第 1 章　JSP入门

JSP 是 Java Server Pages 的首字母缩写，是由 Sun Microsystems 公司倡导、许多公司一起参与建立的一种动态网页技术标准，用于辅助对 Web 请求的处理。JSP 是建立在 Java Servlet 模型之上的表达层技术，其使用的是 Servlet 引擎。JSP 是由类似于 HTML 的标记、JSP 标记以及嵌入的 Java 代码片组成，允许将静态 HTML 内容与服务端脚本语言混合起来动态输出。第四章将重点介绍 JSP 中所使用到的 HTML 各标签、JavaScript 脚本语言以及 CSS（层叠样式表）技术。

在介绍 JSP 过程，本章并将 JSP 与 Servlet 以及 ASP、PHP 等类似技术进行比较，阐述它们各自所存在的优缺点。

本章要点包括以下内容：

- Web 介绍
- Web 客户端运行原理
- Web 服务器端运行原理
- Web 开发所采用技术的演变过程
- JSP 的介绍
- JSP 与 ASP 以及 PHP 的比较

1.1　Web介绍

以往基于客户、服务器的 C/S 结构应用程序存在很多缺点，它需要安装客户端程序。当应用程序升级时，客户端同样需要下载升级程序才能使用新的功能。这无形当中给客户端带来一定的麻烦，限制了该应用程序的广泛使用。当今更多的下载软件、及时通信软件等都是 C/S 结构的应用程序。

随着因特网技术的迅速发展，现在几乎所有的企业应用程序开发工作都要通过 Web 提供这些应用程序的服务。Web 应用程序的访问不再需要安装客户端程序，可以通过统一的浏览器（Microsoft 的 Internet Explorer 以及 Netscape 公司的 Navigator 浏览器）来访问各类 Web 应用程序。当 Web 应用程序进行升级时，并不需要在客户端做任何事。和 C/S 结构的应用程序相比，Web 应用程序可以在网络上更加广泛地进行传播和使用。

在了解如何开发 Web 应用程序之前，很有必要首先了解一下这些应用程序地运行平台和环境。下面小节就将重点介绍 Web 应用程序所涉及的两大运行平台：Web 浏览器和 Web 服务器。

1.2　Web客户端

Web 客户端是用来向 HTTP 服务器发送请求以及从 HTTP 服务器端接受响应信息的任何应用程序。当今最为常用的客户端程序是 Web 浏览器。Web 浏览器是将 GUI 请求变成 HTTP 请求，然后再将从服务器端接受的 HTTP 响应变成 GUI 显示内容。那么客户端是如何通过客户请求访问到服务器相应资源。客户首先通过浏览器输入访问服务器的 URL 地址，如图 1.1 所示。在浏览器的地址栏中输入

www.sina.com/news/shownews.jsp。

图 1.1　浏览器中的 URL 输入

Web 请求就是通过这里输入的 URL 地址来标识可以通过 Internet 访问的远程服务器中的资源。首先通过 www.sina.com 域名（对应一个惟一的 IP 地址）在网络中查找到对应的服务器主机，然后通过 news/shownews.jsp 指定访问 news 目录下的 shownews.jsp 页面程序。

Web 响应除了可以是 JSP 页面程序，还可以是其他类型的资源。但是 Web 响应通常是文档形式，最常见的是 Web 页面，包括多媒体和 HTML 表示内容，如文本、静态与动态图像、超链接、GUI 组件、声频片断和视频片断。

此外，HTTP Web 响应中还可以返回外部处理器管理的不同类型文档的引用、Java 小程序（如 Applet）和可执行浏览器脚本语言命令（如 JavaScript）。

1.2.1　浏览器访问Web服务器过程

了解一下使用一个浏览器去访问一个网站时，后台会发生什么的情况，网站是怎么实现请求和响应功能的。图 1.2 给读者展示了一个浏览器访问一个 Web 服务器的整个过程：

（1）用户首先在浏览器上（这里以 Microsoft Internet Explorer 浏览为例）输入网站的 URL 地址，这个地址告诉浏览器要和网络中的哪台主机进行联系。一般输入的是主机的域名（例如 www.sina.com.cn），域名惟一对应一个 IP 地址，一个 IP 地址又惟一识别一台联网的主机。

（2）浏览器寻找到指定的主机之后，向 Web 服务器发出请求。

（3）Web 服务器接受到请求并做出相应的分析，然后从存储器中获取一个采用 HTML 编码的 Web 页面。Web 服务器一般和 HTML 文件放置在同一个主机上。

