" contentType="text/html;charset=GBK" %>
<html>
<body>
<%
  request.setCharacterEncoding("GBK");
  String name = request.getParameter("user");
  String pwd = request.getParameter("pwd");
  if(name.equals("南京交通") && pwd.equals("njjt")){
%>
```

```
        <jsp:forward page="example3_5_1.jsp"/><%-- 登录成功 --%>
    <%
      }
      else{
    %>
        <jsp:forward page="example3_5_2.jsp"/><%-- 登录失败 --%>
    <%
      }
    %>
    </body>
    </html>
```

【例 3-1】中 example3_1.jsp 页面的处理页面更改为 example3_15.jsp。运行 example3_1.jsp 页面，输入相关数据验证。【例 3-15】运行结果参考图 3-5 和图 3-6。与【例 3-5】类似，用 forward 动作实现页面跳转后，客户端并没有建立新的链接，url 地址没有改变，仍保留为 http://localhost:8080/myPro/ch3/example/example3_15.jsp。

3.5 小　结

我们学习了 9 个 JSP 内置对象 request、response、out、session、application、pageContext、page、config 和 exception 及其最常用的方法；我们还学习了 Cookie 的使用及 jsp:param 动作、jsp:include 动作和 jsp:forward 动作的使用。利用 JSP 开发技术可以着手处理客户端请求，从而与用户进行信息交流。

3.6 习　题

1. 编写一个页面输出单选试题，再编写一个信息处理页面获取用户做出的选择，并统计得分。其运行结果如图 3-15 和图 3-16 所示。

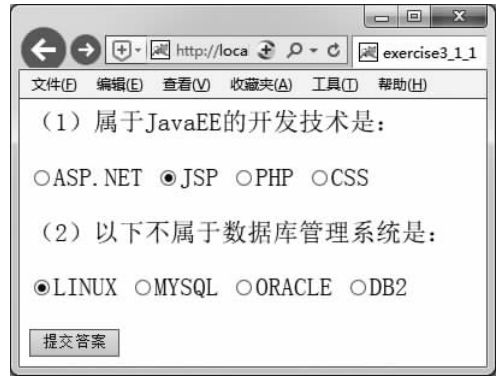

图 3-15　习题 1 图 1

图 3-16　习题 1 图 2

2. 在【例 3-7】中，我们学习了通过 response 内置对象的 setHeader 方法设置每隔 1 秒的页面刷新，实时显示了系统时间。对【例 3-7】做一些改动，实现页面 5 秒后转至 http://www.njitt.edu.cn。

3. 分析以下 JSP 页面代码，推算并验证运行结果。思考一下为什么会有这样的结果。

(1)

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<%
  int choice = 1;
  if(choice == 1){
%>
<%@ include file="exercise3_3_1.jsp" %>
<%
  }else{
%>
<%@ include file="exercise3_3_2.jsp" %>
<%
  }
```

```
%>
<%= "你好," + s1 %><br>
<%= "你好," + s2 %>
```

exercise3_3_1.jsp 代码如下。

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<%!
  String s1 = "交通信息工程学院!";
%>
```

exercise3_3_2.jsp 代码如下。

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<%!
  String s2 = "运输管理工程学院!";
%>
```

(2)

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<%
  int choice = 1;
  String included = "";
  if(choice == 1)
    included = "exercise3_3_1.jsp";
  else
    included = "exercise3_3_2.jsp";
%>
<jsp:include page="<%= included %>"/>
<%= "你好," + s1 %>
```

4. 分析以下 JSP 页面代码，推算并验证运行结果。思考一下为什么会有这样的结果。如果难以判断，可在 out 内置对象前添加“System.”，页面运行后打开 MyEclipse 控制台查看结果。

(1)

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<%
  out.println("执行 forward 动作之前。");
%>
<jsp:forward page="exercise3_4c.jsp"/>
```

```
<%
  out.println("执行 forward 动作之后。");
%>
```

exercise3_4c.jsp 代码如下。

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<%
  out.print("这是 exercise3_4c.jsp 页面。" + "<br>");
  out.print("url 改变了吗?");
%>
```

(2)

