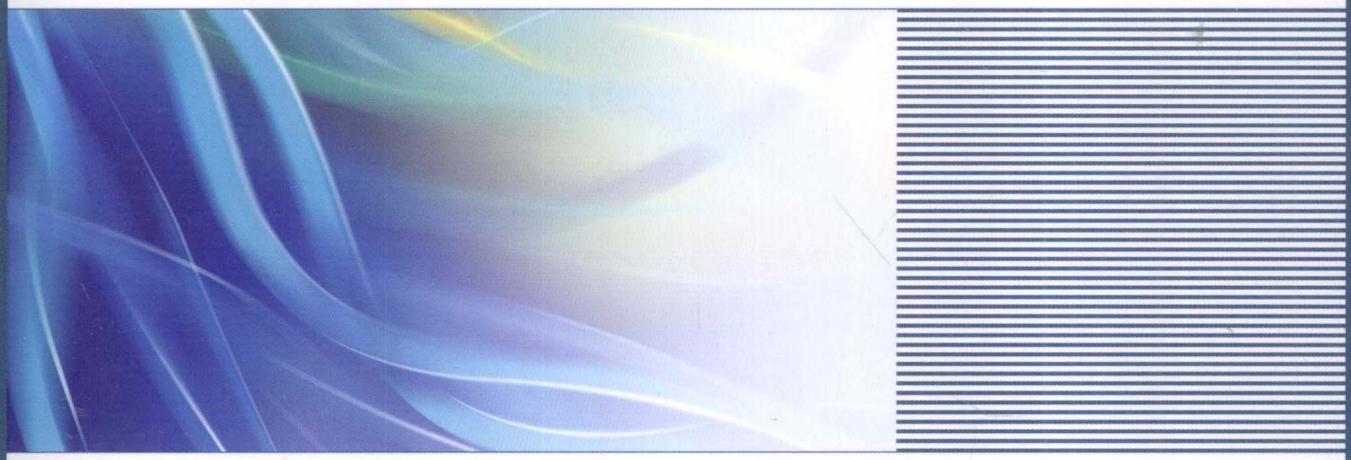
21世纪高等院校网络工程规划教材
21st Century University Planned Textbooks of Network Engineering



Java Web 应用开发技术

Java Web Programming

王电钢 刘孙俊 主编
刘念 丘建川 副主编

- 理论实践并重
- 讲解合理透彻
- 实例简短实用

人民邮电出版社
POSTS & TELECOM PRESS

21世纪高等院校网络工程规划教材
21st Century University Planned Textbooks of Network Engineering

Java Web 应用开发技术

Java Web Programming

王电钢 刘孙俊 主编
刘念 丘建川 副主编

人 民 邮 电 出 版 社
北 京

图书在版编目（CIP）数据

Java Web应用开发技术 / 王电钢，刘孙俊主编. --
北京 : 人民邮电出版社，2012.2
21世纪高等院校网络工程规划教材
ISBN 978-7-115-27129-7

Ⅰ. ①J… Ⅱ. ①王… ②刘… Ⅲ. ①
JAVA语言－程序设计－高等学校－教材 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2011)第278439号

内 容 提 要

本书系统地介绍了使用 Java 语言开发 Web 应用的基础技术。首先，从 Web 应用的基础——HTTP 入手；然后，逐章讲述了 Java 开发 Web 应用的各种技术和规范，涉及了 Java Web 容器、Servlet、JSP、过滤器、监听器、标记库、Web 应用安全、Web 开发模式；每章为读者配备了简明而又实用的示例。通过本书的学习，读者能够全面掌握 Java 语言 Web 编程技术，并对 Java Web 应用的基础知识有更深刻的了解。

本书可作为普通高等院校计算机及相关专业的教材，同时也可作为 Java 编程爱好者及开发人员的参考用书。

21 世纪高等院校网络工程规划教材

Java Web 应用开发技术

◆ 主　　编　王电钢　刘孙俊
副 主 编　刘　念　丘建川
责任编辑　刘　博

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
三河市海波印务有限公司印刷

◆ 开本：787×1092　1/16
印张：15.25　　2012 年 2 月第 1 版
字数：379 千字　　2012 年 2 月河北第 1 次印刷

ISBN 978-7-115-27129-7

定价：29.80 元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154

前　言

随着 Internet 的广泛使用以及相关技术的发展，网络已经像自来水和电力那样成为整个社会最为基础的公共服务，人们的生活也越来越离不开 Internet。而 Web 应用程序是 Internet 上使用最为广泛的应用程序，除了 Internet 外，Web 应用程序还被广泛地应用到各种企业应用领域中。在 Java EE 平台上使用 Java 开发 Web 应用程序已经成为一种重要的应用系统开发技术方案，从有着上亿用户的电子商务网站到银行的内部各种应用系统都有数量众多的成熟的 Java Web 应用程序案例，Java Web 应用程序几乎成为了 Web 应用程序的非官方标准。学习和掌握使用 Java 开发 Web 应用程序，对于在 Java EE 平台上从事应用系统开发的技术人员和希望以后能够从事软件开发的在校学生是非常重要的。一本组织合理，讲解透彻，能直接指导实践的教材能够帮助技术人员快速地掌握使用 Java 开发 Web 应用程序技术。基于以上目的，本书遵从“通过已知知未知”的思想来进行内容的组织。我们认为只有掌握原理，才能举一反三地加以应用，所以本书从介绍 Web 应用的基础——HTTP 开始，为读者打下坚实的基础。在具体内容的安排上力求做到理论与实践并重，既要讲清楚技术原理，也要能够指导读者进行实践。