（4）服务器把取出的 Web 页面返回给发出请求的浏览器作为响应。

（5）浏览器接收到相应的 Web 页面之后，在显示屏上向用户显示这个页面。

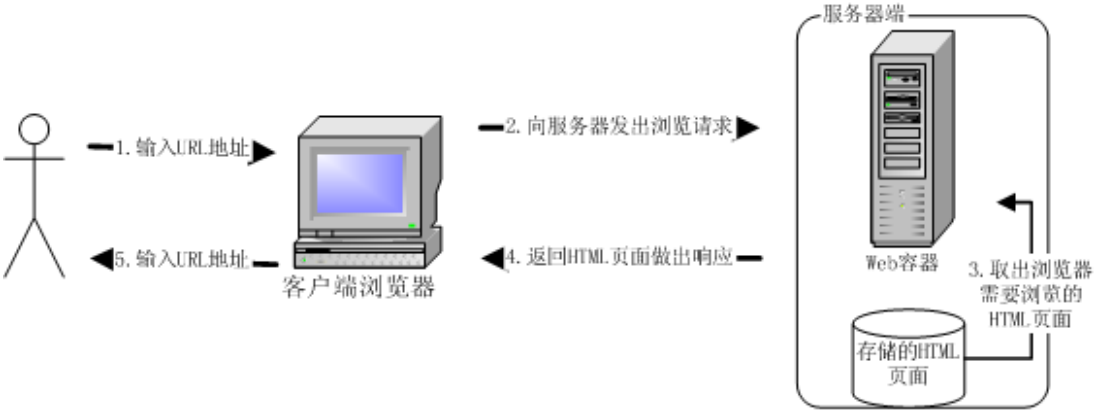

图 1.2　浏览器访问 Web 过程

1.2.2　HTTP超文本传输协议

浏览器与 Web 服务器之间是通过因特网进行会话的，而这种会话是通过一种叫着 HTTP（Hypertext

Transfer Protocol，超文本传输协议）的标准网络传输协议完成的。

HTTP 是一个请求/响应协议。在这个协议的基础上，Web 服务器才能和浏览器通过 Web 交换数据。HTTP 又是建立在 TCP/IP 协议基础之上的，而 TCP/IP 则是将因特网中所有计算机连接起来的一个协议组。

这里并不需要读者对 HTTP 协议以及 TCP/IP 知识有太多的了解，但要求读者对浏览器发出请求以及服务器如何发出响应的机制和原理有一定的了解，这对读者了解 Web 整体结构以及 JSP 有所帮助。

1.2.3　Web浏览器体系结构

经常使用浏览器上网浏览网页的读者应该对浏览器的运行机制有一定的初步了解。下面通过图 1.3 来对浏览器体系结构的加深理解。

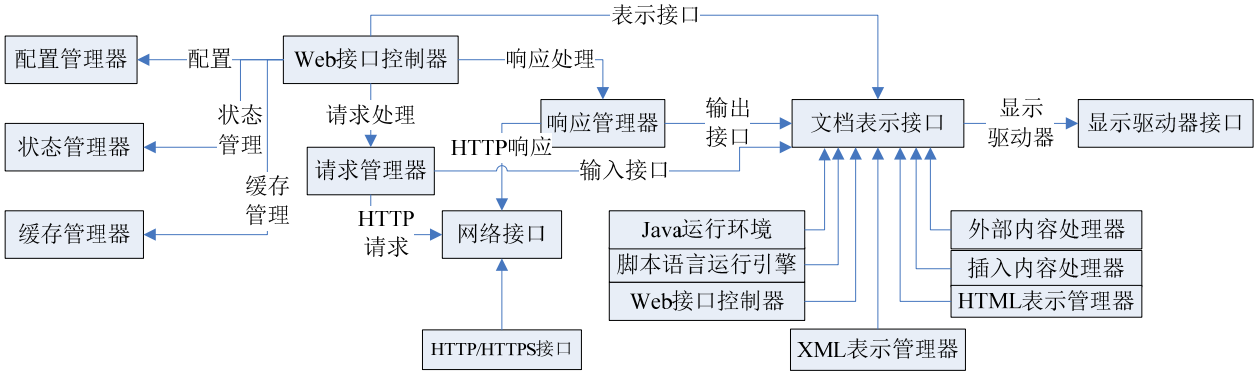

图 1.3　浏览器体系结构

由图 1.3 可以清晰地看出 Web 浏览器的核心是主控制器过程。负责管理本地内容的缓存和状态、配置 Web 浏览器信息以及驱动 Web 浏览器将要显示的页面内容，见图 1.2 中的“文档表示接口”功能。下面对各类管理器分别介绍：

（1）请求管理器将客户端的 GUI 请求（客户通过在浏览器地址中输入 URL 地址或者单击页面中的链接来发出请求）映射为 HTTP 网络请求，通过 HTTP 请求去访问网络中指定服务器主机中的资源。

（2）响应管理器将来之服务端的 HTTP 响应映射为 GUI 事件，并在浏览器中显示请求的内容。

（3）Web 浏览器通常使用缓存管理器来存储客户端请求数据，从而可以避免数据流 I/O 处于排队等待以及出现多余的网络请求。

（4）浏览器还可以通过状态管理器中的 cookies 来提供会话管理以及支持其他形式的会话跟踪。

（5）配置管理器可以允许客户根据需要对 Web 浏览器相应的属性和行为进行配置。

（6）文档表示接口驱动用于处理 Web 服务器的响应内容（HTML、JavaScript 或者包含 Applet 小程序等），使这些数据能够在浏览器端实际显示。Web 浏览器通常支持如下一种或者多种文档表示接口：

- HTML 表示管理器：基本上所有的浏览器都有一个 HTML 表示管理器，其可以通过服务器返回的 HTML 实体来输出 HTML 格式所显示的内容以及接受客户端的用户输入，例如表单提交。
- XML 表示管理器：现在很多浏览器也已经支持 XML 格式的文档分析和显示。
- Java 运行环境：Java 运行环境可以支持浏览器去执行类似 Applet 的 Java 小程序以及调用下载的 JavaBean 组件的服务。
- ActiveX 运行引擎：该运行引擎可以支持浏览器执行下载的 ActiveX/COM 组件。
- 脚本语言运行引擎：该引擎嵌入到浏览器中，可以执行类似于 JavaScript 的浏览器脚本语言。