```
<%@ page language="java" contentType="text/html;charset=GBK" %>
<%
  out.println("执行 forward 方法之前。");
  response.sendRedirect("exercise3_4c.jsp");
  out.println("执行 forward 方法之后。");
%>
```

3.7 上机实践

1. 编写一个图 3-17 所示的表单页面，页面信息提交后的输出结果如图 3-18 所示。

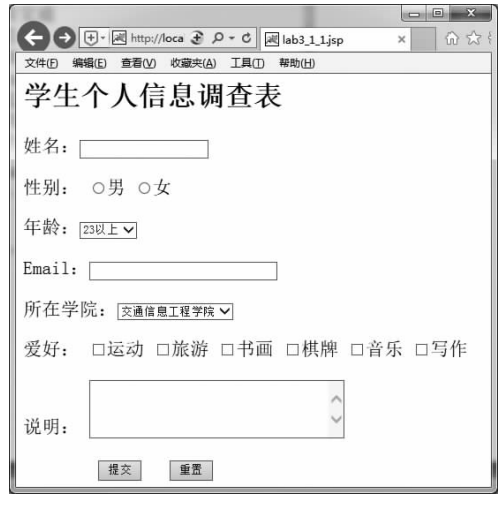

图 3-17 上机实践 1 的输出结果 1

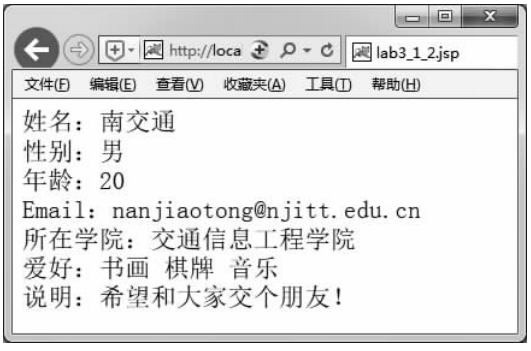

图 3-18 上机实践 1 的输出结果 2

2. 使用 Cookie 实现 30 秒内用户免登录功能，跳转至登录成功页面。

提示：参考【例 3-12】，修改部分功能。

3. 用 session 内置对象和 application 内置对象设计一个统计网站访问次数的页面，要求用户无法通过刷新页面来提高网页的访问次数。

提示：用 session 内置对象的 isNew()方法判断访问用户是否为新用户。

4. 实现一个简单的购物车功能。要求列出商品信息；当用户选中某件商品，单击“提交”按钮时，就把该商品加入购物车中；在购物车中既可以看到已订购的商品，也可以继续购物或清空购物车。

提示：把商品存放在一个数组中，再用 StringTokenizer 类的相关方法析出。

首先设计一个商品展示页面，如图 3-19 所示。

图 3-19 商品展示页面

选中想购买的商品，单击“提交”按钮后，进入已购商品显示页面，如图 3-20 所示。

在该页面可以选择单击“继续购买”超链接返回商品展示页面继续购物，如图 3-21 所示。

图 3-20　已购商品显示页面

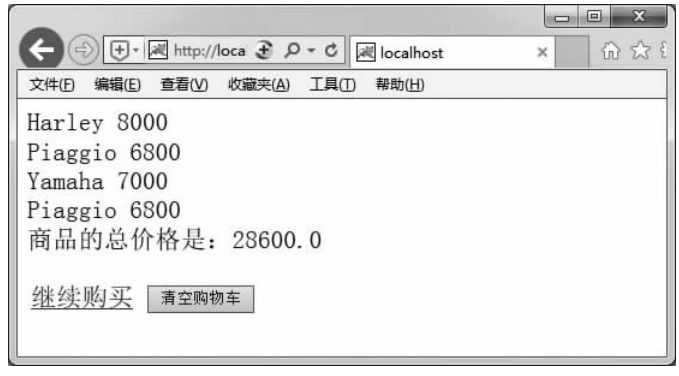

图 3-21　继续购物后的商品显示页面

也可以选择单击“清空购物车”按钮，如图 3-22 所示。

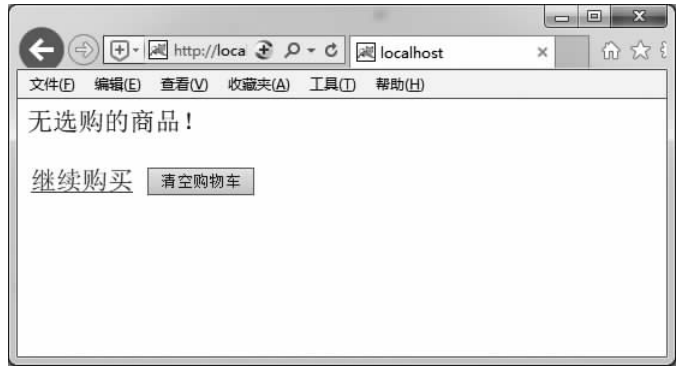

图 3-22　清空购物车后的页面