第 1 篇　JSP 基本技术

第 1 章　JSP 运行环境

1.1　动态网页技术

Internet 的传统应用有：远程登录(Telnet)、文件传输(FTP)、Web 应用(HTTP)、电子邮件(E-mail)、网络聊天(NetChat)、网络新闻(NetNews)等，其中电子邮件曾经是使用最广泛的应用，而目前 Web 应用在 Internet 上是最流行的。Web 是 World Wide Web(WWW)环球信息网或万维网的简称，是一张附着在 Internet 上的覆盖全球的信息“蜘蛛网”，镶嵌着无数以超文本形式存在的信息。Web 成为人们共享信息的主要手段，WWW 几乎成为 Internet 的代名词。Web 应用由 Web 服务器发布，客户端用浏览器(如 IE、Navigator 等)进行浏览，使用 HTTP 通过 Internet 进行信息传输。Web 的发展可分为三个阶段。

1.1.1　Web 发展的三个阶段

(1)静态网页：早期单纯以 HTML 编写的网页。静态页面的请求处理过程比较简单。

静态网页以 HTML 语言编写，保存在 Web 服务器上，客户端浏览器根据用户输入的网址向服务器发出请求，服务器接受浏览器的请求后，查找所请求的页面文件，并进行权限验证，如果验证通过，将该网页发回给浏览器显示。请求与应答在网络上使用的传输协议为 HTTP，如图 1-1 所示。

图 1-1　静态网页的请求处理过程

网页的请求处理过程有两个重要的参与者，客户端的浏览器和服务端的 Web 服务器，可以说这两个软件是 Web 技术的核心体现。浏览器的主要功能是发起 HTTP 请求，解析与显示 HTML 网页。进一步讲，浏览器是客户端应用层协议 HTTP 的实现者和 HTML 标记解析器。目前常用的浏览器有：Microsoft(微软)的 IE(Internet Explorer)、Netscape(网景)的 NN(Netscape Navigator)、Mozilla 基金会的 Firefox(火狐狸)、傲游的 Maxthon、凤凰工作室(Phoenix Studio)的 The World(世界之窗)、腾讯的 TT(Tencent Traveler)。Web 服务器又称 HTTP 服务器，是典型的实现应用层协议 HTTP 的软件。Web 服务器的主要功能是处理 HTTP 请求，管理 Web 页面。评价 Web 服务器的因素有：承载力、效率、稳定性、安全性、日志和统计、虚拟主机、代理服务器、缓存服务和集成应用程序等。常见的 Web 服务器有：Microsoft(微软)的 IIS(Internet Information Server)，运行于 Windows 平台，支持 ASP 及 ASP.NET；The Apache Software Foundation 的 Apache，是最流行的 HTTP 服务器，早期运行于 UNIX 类系统上，目前也有运行于 Windows 系统的版本；IBM 的 Websphear，是功能完善、开放的大型 Web 应用程序服务器，支持 JSP； W3C(World Wide Web Consortium)的 Jagsaw；使用 Java 语言，采用完全的面向对象架构，以最新的 Web 技术协议为标

准设计的开放源码服务器，支持 JSP；AOL(America On Line，美国在线)的 AOL Server，是高效能、高稳定性、高扩充性的开源 Web 服务器，运行于 UNIX 类系统平台。Kerio 的 WebSTAR，Kerio 是专业生产防火墙的厂家，WebSTAR 运行于 Apple(苹果)Mac OS 平台，其特点是高安全性。

(2)客户端动态网页：以 DHTML 和其他客户端交互技术编写的网页，DHTML(Dynamic HTML)是一种通过结合 HTML、客户端脚本语言(JavaScript、VBScript)、层叠样式表(CSS)和文档对象模型(DOM)来创建动态网页内容的技术总称，其他客户端交互技术有：Flash、ActiveX、Java Applet 等。客户端动态技术需浏览器的支持，对浏览器的可扩展功能有了更高的要求，此时的浏览器不仅仅是一个标记解析器，还嵌入了脚本解析等各种功能模块。客户端动态网页的请求处理过程基本与早期的静态网页一致。

(3)Web 应用程序：即服务器端动态网页，浏览器请求服务器端动态的网页时，服务器必须调用相应的解析器对页面进行处理，一般是运行其中嵌入的程序代码，将处理结果返回给浏览器显示，其处理过程如图 1-2 所示。

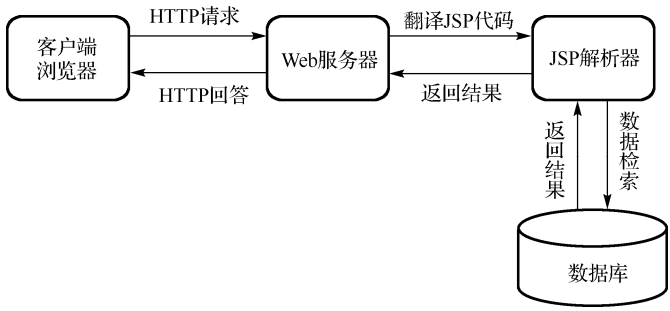

图 1-2 动态网页的请求处理过程

服务器端动态技术需要 Web 服务器的支持，对 Web 服务器的技术发展提出了新的要求，不同的服务器端动态网页设计技术要有特定的 Web 服务器，或者在 Web 服务器上安装特定的功能模块来支持。Web 应用程序的发展也经历了三个阶段。