- 外部内容处理器：嵌入外部内容处理器可以使 Web 浏览器在外部过程中运行的应用程序动态地执行所读取到的 URL 信息内容（例如，使用 Unzip 使用程序打开压缩档案文件）。该功能在平常比较少用，读者如果对此不太明白，并不影响后面的阅读。
- 插入内容处理器：可以在浏览器中选择性地加载插入内容处理器插件，这样可以使得浏览器能够直接执行读取的某些 URL 信息内容。这些插件是对浏览器的一种扩展，一般不是浏览器所特有的。

1.2.4 Web浏览器实现

现在 Web 浏览器的实现版本种类很多，但是比较普及和流行的为 Netscape Navigator（NN）和 Microsoft 公司的 Internet Explorer（IE）浏览器。这两个浏览器都能很好地支持最新最好的 HTML 表示标准，以及各种 HTML 扩展功能。另外，它们也都能支持 JavaScript 脚本语言以及类似 Applet 的 Java 小程序运行。

但是这两个浏览器还是存在区别的，NN 可以支持不同的多种平台，而 IE 则只能支持 Microsoft 平台。所以开发人员在开发 Web 应用程序时，需要考虑到这两种浏览器的显示支持。在本书中，所有的实例程序都只考虑 IE 浏览器。至于 NN 浏览器的显示标准以及与 IE 的详细区别，读者可以查看相关书籍和资料。

1.3 Web服务器

顾名思义，Web 服务器就是在服务器端执行的应用程序，其用来接受和处理客户端的 HTTP 请求以及生成与发送 HTTP 响应。Web 服务器的类型有很多种，例如：

- 从最初的只能接受 GET 或 POST 请求，只对静态页面的请求进行处理和响应的 Apache 服务器；
- 后来有处理 GUI、Servlet 以及 JSP 等动态效果的服务器，如 Tomcat、Resin 及大型商业服务器。

现在商用的 Weblogic 以及 Websphere 服务器，还可以很好支持可伸展 Web 客户机个数的并发请求，以及某种安全访问控制和支持各种可扩展功能的 API。

1.3.1 Web服务器体系结构

本书还是通过图来形象地展示 Web 服务器的基本体系结构，如图 1.4 所示。Web 服务器控制器通常管理一个线程池（多个初始化线程放在一个容器内进行统一管理），用于接受和处理客户端的各种请求。被分配的某个 Web 处理器线程专门管理一个特定的客户端请求和作出响应。

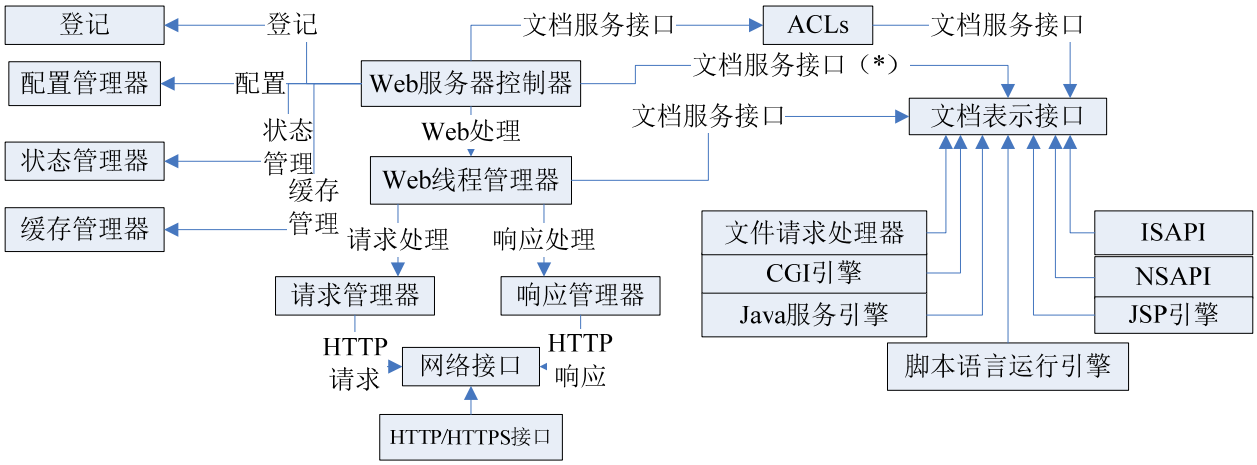

图 1.4　服务器体系结构

同样，Web 服务器也可以通过缓存管理器来维护服务器端的响应信息，这样可以使得进行同一请求的客户可以迅速生成缓存响应。通过状态管理器，Web 服务器可以维护客户机之间的会话管理信息，使得无状态 HTTP 协议实现状态。Web 服务器还可以通过配置管理器来对服务器环境进行适当的配置。在服务器中，可以指定特定 ACL（过滤器功能，可以是一个或者多个），来限制对服务器端某些资源或者某些客户的访问。除此之外，现在大多数服务器还提供了登记 Web 服务器请求和响应的机制。