1.1.2 Web 应用程序开发的三个阶段

(1)代码输出：最初的服务器端动态网页技术是 CGI(Common Gateway Interface，通用网关接口)，CGI 是 Web 服务器支持的允许客户端浏览器请求调用服务器上特定程序的技术，这个程序又称为 CGI 程序，它接收客户端提交的数据，CGI 描述了浏览器和所请求程序之间传输数据的标准；一般输出 HTML 代码给浏览器，所以称这样的动态网页设计为代码输出。CGI 程序以编译后可执行代码的形式发布，所以 CGI 程序是语言独立的，可以用任何编程语言实现，Perl 是最广泛使用的 CGI 程序设计语言。

(2)代码混合：CGI 程序要输出各种 HTML 标记，编程很复杂。所以发展了在 HTML 文件中使用服务器端标记嵌入服务器端代码的网页，请求这些网页时，服务器调用特定的解析器(程序)对其进行处理，将其中的 HTML 标记直接输出，执行其中的服务端代码，在网页中代码所在处输出运行的结果。这样的动态网页中 HTML 标记与服务器端代码混合在一起，所以称为代码混合形式的 Web 应用程序开发。常用的代码混合动态网页编程技术有 ASP、PHP、JSP 三种。

ASP(Active Server Page)：ASP 是一种服务器端脚本编写环境，可以用来创建和运行动态网页或 Web 应用程序。ASP 网页可以包含 HTML 标记、普通文本、脚本命令以及 COM 组件等，ASP 的强大不在于它的脚本语言，而在于它后台的 COM 组件，这些组件无限地扩充了 ASP 的功

能。ASP 已经升级为 ASP.NET，ASP.NET 采用组件技术，以代码分离的方式来开发 Web 应用程序，已完全不同于 ASP。

PHP（PHP Hypertext Process，PHP 原指 Personal Home Page）：PHP 是一种跨平台的服务器端嵌入式脚本语言。它大量地借用 C、Java 和 Perl 语言的语法，并结合 PHP 自己的特性，使 Web 开发者能够快速地写出动态页面。它支持绝大部分数据库，而且是完全免费的。

JSP（Java Server Page）：JSP 是基于 Java 语言的一种 Web 应用开发技术，利用这一技术可以建立安全、跨平台的先进动态网站。利用 JSP 技术创建的 Web 应用程序，可以实现动态页面与静态页面的分离。与其他 Web 技术相比，JSP 具有跨平台、编译后运行等特点。

（3）代码分离：指页面的程序逻辑与表现相互分离的动态网页设计机制， ASP.NET 以服务器控件和页面模型抽象方式实现代码分离，JSP 以 Bean 和自定义标签的方式实现代码分离，以组件方式构建 Java Web 应用程序的技术为 JSF（Java Server Faces），其他语言大多以模板方式实现代码分离，如 PHP Template。动态网页设计技术正在迅速发展中，从代码分离、Web 组件到 Web Service。

1.1.3　HTTP 请求地址——URL

Web 服务器发布的资源在 Internet 上是以 URL（Uniform Resource Locator，统一资源定位器）标识的，客户端浏览器请求时，必须在地址栏中输入 URL，即必须提供所请求网页等目标的位置。URL 的格式为：

protocol://hostname[:port]/website/path/ [file][?query][#fragment]

协议://主机名:端口号/网站名称/目录/文件名?查询参数#信息片段

例如，http://www.lytu.edu.cn:80/chpage/index.html?str=abc#a1

protocol（协议）：http、ftp、file、gopher、https、mailto、news。

hostname（主机名）：机器名+域名+域树+域林。

port（端口号）：http 的默认端口为 80，可以省略。

其他常用协议的默认端口：telnet 为 23、ftp 为 21、smtp 为 25、pop3 为 110、dns 为 53。

website（网站名称）：Web 应用程序上下文、虚拟目录名、网站根目录。

path/file（目录/文件）：网页相对于网站根目录的子目录和文件名。

?query（查询参数）：?名 1=值 1&名 2=值 2。

#fragment（信息片段）：网页锚点，使用<a>标记 name 属性在网页内部定义的位置标记。只在同一应用程序上下文（同一网站内）有效。

1.1.4　HTTP 状态码

Web 服务器对客户端的响应一般包含：一个状态行、一些响应报头、一个空行和相应的内容文档。

（1）状态行：状态行由 HTTP 版本、一个状态代码以及一段对应状态代码的简短说明信息组成，表示请求是否被理解或被满足。HTTP 版本由服务器决定。请求被正常响应时，状态码一般由系统自动设置为 200。也可以在页面中用代码设置状态码，说明信息也可自定义。

（2）一些响应报头（几个应答头）：HTTP 头消息，对应于 HTTP 协议的头部，在大多数情况下，除了 Content-Type，所有应答头都是可选的。

（3）空行：起分隔、标识作用。

（4）内容文档：数据报内容，封装在 HTTP 协议的体内。

下面是一个最简单的应答：

HTTP/1.1 200 OK
Content-Type: text/plain

Hello World

HTTP 1.1 中的状态码见表 1-1，总体上分为五大类。

表 1-1 常见 HTTP 1.1 状态代码以及对应的状态信息和含义

代码	HttpServletResponse 符号常量	信 息	含 义
100	SC_CONTINUE	Continue	继续
101	SC_SWITCHING_PROTOCOLS	Switching Protocols	转换协议
200	SC_OK	OK	一切正常
201	SC_CREATED	Created	创建
202	SC_ACCEPTED	Accepted	接收
203	SC_NON_AUTHORITATIVE_INFORMATION	Non authoritative Information	非授权信息
204	SC_NO_CONTENT	No Content	无内容
205	SC_REST_CONTENT	Reset Content	重置内容
206	SC_PARTIAL_CONTENT	Partial Content	部分内容
300	SC_MULTIPLE_CHOICES	Multiple Choices	多选
301	SC_MOVED_PERMANENTLY	Moved Permanently	永久移动
302	SC_MOVED_PERMANENTLY	Moved Temporarily	暂时移动
304	SC_NOT_MODIFIED	Not Modified	未更改
305	SC_USE_PROXY	Use Proxy	使用代理服务器
400	SC_BAD_REQUEST	Bad Request	错误请求
401	SC_UNAUTHORIZED	Unauthorized	未授权
402	SC_PAYMENT_REQUIRED	Payment Required	要求支付
403	SC_FORBIDDEN	Forbidden	禁止
404	SC_NOT_FOUND	Not Found	未找到
405	SC_METHOD_NOT_ALLOWED	Method Not Allowed	不可用方法
406	SC_NOT_ACCEPTABLE	Not Acceptable	不接受
407	SC_PROXY_AUTHENTICATION_REQUIRED	Proxy Authentication Required	需确认代理服务器
408	SC_REQUEST_TIMEOUT	Request Time Out	请求超时
409	SC_CONFLICT	Conflict	冲突
410	SC_GONE	Gone	离开
411	SC_LENGTH_REQUIRED	Length Required	需要长度
412	SC_PRECONDITION_FAILED	Precondition Failed	预处理失败
413	SC_REQUEST_ENTITY_TOO_LARGE	Request Entity Too Large	请求实体过大
414	SC_REQUEST_URI_TOO_LONG	Request URL Too Large	请求 URL 过长
415	SC_UNSUPPORTED_MEDIA_TYPE	Unsupported Media Type	不支持的媒体类型
500	SC_INTERNAL_SERVER_ERROR	Server Error	服务器错误
501	SC_NOT_IMPLEMENTED	Not Implemented	未执行
502	SC_BAD_GATEWAY	Bad Gateway	网关坏
503	SC_SERVICE_UNAVAILABLE	Out of Resources	超出资源
504	SC_GATEWAY_TIME_OUT	Gateway Time Out	网关超时
505	SC_HTTP_VERSION_NOT_SUPPORTED	HTTP Version Not Supported	不支持 HTTP 版本

- 100～199 信息性的标识用户应该采取的其他动作。
- 200～299 表示请求成功。
- 300～399 用于那些已经移走的文件，常常包括 Location 报头，指出新的地址。
- 400～499 表明客户引发的错误。
- 500～599 指出由服务器引发的错误。

在 JSP Web 应用程序开发过程中常遇到的错误是 404 和 500。404 表示无法找到客户端所给地址的任何资源，即没有所请求的页面。这是最简单、最容易解决的错误，主要原因是请求地址错误，此时应仔细核对输入的路径和文件名是否正确，是否存在这样的路径和文件；另一个原因是虚拟路径不起作用，即 Web 应用程序未加载，此时应仔细核对虚拟路径配置标记是否正确，Web 应用程序的配置文件 web.xml 的格式是否正确，也可从服务器控制台显示的信息或日志文件中记录的信息查找原因。500 表示程序逻辑有错误，此时应根据错误提示调试程序代码。有时浏览器中显示不能加载类的错误提示，这是由于页面有错误，服务器在将 JSP 转化为 Servlet 类或进行编译时出现错误，所以服务器无法加载这个类，这种错误信息提示不明确，此时应将浏览器刷新几次，直到出现页面错误信息提示为止。

1.1.5　JSP 动态网页的处理过程

JSP Web 服务器在处理 JSP 网页时，首先由 JSP 引擎将 JSP 文件转化为 Servlet（一种 Java 类），其次将该 Servlet 编译为.class 文件，然后调用 Servlet 引擎执行 class 文件，输出 HTML 网页发送到客户端的浏览器中显示。其处理过程如图 1-3 所示。

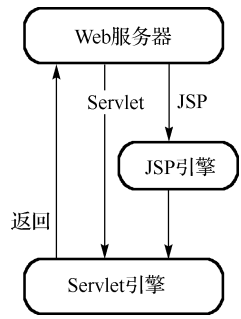

图 1-3　JSP 页面的处理过程

JSP 动态网页需要能够解析处理 JSP 代码的 Web 服务器支持，支持 JSP 的 Web 服务器有：Apache Tomcat、SUN JSWDK（Java Server Web Development Kit）、caucho Resin、W3C Jigsaw、IBM Websphear、BEA Weblogic、Jboss.org Jboss、Allaire Jrun 等。前四个是轻量级的 JSP 服务器，并且是开源软件。后四个是全功能的 Java EE 服务器，除 Jboss 外都是商业软件，其中 Tomcat 是最常用的 JSP Web Server，Jboss 也是通过内置 Tomcat 支持 JSP 的。本书中的 JSP Web 服务器使用 Tomcat。

1.2　Tomcat 服务器的安装与配置

Tomcat 是一个开放源代码的 Servlet 容器，它是 Apache 软件基金会（Apache Software Foundation）的一个顶级项目，由 Apache、SUN 和其他一些公司及个人共同开发而成。由于有了 SUN 的参与和支持，最新的 Servlet 和 JSP 规范总能在 Tomcat 中得到体现，Tomcat 的版本比较多，各版本的详细信息见附录 A。因为 Tomcat 技术先进、性能稳定，而且免费，因此深受 Java 爱好者的喜爱，并得到了部分软件开发商的认可，成为目前比较流行的 Java Web 服务器。Tomcat 运行依赖于 JSDK（Java SE Development Kit），安装 Tomcat 之前需下载安装 JSDK。