- 文档表示接口是整个服务器体系结构中的中心，它根据 HTTP 请求信息生成相应的 Web 文档作为响应。Web 服务器可以支持如下的一种或者几种文档服务接口：
- 文件请求处理器：大多数 Web 服务器都提供文件请求处理器，它将客户端的 URL 地址映射成服务器本地将要访问的文件（例如 HTML 或者 JSP 等文件），然后进行读取处理并通过 HTTP 响应数据流返回给客户端。
- CGI 引擎：公共网关接口（CGI）引擎用于控制任何语言派生外部进程的标准接口，可以处理 HTTP 请求和生成 HTTP 响应。
- ISAPI：Internet 服务器应用程序接口（ISAPI）定义了调用 Miscrosoft 平台中的 DLL 的接口方法，也用于处理 HTTP 请求和生成 HTTP 响应。
- NSAPI：Netscape 服务器应用程序接口（NSAPI）定义了调用二进制库的接口，用于处理 HTTP 请求和生成 HTTP 响应。
- 脚本语言运行引擎：该引擎的嵌入可以使得浏览器能够处理 HTML 文件中存储的 JavaScript 和 VBScript 等脚本语言命令。这些脚本语言动态生成 HTML 页面页面，并发送回客户端。
- Java Servlet 引擎：该引擎在服务端解析和执行符合特定接口的 Java 代码程序（Servlet），进行处理 HTTP 请求和生成 HTTP 响应。
- JSP 引擎：Java 服务器页面（JSP）引擎首先将嵌入到页面中的特殊 Java 脚本语言编译成可执行的 Java Servlet 代码，然后在服务器进程中执行，处理 HTTP 请求和生成 HTTP 响应。

1.3.2　Web服务器实现

Web 服务器种类非常之多，其中可以有非常简单的，也可以有支持非常复杂的企业级特性（例如，安全和远程访问）。一般服务器都需要根据实际情况进行相应的配置，可以使用一个配置文件（一般为 XML 格式文件）进行环境参数的配置，有些服务器还提供了 GUI 界面的配置服务。

1.3.2.1 IIS 服务器

Microsoft Internet Information Server（IIS）在服务器端的 Microsoft 平台上作为 Web 服务的特定解决方案。它可以很好的和 Windows NT 以及 Windows2000 平台密切集成在一起。IIS 可以提供文件服务、CGI、ISAPI 以及 Active Server Pages（ASP）脚本环境之类的文档服务接口。

1.3.2.2 Apache Tomcat 服务器

Apache Tomcat 服务器是一个开源软件，可以很好的在 Windows 和 UNIX 平台上进行操作。在开发过程中 Tomcat 服务器的使用比较普遍，它没有一些商业服务器那么复杂。Tomcat 服务器不仅包括基本文件服务与 CGI 支持，还扩展成使用各种脚本语言和 Java Servlet。除了 Tomcat 为开源程序，Resin 也是一个开源服务器，由于这两个服务器都属于轻量级的，所以并不适合于企业级大型系统。

1.3.2.3 BEA Weblogic Server 服务器

BEA WebLogic Server 是一个企业级的 Web 服务器，主要用于 Java 平台。该服务器符合 J2EE 模型，并且提供 JSP 和 Java Servlet 的运行环境。WebLogic 服务器除了提供文件服务和 CGI 支持外，还支持基本 NSAPI 以及 ISAPI 文档服务接口。另外一个出色的企业级服务器是 IBM 公司发布的 Websphere 服务器。企业级服务器在安全和远程访问方面得到了很好的支持和扩展，只是在配置方面比较复杂，需要有专业人员参与。

本书实例使用的是 Tomcat 服务器，下面章节将具体介绍 Tomcat 服务器的安装和配置过程。

1.4 Web开发

前面详细介绍了 Web 客户端和服务器端的基本模型与体系结构，以及 Web 应用程序的请求和响应过程。在了解了 Web 应用程序的运作原理之后，下面将重点介绍 Web 应用程序的开发方法和过程，了解开发 Web 应用程序的各类技术，以及这些技术的发展过程和优缺点。

1.4.1 传统Web服务器模式开发

传统的 Web 应用开发仅仅能够提供有限的静态 Web 页面（HTML 静态页面），每个 Web 页面的显示内容是保持不变的。这样模式开发的 Web 应用很不利于系统的扩展，如果网站需要提供更多新的信息资料时，就只能修改以前的页面或者重新编写 HTML 页面并提供链接。而且 Web 网站的信息更新周期一般都比较长（因为需要重新编写代码）。

总结起来，传统 Web 应用开发模式存在如下多个不足：

- 不能提供及时信息，页面上提供的都是静态不变的信息。
- 当需要添加新的信息时，必须重新编写 HTML 文件。
- 由于 HTML 页面是静态的，所以并不能根据用户的需求提供不同的信息（包括不同的内容和格式），并不能满足多样性的需求。

静态页面的应用程序存在着这么多的缺点，决定这样模式必然不能适应大中型系统和商业需求。因此，很快因特网软件工程师转向了 CGI（Common Geteway Interface，公共网关接口），系统能够提供页面的动态生成。

1.4.2　动态构建页面的重要性

当发布全部为静态页面的 Web 应用程序（即传统 Web 服务器模式开发）时，随着企业业务的增多，HTML 页面程序会越来越多，非常不利于后期代码的维护，而且新信息发布过程非常麻烦。所以建立一个动态 Web 应用程序就显得非常重要。一方面可以根据访问者的不同请求返回不同的访问信息，即满足服务的多样性；另一方面，可以直接通过后台管理页面发布和修改信息即可，再也不需要修改页面程序或者添加更多页面程序。