1.2.1　安装 Java SE

Java SE 可从 http://www.oracle.com/technetwork/java/javase/downloads/中下载，当前的最新版本是 JSDK 9。不建议下载安装最新的 JSDK 版本，因为 Tomcat、Eclipse 等程序不可能立刻支持新版本的 JSDK，一般选择 Tomcat 和 Eclipse 主流版本支持的 JSDK，通常是次新版本的 JSDK。

除了版本的选择，还应根据操作系统及其字长选择 JSDK 为 Windows、Linux 或者 Mac 发布包，以及是 32 位还是 64 位。32 位的操作系统只能安装 32 位的 JSDK，64 位的操作系统可以安装 64 位的 JSDK，也可以安装 32 位的 JSDK。JSDK 下载选择页面如图 1-4 所示。

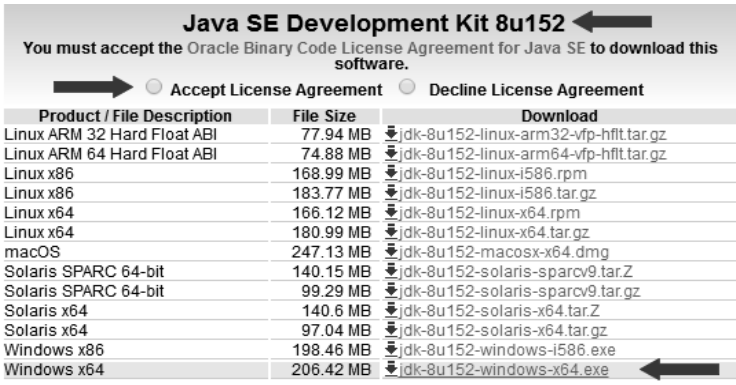

图 1-4　JSDK 下载选择页面

对 Windows 系统，双击下载的 JSDK 安装包，运行安装程序，逐步安装即可。安装中会有两个程序的安装过程，第一个程序是 JSDK，第二个程序是 JRE，注意不要取消对 JRE 的安装，两个程序的安装路径设置最好一致。

安装完成后，添加环境变量：我的电脑→属性→高级→环境变量→系统变量中添加以下环境变量：

```
JAVA_HOME=C:\Program Files\Java\jdk1.6.0
CLASSPATH=.;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar;
```

在 Path 环境变量中添加%JAVA_HOME%\bin；即 Path=%JAVA_HOME%\bin;原有内容。

用记事本(Notpad)写一个简单的 Java 程序来测试 JSDK 是否已安装成功：

```
public class JreTest {
    public static void main(String args[]) {
        System.out.println("Hello! The Java running enviorenment is OK! ");
    }
}
```

将程序保存为文件名是 JreTest.java 的文件。

打开命令提示符窗口(开始→运行 cmd 命令)，进入 JreTest.java 所在目录，输入下面的命令：

```
javac JreTest.java
java JreTest
```

此时若在命令窗口中显示“Hello! The Java running enviorenment is OK!”，则 JSDK 安装成功，环境变量设置正确。若没有显示出该字符串，请仔细检查以上配置是否正确。也可在命令提示符窗口中输入 javac 以及 java 命令直接运行 Java 编译器和虚拟机，如果显示这两个程序的帮助说明，则 JSDK 安装成功，环境变量设置正确。Java 编译器与虚拟机易受病毒感染，必须保证计算机无病毒才能正常运行。

注意，要显示记事本程序的扩展名，必须将文件夹选项中“隐藏已知文件类型的扩展名”(我的电脑→工具→文件夹选项→查看→隐藏已知文件类型的扩展名)不勾选。

1.2.2　安装 Tomcat

Tomcat 可从 http://tomcat.apache.org 中下载，目前的版本为 9.0，必须根据操作系统类型和所安装的 JSDK 来选择 Tomcat 的发布类型、版本和字长。下载页面中有个 Readme 链接，单击打开 Readme 文件，其中有一行红色文本指出该版本 Tomcat 所要求的 JSDK。Tomcat 各版本的详细信息见附录 A。注意，32 位的 JSDK 只能运行 32 位的 Tomcat，64 位的 JSDK 只能运行 64 位的 Tomcat。Tomcat 的发布文件还有安装版和非安装版两种，其文件扩展名分别是.exe 和.zip。推荐下载无须安装的绿色版。Tomcat 下载选择页面如图 1-5 所示。

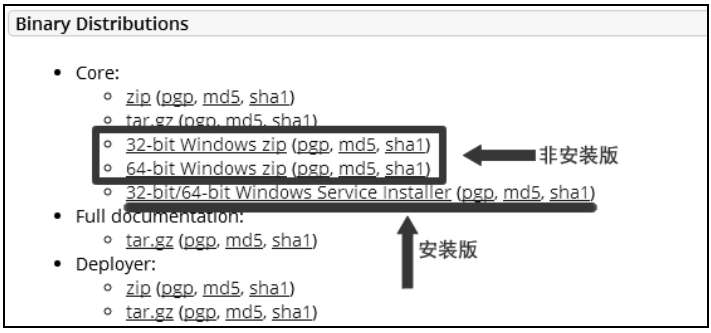

图 1-5　Tomcat 下载选择页面

如果下载安装版，运行下载的.exe 文件，按向导提示单击下一步按钮进行安装，安装后可以从开始菜单启动 Tomcat 的服务管理器，此时服务管理器的图标将显示在任务栏右侧的通知区域，右击图标通过其右键菜单来启动、关闭 Tomcat 服务器。也可以运行 Tomcat 的安装目录 bin 子目录下的 tomcat6.exe 或 tomcat6w.exe 启动 Tomcat 服务器。

非安装版解压后即可使用，通过运行 Tomcat 安装目录下的批处理可执行文件\bin\startup.bat 启动服务器，Tomcat 的启动与关闭如图 1-6 所示。启动后会打开一个命令提示符窗口，这是 Tomcat 服务器的控制台界面，其上显示了服务器的状态信息，如图 1-7 所示。不要直接关闭该窗口。如果不小心关闭，必须运行一次 Tomcat 安装目录\bin\shutdown.bat 才能再次启动。该文件是关闭 Tomcat 服务器的批处理程序。安装版 Tomcat 的控制台已重定向到日志文件 stdout.log，服务器输出的状态信息可从该文件中读取。

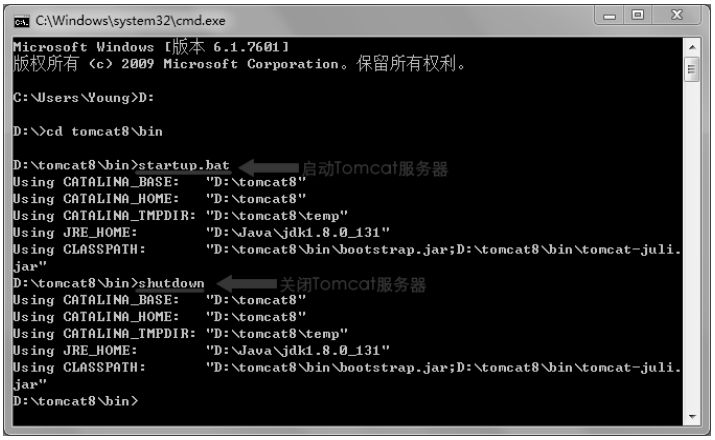

图 1-6　Tomcat 的启动与关闭

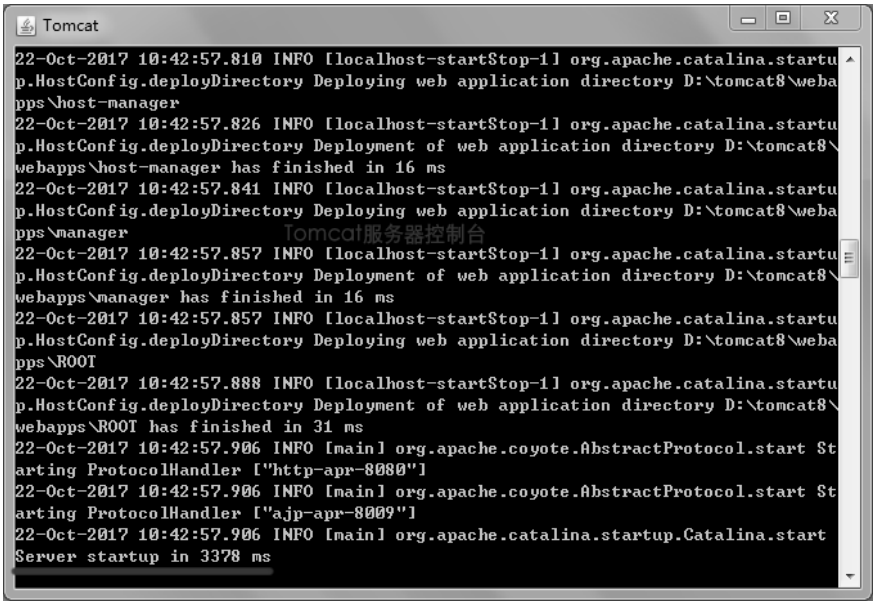

图 1-7　Tomcat 控制台界面

打开浏览器，在地址栏中输入 http://localhost:8080，如果显示 Tomcat 的欢迎界面(见图 1-8)，则说明服务器安装成功，并已正常启动。如果 Tomcat 服务器不能启动，可以先打开命令提示符窗口，切换到 Tomcat 安装目录下的 bin 子目录，运行 startup.bat，查看服务器控制台窗口输出的错误信息，安装版查看日志文件中记录的错误信息，根据服务器显示的错误信息查找原因，有时是 8080 端口已被占用，更多的情况是计算机感染病毒所致。