动态 Web 应用程序的建立，可以给客户提供及时信息以及多样化服务，可以根据客户不同请求，动态地返回不同需求信息。下面将一一介绍创建动态页面的方法和技术。

1.4.3　CGI实现页面的动态生成

实现动态输出的 CGI 程序是运行在服务器端的，根据不同客户端请求输出相应的 HTML 页面，然后 Web 服务器再把这个静态页面返回给浏览器作为客户端的响应。具体的 CGI 操作流程如图 1.5 所示。

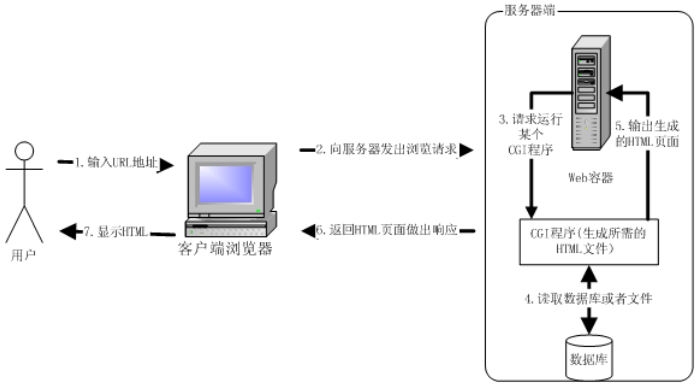

图 1.5　CGI 的操作过程

首先用户需要在浏览器地址栏输入 URL 地址或者单击链接来调用相应的 CGI 程序，例如 www.aaa.com/cgi/createhtml.cgi。通过 URL 地址，客户端取得和网络上域名为 www.aaa.com 的服务器主机连接。通过 Web 服务器调用执行 cgi 目录下的 createhtml.cgi 程序，然后将动态生成的 HTML 页面输出，最后由 Web 服务器通过网络将生成的 HTML 页面返回给客户端。

注意：CGI 程序在服务器端执行，并可以和 Web 服务器在同一个主机上。最流行的 CGI 语言是 Perl 和 shell 脚本，但是也可以使用 C、C++以及 Java 等语言进行编写。CGI 可以访问存储数据库中的数据或者其他系统中的文件，实现动态生成的效果。

虽然 CGI 实现了网站动态性，但是 CGI 也存在很多的不足之处：

- 需要为每个请求启动一个操作 CGI 程序的系统进程。如果请求非常频繁，这将会带来很大的开销。
- 需要为每个请求加载和运行一个 CGI 程序，这也将带来很大的开销。
- 需要重复编写处理网络协议的代码以及进行编码，这些工作都是非常耗时的。

前面已经介绍过了 Java 语言可以使用来编写 CGI 程序。但是遗憾的是，使用 Java 编写的 CGI 程序执行效率更加的低下。这是因为要执行一个 Java 编写的 CGI 程序，除了首先需要启动一个系统进程之外，还要在进程中启动一个 JVM（Java Virtual Machine，Java 虚拟机），然后才能在 JVM 中执行 Java CGI

程序（读者应该对 Java 程序的运行机制有所了解）。

为了解决 CGI 所留下来的问题，并产生了 Servlet。在下面这一小节将重点向读者介绍 Servlet 的基本原理。

1.4.5　Java Servlet：改进的CGI

由前面讨论可以知道，使用 Java 编写的 CGI 程序需要为每个请求都要启动一个系统进程以及 JVM，这大大降低了执行效率。如果能有办法取消这些开销，即只需要启动一个操作系统进程以及一个 JVM 映象，基于 Java 的 CGI 就能得到很好的改善。

Servlet 正是基于这样的想法才产生的。另外，可知 Java 可以在运行的时候动态地进行加载，所以可以利用这样的功能加载新的 Java 代码来处理新的请求。这样就可以只启动一次服务器进程，而且只需要加载一次 JVM，之后这个 JVM 再加载另外的类。基于这样的思想而出现的 servlet 执行效率就高得多了。

和传统的 CGI 程序相比，Servlet 有如下多个优点：

（1）只需要启动一个操作系统进程以及加载一个 JVM，大大降低了系统的开销。

（2）如果多个请求需要做同样处理的时候，这时只需要加载一个类，这也大大降低了开销。

（3）所有动态加载的类可以实现对网络协议以及请求解码的代码共享，大大降低工作量。

（4）Servlet 能够直接和 Web 服务器交互，而普通的 CGI 程序不能。Servlet 还能够在各个程序之间共享数据，使得数据库连接池之类的功能很容易实现。

Sun 公司在上世纪 90 年代末就发布了基于 servlet 的 Web 服务器。为了确保加载的各个类之间不起冲突，已经建立了一个称为 Java Servlet API（应用编程接口）的编码标准。现在基本上所有的服务器都遵循这个编码标准，所以 servlet 有很好的移植性。

现在的 Web 服务器（例如 Tomcat）已经集成了 servlet 容器，servlet 容器负责管理加载、卸载、重新加载和执行 servlet 代码等操作。

看下面一个比较典型的 servlet Java 源代码：

```
public class HelloWorldTest extends HttpServlet {
    public void doTest(HttpServletRequest request,HttpServletResponse reponse)
                        throws IOException,ServletException
    {
        String msg = "登陆成功";
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>JSP 2.0 Test</title>");
        out.println("</head>");
        out.println("<body>");
        out.println(msg);
        out.println("</body>");
        out.println("</html>");
     }
```