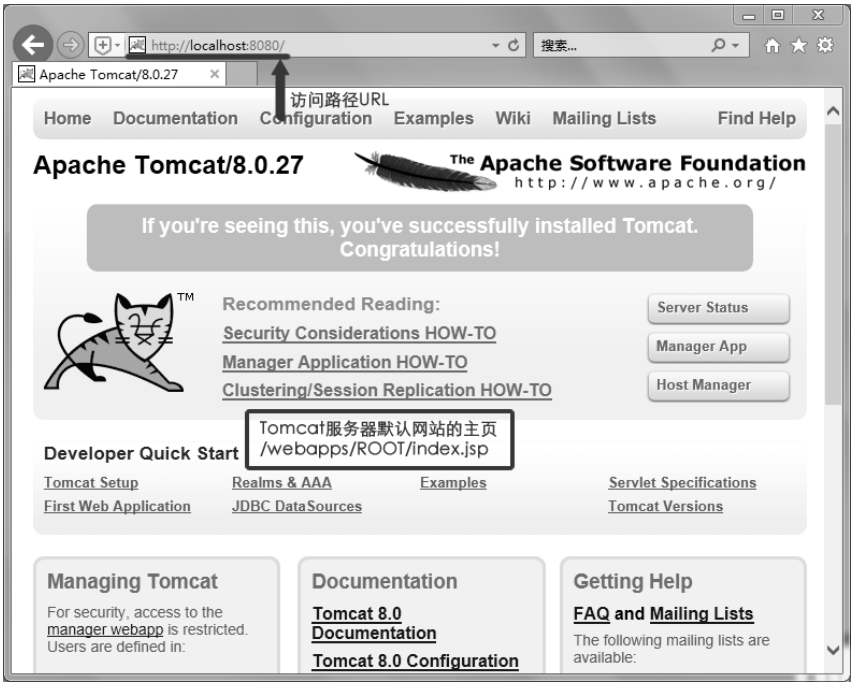

图 1-8　Tomcat 主页界面

1.2.3　Tomcat 服务器的目录结构

Tomcat 的安装目录结构及各目录下文件的功能如表 1-2 所示。

表 1-2　Tomcat 的安装目录结构及各目录下文件的功能

目　录	描　述
bin	包含启动、关闭 Tomcat 或者其他功能的脚本(.bat 文件和.sh 文件)
conf	包含各种配置文件，包括 Tomcat 的主要配置文件 server.xml 和为不同的 Web 应用设置默认值的文件 web.xml
lib	包含 Tomcat 中使用的 jar 文件。在 UNIX 平台中，此目录下的任何文件都被加到 Tomcat 的 classpath 中
logs	日志文件目录
temp	临时文件
webapps	Web 应用程序主目录
work	Tomcat 自动生成，放置 Tomcat 运行时的临时文件，有 JSP 编译出的 Servlet 的.java 和.class 文件

1.2.4　Tomcat 服务器的配置文件

Tomcat 服务器的配置文件是 conf 目录下的 server.xml，该文件的总体结构如下。

```
<!--根元素，代表整个服务器。-->
<Server>    <!--服务器是服务的生存环境；关闭端口 8005，关闭指令 SHUTDOWN-->
    <!--服务器可包含多个服务，默认服务的名称为 Catalina-->
    <Service>    <!--服务，由 Engine 及相关的一组 Connectors 构成-->
        <!--每个服务由多个 Connector 和一个 Engine 容器组成-->
        <Connector/>    <!--客户端与服务器之间的连接， port 属性定义端口-->
        <Engine> <!--引擎，处理 Service 的所有请求，实现 Container 接口-->
            <!--每个 Engine 容器可包含多个 Host-->
            <Host>    <!--虚拟主机  appBase 属性设置应用程序目录-->
                <!--每个 Host 可包含多个 Context-->
                <Context/>  <!--虚拟目录，每个 Context 是一个独立的网站-->
             </Host>
        </Engine>
    </Service>
</Server>
```

配置文件各元素详解如表 1-3 所示。

表 1-3　配置文件各元素详解

元　素　名	属　性	解　释
server	port	指定一个端口，这个端口负责监听关闭 Tomcat 的请求
	shutdown	指定向端口发送的关闭命令字符串
service	name	指定 service 的名字
Connector(表示客户端和 service 之间的连接)	port	指定服务器端要创建的端口号，并在这个端口监听来自客户端的请求
	minProcessors	服务器启动时创建的处理请求的线程数
	maxProcessors	最大可以创建的处理请求的线程数
	enableLookups	如果为 true，则可以通过调用 request.getRemoteHost()进行 DNS 查询来得到远程客户端的实际主机名，若为 false 则不进行 DNS 查询，而是返回其 IP 地址
	redirectPort	指定服务器正在处理 HTTP 请求时收到了一个 SSL 传输请求后重定向的端口号
	acceptCount	指定当所有可以使用的处理请求的线程数都被使用时，可以放到处理队列中的请求数，超过这个数的请求将不予处理
	connectionTimeout	指定超时的时间数(以毫秒为单位)

续表

元 素 名	属 性	解 释
Engine(指定 service 中的请求处理机，接收和处理来自 Connector 的请求)	defaultHost	指定默认的处理请求的主机名，它至少与其中的一个 host 元素的 name 属性值是一样的
Context(表示一个 Web 应用程序，通常为 WAR 文件)	docBase	应用程序的路径或者是 WAR 文件存放的路径
	path	表示此 Web 应用程序的 URL 的前缀，这样请求的 URL 为 http://localhost:8080/path/****
	reloadable	如果为 true，则 Tomcat 会自动检测应用程序的/WEB-INF/lib 和/WEB-INF/classes 目录的变化，自动装载新的应用程序，我们可以在不重启 Tomcat 的情况下使应用程序的变化生效，一般在调试阶段设为 true，正式发布后改为 false
host(表示一个虚拟主机)	name	指定主机名
	appBase	应用程序基本目录，即存放应用程序的目录
	unpackWARs	如果为 true，则 Tomcat 会自动将 WAR 文件解压，否则不解压，直接从 WAR 文件中运行应用程序
Logger(表示日志，调试和错误信息)	className	指定 logger 使用的类名，此类必须实现 org.apache.catalina.Logger 接口
	prefix	指定 log 文件的前缀
	suffix	指定 log 文件的后缀
	timestamp	如果为 true，则 log 文件名中要加入时间，如 localhost_log.2001-10-04.txt
Realm(表示存放用户名,密码及 role 的数据库)	className	指定 Realm 使用的类名，此类必须实现 org.apache.catalina.Realm 接口
Valve(功能与 Logger 差不多，其 prefix 和 suffix 属性解释和 Logger 中的一样)	className	指定 Valve 使用的类名，如用 org.apache.catalina.valves.AccessLogValve 类可以记录应用程序的访问信息
	directory	指定 log 文件存放的位置
	pattern	有两个值，common 方式记录远程主机名或 IP 地址、用户名、日期、第一行请求的字符串、HTTP 响应代码，发送的字节数。combined 方式比 common 方式记录的值更多