从这段代码可以看到 Servlet 重复使用了 PrintWriter.println()方法来输出相应的 HTML 页面，以达到动态生成 HTML 页面的效果。

虽然 Servlet 改变了传统 CGI 程序的缺点，但是它也有不足的地方：Servlet 可以建立动态生成的网

页，而网页中可以包含从服务器端的 Java 对象所获得的数据。但是 Servlet 生成网页的方法是在 Java 类中嵌入 HTML 标签和表达式。也就是说对 HTML 做一个小小的改动时，都需要修改和重新编译 Servlet 源文件，然后重新部署到 Servlet 容器当中。或许设计 HTML 页面和编写 Servlet 代码的人不是同一个人，这就使得修改 servlet 变得非常麻烦。

JSP 的引入就是要解决以上 Servlet 所存在的问题。下一小节将重点向读者介绍 JSP。

1.5　JSP介绍

Java 服务器页面（Java Server Pages，JSP）可以看着是 HTML 与 Java 之间的交积，即是实现普通静态 HTML 代码与 Java 混合编码的技术，它是 Servlet API 接口的一个扩展。JSP 语言提供了一个非常强大的工具，使得 Web 开发者可以更方便地进行企业 Web 支持工作。JSP 是在 HTML 中嵌入 Java 代码，调用执行时，会首先被编译成 Servlet，从而实现了页面的动态效果，但是 JSP 的编写要比 Servlet 方便的多。

1.5.1　JSP概述

Java 服务器页面（JSP）技术是指定在格式化文本（HTML）中嵌入特殊的脚本语言命令（现在只能是 Java 脚本语言），然后由 Web 服务器中的 JSP 引擎来编译和执行嵌入的脚本语言命令，然后再将整个生成的页面信息返回给客户端。和 Java Servlet 以及纯 HTML 页面一样，JSP 也是使用的 HTTP 作为默认的请求和响应机制。

JSP 语言对 Web 开发人员来说，是很有吸引力的。因为在 JSP 中直接使用的是 HTML 正确模板数据，它具有简单的 XML 式语法，然后可以使用<%　%>标记来嵌入 Java 代码，从而实现逻辑处理和动态输出。与 Servlet 的编写规则来说，JSP 的代码编写实现要简单容易多。惟一不足的，当今的 JSP 规范所能使用的脚本语言只能是 Java，今后的 JSP 版本应该可以支持除 Java 之外的其他嵌入语言。