配置文件 server.xml 中的默认设置一般无须修改，其服务器的端口为 8080，主目录为 $TOMCAT_HOME/webapps/Root。通过修改属性 protocol="HTTP/1.1"的 Connector 元素的 port 属性值可以修改 Tomcat 服务器的端口号。

```
<Connector port="8080" protocol="HTTP/1.1" connectionTimeout="20000"
        redirectPort="8443" />
```

通过修改 Host 元素的 appBase 属性值可以更改 Tomcat 服务器默认的应用程序发布目录，该目录下的 Root 目录为默认的主目录。

```
<Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy=
            "true" xmlValidation= "false" xmlNamespaceAware="false">
```

在 server.xml 中还可以配置虚拟主机、用户验证方式、设置单点登录等功能。实际上最常见的操作是建立虚拟路径或直接部署 Web 应用程序，1.3 节将具体介绍。注意，在修改默认配置前应对 server.xml 文件进行备份，任何错误设置将导致服务器无法启动。

在 config 目录下还有一个配置文件 web.xml，用来设置服务器上所有网站(Web 应用程序)的默认值(通用配置项)。文件 tomcat-user.xml 用来设置远程登录 Tomcat 服务器的角色和用户名，admin 角色的用户可以使用 Admin Web Application 进行系统管理，manager 角色的用户可以使用 Manager Web Application 进行应用管理。在 Tomcat 的默认主页(欢迎界面)有这两个 Web 管理程序的登录入口，还有帮助文档、Servlet 和 JSP 例程等学习资源的链接入口。

1.3　JSP Web 应用程序的目录结构与发布

浏览 Tomcat 默认主目录$TOMCAT_HOME/webapps/Root 的结构，可以了解到 JSP Web 应用程序的目录结构：

Web 应用程序的根目录

\WEB-INF\web.xml Web 应用程序的配置文件

\WEB-INF\classes\　放置包目录结构及 Java 类

\WEB-INF\lib\　放置打包后的.jar 文件

\JSP　文件和其他资源

WEB-INF 子目录是 Web 应用程序的系统目录，该目录下的资源文件只能由服务器读取或调用，不允许从客户端来请求。注意其目录名必须大写，中间是连字符而不是下画线，如果目录名错误，该 Web 应用程序的配置文件将不起作用，classes 目录下的 Java 类和 lib 目录下的 jar 文件将不能调用。配置文件 web.xml 是 XML 格式的，其结构如下。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
    <display-name>JSP Web Application Examples</display-name>
    <description>This Web site is the examples for study JSP</description>
    <!-- Web 应用程序参数设置 -->
    <context-param>
        <param-name>adminEmail</param-name>
        <param-value>admin@servername.com</param-value>
    </context-param>
    <!-- 过滤器声明 -->
    <filter>
        <filter-name>filtername</filter-name>
        <filter-class>packages.filterclass</filter-class>
        <!-- 过滤器初始化参数设置 -->
        <init-param>
            <param-name>param1</param-name>
            <param-value>value1</param-value>
        <init-param>
    </filter>
    <!-- 过滤器映射 -->
    <filter-mapping>
        <filter-name>filtername</filter-name>
        <url-pattern>/*</ url-pattern >
    </filter-mapping>
    <!-- 监听器配置 -->
    <listener>
        <listerner-class>packages.listernerclass</listerner>
    </listener>
```

```
<!-- Servlet 声明 -->
<servlet>
    <servlet-name>servletname</servlet-name>
    <servlet-class>packages.servletclass</servlet-class>
    <!-- Servlet 初始化参数设置 -->
    <init-param>
        <param-name>param1</param-name>
        <param-value>value1</param-value>
    <init-param>
</servlet>
<!-- Servlet 映射 -->
<servlet-mapping>
    <servlet-name>servletname</servlet-name>
    <url-pattern>/otherpath/requestname</ url-pattern >
</servlet-mapping>
<!-- 会话配置 -->
<session-config>
    <session-timeout>30</session-timeout>
</session-config>
<!-- MIME 文件类型配置 -->
<mime-mapping>
    <extension>gif</extension>
    <mime-type>image/gif</mime-type>
</mime-mapping>
<!-- 默认文档设置 -->
<welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
<!-- 错误页面配置 -->
<error-page>
    <exception-type>java.lang.SqlException</exception>
    <location>dberror.html</location>
</error-page>
<error-page>
    <error-code>404</error-code >
    <location>404.html</location>
</error-page>
<!-- JSP 配置元素 -->
<jsp-config>
    <taglib>
       <taglib-uri></taglib-uri>
       <taglib-location></taglib-location>
    </taglib>
    <jsp-property-group>
       <url-pattern>*.jsp</url-pattern>
```