下面演示一个最为简单的 JSP 文件代码，让读者首先有个整体的认识，代码如下：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD><TITLE>欢迎访问网上商店</TITLE></HEAD>
<BODY>
<H1>欢迎</H1>
<SMALL>欢迎,
<!--首次访问的用户名字为"New User"-->
<%out.println(Utils.getUserNameFromCookie(request));%>
要设置帐号信息，请单击
<A HREF="Account-Settings.html">这里</A></SMALL>
<P>
页面的其余内容。
</BODY></HTML>
```

程序说明：程序中除了<% %>之间的 Java 代码之外都是读者比较熟悉的 HTML 页面。这种 HTML 代码和脚本程序的很好结合使得动态网站的开发变得非常容易。如果读者对这段代码不是很理解，下面将慢慢地详细介绍。

接下来还是使用图的形式向读者展示 JSP 的整个访问和调用过程，如图 1.6 所示。

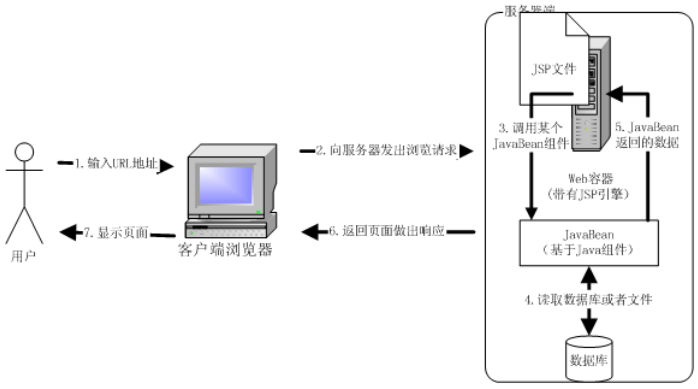

图 1.6　JSP 访问和调用过程示意图

由图可见，使用 JSP 开发的应用程序，Web 服务器的容器必须包含 JSP 引擎，它负责动态地对 JSP 文件进行检测和编译。

1.5.2　Web容器处理JSP文件的三个阶段

Web 容器处理 JSP 文件请求需要经过三个阶段：

- ❑ 被翻译阶段（Translation phase）：在这一阶段，编写好的 JSP 文件首先会被 Web 容器中的 JSP 引擎转换成 Java 源代码。
- ❑ 被编译阶段（Compilation phase）：JSP 文件所翻译成的 Java 源代码会被编译成可执行的字节码（可执行的字节码是二进制格式）。
- ❑ 请求阶段（Request phase）：当容器接受了客户端的请求之后，就执行前面已经编译成二进制字节码的 JSP 文件。处理完请求之后，容器再把生成的页面反馈给客户端进行显示。

下面通过图形象地显示出 Web 容器执行的这三个阶段，如图 1.7 所示。

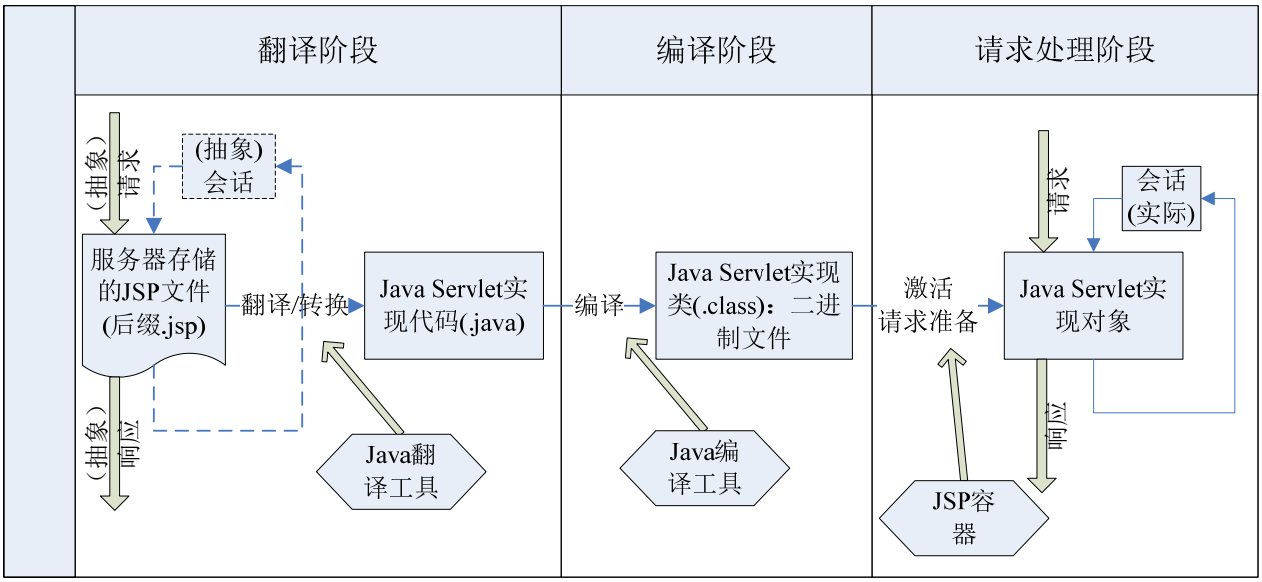

图 1.7　容器处理 JSP 的三阶段

注意：一旦容器把 JSP 文件翻译和编译之后，来自客户端的每一个 JSP 请求就可以重用这个编译好的二进制字节码，没有必要再把同一个 JSP 进行翻译和编译。这大大提高了 Web 应用系统的性能。如果对 JSP 进行了修改，容器就会及时地探测到这个修改，并进行重翻译和编辑。所以 JSP 文件在第一次请求时会比较慢，而之后同样 JSP 文件的请求会非常快。

1.5.3　JSP的优点

JSP 技术的设计目的是使得构造基于 Web 的应用程序更加容易和快捷，而这些应用程序能够与各种 Web 服务器、应用服务器、浏览器和开发工具能够很好地共同工作。JSP 网页可以非常容易的与静态模板结合，包括 HTML 或 XML 片段，以及生成动态内容的代码。它比以上讲的 Servlet 要更加优越。具体而言，JSP 有以下多个优点。

（1）JSP 提供一种模块机制，可以在 HTML 页面中嵌入基于 Java 的逻辑代码。

（2）使用 JSP 的时候，再也没有必要编写和编译用 Java 语言写的任何代码。而且对 JSP 进行修改会很快看到效果，这是因为 JSP 容器（或者称 JSP 引擎）会自动检测和重新编译 JSP。

（3）JSP 的使用大大缩短了服务器端基于 Java 的 CGI 的开发周期，实现了快速开发的目的。

（4）由于 JSP 标记是内嵌在 HTML 页面中的，所以完全可以先让 Web 页面设计者来设计页面模板。然后再让 Java 程序员处理所用的标记以及实现必要的逻辑功能，从而实现了图形和布局设计工作与应用开发任务的分离（后面介绍的 MVC 设计更能体现这一点）。

许多由 CGI 程序生成的页面大部分仍旧是静态 HTML，动态内容只在页面中有限的几个部分出现。但是包括 Servlet 在内的大多数 CGI 技术及其变种，总是通过程序生成整个页面。JSP 使得我们可以分别创建这两个部分。

1.5.4　JSP与ASP的比较

ASP 也是一个 Web 服务器端开发的技术，利用它可以产生和执行动态的、互动的以及高性能的 Web 应用程序。

1.5.4.1　相似点

ASP 是由 Microsoft 公司发布的，JSP 与 ASP 技术非常的相似。它们都提供了在 HTML 代码中嵌入某种程序代码，并由服务器提供的引擎来解释和执行这些程序。在 ASP 和 JSP 文件中，HTML 部分都是主要负责规定信息的显示样式，而嵌入的程序代码则用来实现逻辑和控制操作。

普通的 HTML 页面只依赖于 Web 服务器（例如 Apache、Microsoft IIS 以及 Sun Java System Web Server 等服务器），而 ASP 和 JSP 都需要有相应的语言引擎来分析和执行嵌入的程序代码。程序代码被引擎执行完之后会重新嵌入到 HTML 代码当中去，然后一起由服务器反馈给相应的浏览器作为相应。

ASP 和 JSP 都属于面向 Web 服务器的技术，它们都是在服务器端进行执行，而客户端浏览器不需要再安装任何附加的软件。