```
            <el-ignored>false</el-ignored> <!-- 控制表达式语言的启用 -->
            <scripting-invalid>false</scripting-invalid><!--控制脚本元素的启用 -->
            <include-prelude>/header.jsp</include-prelude> <!-- 自动包含序言 -->
            <include-coda>/footer.jsp</include-coda> <!-- 自动包含页尾 -->
            <page-encoding>iso-8859-1</page-encoding> <!-- 页面编码设置 -->
        </jsp-property-group>
    </jsp-config>
    <!-- 安全控制配置 -->
    <security-constraint>
        <web-resource-collection>
            <web-resource-name>resourcename</web-resource>
            <url-pattern>/admin/*</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <role-name>admin</role-name>
        </auth-constraint>
    </security-constraint>
    <!-- 验证机制配置 -->
    <login-config>
        <auth-method>BASIC</auth-method>
    </login-config>
    <!-- 安全角色配置 -->
    <security-role>
        <role-name>admin</role-name>
    </security-role>
    <!-- 其他配置项 -->
</web-app>
```

该配置文件必须正确设置，如果有错误将导致整个 Web 应用程序无法加载。根元素为 web-app，它的各个属性指定了该 XML 文件的名称空间，其中的 web-app_2_5.xsd 是配置文件的架构(Schema)，它实际上决定了该 Web 应用程序所适用的 Servlet/JSP 版本。

display-name 元素用来为整个 Web 应用程序指定一个简略名字，以便使 GUI 工具表示 Web 应用程序的名字。description 元素为 Web 应用程序编制一个简短的文本描述，它是整个 Web 应用程序的一个简化形式的文档。context-param 元素为整个 Web 应用程序定义初始化参数。mime-mapping 元素设置 Web 应用程序中使用的 MIME 文件类型，这些类型一般在 Tomcat 安装目录 config 子目录下的全局 web.xml 配置文件中设置，这两个配置文件中的设置不能冲突。welcom-file 元素设置了应用程序的默认主页。过滤器的声明及映射在过滤器一章中介绍；监听器配置在监听器一章中介绍；Servlet 的声明及映射在 Servlet 一章中介绍；会话设置参见内置对象一章中的 session；错误页面配置、脚本元素的启用、页面编码等的使用实例见 JSP 基本语法一章；taglib 设置在自定义标签一章中介绍；表达式语言的启用参见表达式语言一章；自动包含序言的设置实例参见 JDBC 一章的 JSP 数据库开发实例；安全选项在安全性一章中介绍。可见 web.xml 配置文件中的配置项贯穿于整个 JSP 知识体系。

将 Web 应用程序发布到 Tomcat 服务器，有以下三种方法。

(1)将 Web 应用程序目录复制到 Tomcat 的应用程序目录 webapps 下。

(2)创建 Web 应用程序存档文件(WAR)，将.war 文件复制到应用程序目录 webapps 下。

(3)建立虚拟路径。

编写 Context 标记的 XML 片段:

```
<Context path="/虚拟目录名"  docBase="Web 应用程序目录或.war 文件的物理路径"
        reloadable="true" debug="0">
</Context>
```

属性 docBase 指定应用程序的路径或者 war 文件的存放路径,path 表示此 Web 程序的虚拟目录名(URL 前缀),reloadable 如果为 true,Tomcat 会自动检测应用程序/WEB-INF/lib 和/WEB-INF/classes 目录的变化,自动装载新的应用程序,改变 Java 类或.jar 文件,无须重新启动 Tomcat 即可使更改生效。

将这段 XML 片段放在服务器配置文件 server.xml 的 Host 标记中,(结束标记</Host>前即可)。或者将其保存在单独的一个 XML 文件中,放置在$TOMCAT_HOME/conf/Catalina /localhost /目录下,在 Tomcat 6.0 中需新建/Catalina /localhost /子目录,其中文件名(除.xml 扩展名部分)为虚拟目录名,此时 path 属性将不起作用。

注意,JSP 中涉及的地址路径(目录、文件名等)不要使用中文,设置虚拟目录后必须重启 Tomcat 才能生效。

虚拟目录的设置和测试例程。

用记事本打开 Tomcat 的配置文件 conf\server.xml,在</Host></Engine>之前添加:

```
<Context path="/jspex" docBase="g:/jsp/jspex" reloadable="true" debug="0">
        </Context>
```

其中 jspex 是虚拟路径名,g:/jsp/jspex 是对应的物理路径。

或者在$TOMCAT_HOME\conf\Catalina\localhost 目录下创建文件 jspex.xml,文件内容为:

```
<Context path="/jspex" docBase="g:/jsp/jspex" reloadable="true" debug="0">
        </Context>
```

用记事本编写一个简单的 JSP 程序。

例程 1-1,jspTest.jsp

```
<html>
<head>
<title>JSP Running Environment Test</title>
</head>
<body>
<h1>JSP Running Environment Test</h1>
<%
   out.print("Hello! JSP running environment is OK!");
%>
</body>
</html>
```

保存到虚拟路径对应位置 g:/jsp/jspex 的 begin 子目录,文件名为 jspTest.jsp。启动 Tomcat,启动浏览器(IE),输入 http://localhost:8080/jspex/begin/jspTest.jsp,向本地 Web 服务器 Tomcat 请求 jspTest.jsp 网页,测试虚拟路径的设置是否成功。

本 章 小 结

本章从 Web 发展过程出发，介绍了动态网页技术涉及的 URL 查询字串、HTTP 状态码、JSP 动态网页的处理过程。然后引出支持 Servlet 及 JSP 的 Web 服务器——Tomcat，介绍了 Tomcat 服务器的安装，Tomcat 服务器的目录结构，Tomcat 服务器的配置文件。最后介绍了 JSP Web 应用程序的目录结构，JSP Web 应用程序的配置文件和主要的配置项，以及将程序发布到 Tomcat 服务器上的三种方法。

思　考　题

1．常用的代码混合动态网页编程技术有哪几种？
2．通过 URL 向服务器传递数据的查询参数用什么字符标识其开始？简述其输入格式。
3．服务器返回 404 和 500 错误，分别表示什么意思？
4．简述 JSP 动态网页的处理过程。
5．Tomcat 服务器配置文件的路径和名称是什么？
6．Tomcat 服务器的主目录是什么？
7．Tomcat 服务器中设置所有网站通用配置项的配置文件的路径和名称是什么？
8．JSP Web 应用程序配置文件的路径和名称是什么？
9．将 JSP Web 应用程序发布到 Tomcat 服务器有哪几种方式？
10．如何在 Tomcat 服务器中配置虚拟目录？