1.5.4.2　区别点

首先，ASP 使用的编程语言是 VBScript 之类的脚本语言，JSP 则是 Java，这是两者最明显的区别。

此外，它们之间还有一个更为本质的区别：两种语言引擎用完全不同的方式处理页面中嵌入的程序代码。在 ASP 下，VBScript 代码被 ASP 引擎解释并执行；在 JSP 下，代码被编译成 Servlet 并由 Java 虚拟机执行。这种编译操作仅在对 JSP 页面的第一次请求时发生，后期请求将直接调用编译好的文件。

1.5.4.3 JSP 的优势

JSP 和 ASP 相比具有两方面的优势：

- 动态部分用 Java 编写，而不是 VB Script 或其他 Microsoft 语言，功能更强大而且更易于使用。
- JSP 应用可以移植到其他操作系统和非 Microsoft 的 Web 服务器上。

1.5.5 JSP与PHP的比较

PHP 是一种跨平台的服务器端的嵌入式脚本语言。它融合了 C、Java 以及 Perl 语言的语法，并结合自身的特点，可以使得 Web 开发者快速地编写出动态的页面。PHP 还有一个优点就是它完全是免费的，可以从 http://www.php.net 网站上自由下载，并可以获取到源代码。

1.5.5.1 相似点

同样，普通的 HTML 页面只依赖于 Web 服务器，而 PHP 页面需要有附加的 PHP 语言引擎来分析和执行相应的程序代码。执行结果再重新嵌入到 HTML 代码当中去，一起由服务器通过 HTTP 协议反馈给相应的浏览器。另外 PHP 和 JSP 的可移植性非常好，基本可以执行于所有的平台。

1.5.5.2 区别点

JSP 在循环语句的执行以及数据库的访问速度上都要比 PHP（甚至 ASP）高的多。另外，由于 PHP 诞生于开源，它得到了迅速普及。但是当 JSP 出现后，这种情况就变化了。这是因为 PHP 只适合小型站点的开发和使用，而不适应大型的电子商务的站点开发。

这是由于 PHP 缺乏规模支持，并且还缺乏多层结构支持。对于大负荷的网站，只能使用分布式计算来解决问题。而 JSP 不同，它得到了 J2EE 框架支持，可以为 JSP 开发很多的 JavaBean，更复杂的系统可以得到 EJB 的支持。

1.5.5.3 JSP 的优势

JSP 很好地实现了多层结构，特别是得到 J2EE 的支持，现在在 MVC 模式当中，JSP 基本上只担任了 View 的工作，控制 View 的工作封装到了 Servlet 中。JSP 很适合开发大型的电子商务网站，而且在安全、可维护以及可扩展性上得到了很高的保障。

1.5.6 为什么选择JSP

或许读者看完了前一小节之后，就已经对这个问题有了一点答案。网站开发最重要的就是开发出来的网站要有很高的稳定性、安全性以及可扩展和维护性，而且使用的工具要尽量简单易用。

正是为了兼顾以上两点，JSP 提供了大量服务器端的标签（Tag），这样可以使 Web 开发者（特别是那些并不懂得 Java 编程的 HTML 设计者）不需要编写 Java 代码就可以完成很多动态内容的操作。而高级脚本编写者或者 Java 程序员也可以使用这些标签来实现动态效果。

除了 JSP 提供大量的标签之外，它还是依赖“组件为中心的网页开发”的技术。所谓“组件为中心”就是尽量把逻辑操作以及数据库操作等交付给 JavaBean 以及 EJB 等组件完成，这使代码的可重用性大大提高。

“组建为中心的网页开发”技术可以使某领域（像银行帐户计算）内的专家能够为本领域内的垂直应用编写专门组件（例如：JavaBean、EJB 组件，此类组件可作为产品出售），而 Web 开发者可以直接拿来使用而不必掌握这一领域的专门技术。这样就可以使得 Web 开发者能够把工作更多放在 Web 的构架和组件的使用上，而不再需要花大量时间在专业功能的实现上，这大大提高了 Web 开发的周期。

再者，JSP 的效率和安全性更高：JSP 文件在执行之前是先被编译成字节码（byte code），字节码由 Java 虚拟机(Java Virtual Machine)解释并执行，比源码解释的效率要高；服务器上还有字节码的 Cache 机制，能提高字节码的访问效率。第一次调用 JSP 网页可能稍慢，因为它被编译成 Cache，以后就快得多了。而且 JSP 源程序不大可能在页面被下载，这是因为 JavaBean 程序是完全放到不对外的目录中的。

另外 JSP 的适应平台更广，而且得到了 J2EE 构架的很好支持。

正是由于 JSP 存在这么多的优势，所以它得到了飞速的发展。这也是 Web 开发者选择 JSP 的原因。

1.6　本章小结

本章节首先向读者介绍 Web 相关知识，了解浏览器请求调用 Web 应用程序机制以及后台服务器的运行体系与原理。接下来讲解开发 Web 应用程序的各项技术，从最初的只是静态页面，到 CGI 实现网站的动态性，然后基于传统 CGI 生成了 Servlet，最后重点介绍 JSP 技术。

在介绍 JSP 时，并把 JSP 与流行的 ASP 以及 PHP 进行了比较，指出之间的相同点和不同点。并详细叙述了 JSP 存在的优势以及选择 JSP 用于开发网站的理由。学完本章节后，读者要了解到 Web 应用程序的请求与响应机制与原理、网站开发技术的发展过程以及 JSP 技术的特点、优势。