本书内容

在具体内容组织上，本书从 Web 应用程序工作原理入手，让读者对 Web 应用程序的网络基础协议 HTTP 有所了解；介绍了 Web 应用程序请求应答的工作方式，让读者对 Web 的基本原理有一个较为清晰的认识；然后再对 Java EE 平台上 Web 应用程序开发的核心技术和技术规范进行介绍，如 Web 容器、Servlet、过滤器、监听器、JSP 等。在对核心技术本身进行介绍的同时，介绍如何应用这些技术以及应用这些技术的各种技巧，如上传文件、动态生成图片等。在介绍了核心技术之后，本书还从 Web 应用程序的整体架构角度介绍了使用 Java 开发 Web 应用程序的常用开发模式；介绍了各种 MVC 框架的基础技术 JSP 自定义标记；介绍了增强 Java Web 应用程序可靠性的异常处理机制和安全机制；最后介绍了用于异构系统整合的 Web 服务。

本书面向的读者

本书可作为普通高等院校计算机及相关专业课程教材，也可供具有一定 Java 语言编程基础，并想快速掌握使用 Java 开发 Web 应用程序技术的初学者及开发人员参考。

本书为了避免大量的源代码占用大量的篇幅导致阻碍初学读者的阅读，没有提供完整的 Web 应用系统作为实例，而是在各个章节中提供实用简短的实例方便读者学习。

由于作者水平有限，书中难免有不足和错误，欢迎广大读者批评指正。

编　者

2011 年 10 月

目　录

第 1 章　Java Web 工作原理

本章将介绍如下内容。

- HTTP 原理。
- 服务器端 Web 编程原理。
- Servlet 与 Web 容器。
- Java Web 应用程序的组成。
- Tomcat 介绍。

当读者已经学习和掌握了 Java SE 相关的技术知识后，就是开始利用 Java SE 所学内容，逐步向 Java 在 Web 上的开发应用迈进的时候了。

本章首先从理解 Java Web 编程的核心基础——HTTP 入手，详细解析 Web 的工作原理。然后，讲解服务器端动态网页编程的工作原理以及 Java 服务器端动态网页编程的解决方案。最后，我们将学习 Java Web 应用程序的组成以及如何打包和部署符合 Java EE 规范的 Java Web 应用程序。

1.1　解析 HTTP

互联网上的所有网站大部分都是使用 HTTP 构建的 Web 应用程序，一个基本的 Web 应用程序需要 Web 服务器和 Web 客户端浏览器。通常情况下，Web 服务器和 Web 客户端浏览器通过 HTTP 进行网络通信。其中，Web 服务器的作用是接收客户端请求，然后向客户端返回一些结果；浏览器的作用是允许用户请求服务器上的某个资源，并且向用户显示请求的结果，通常情况下浏览器从服务器得到的是一个 HTML 页面，HTML 页面可以告诉浏览器怎样向用户显示内容。

要真正理解 Web 的工作原理，必须理解 Web 服务器和浏览器之间如何通信，要理解它们之间是如何通信的，就必须理解 HTTP。

HTTP(HyperText Transfer Protocol，超文本传送协议)，是一套计算机在网络中通信的规则。在 TCP/IP 层次中，HTTP 属于应用层协议，位于 TCP/IP 的顶层。HTTP 是一种无状态的协议，意思是指在 Web 浏览器（客户端）和 Web 服务器之间不需要建立持久的连接。整个过程就是当一个客户端向服务器端发送一个请求（Request)，然后 Web 服务器返回一个响应（Response)，之后连接就关闭了。HTTP 遵循请求/响应（Request/Response）模型，所有的通信交互都被构造在一套请求和响应模型中，浏览 Web 时，浏览器通过 HTTP 与 Web 服务器

交换信息，Web 服务器向 Web 浏览器返回的内容都是与之相关的类型，这些信息类型的格式由 MIME(Multipurpose Internet Mail Extensions)定义。

注释：MIME 类型就是设定某种扩展名的文件用一种应用程序来打开的方式类型，当该扩展名文件被访问的时候，浏览器会自动使用指定应用程序来打开。

HTTP 定义的事务处理由以下四步组成，如图 1.1 所示。

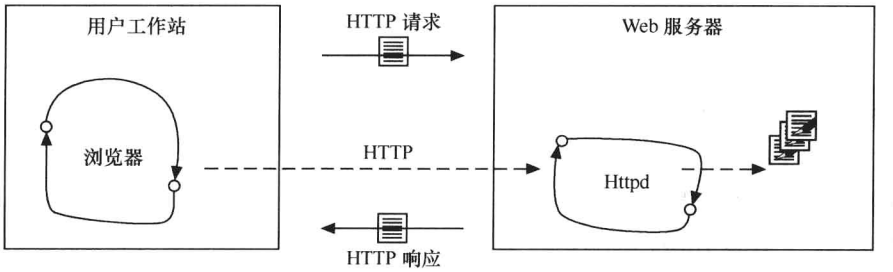


图 1.1　HTTP 工作流程

（1）客户端和 Web 服务器建立连接。

（2）客户端发送 HTTP 请求。

（3）服务器端接收客户端的 HTTP 请求，生成 HTTP 响应回发。

（4）服务器端关闭连接。客户端解析回发响应，恢复页面。

下面以一个简单的静态网页为例，逐步揭开 HTTP 的神秘面纱。假设用 HTML 编写一个网页 index.html，其内容如下。

```
<html>
  <head>
<title>欢迎！</title>
  </head>
  <body>
欢迎使用示例 Web 应用程序
```

假设我们使用的 Web 服务器是 Apache Web 服务器（该服务器是应用最为广泛的 Web 服务器），将这个网页放到本机的 Apache Web 服务器安装目录的\htdocs\book 子目录下，那么访问这个网页的完整路径就是 http://127.0.0.1/book/index.html。

可以使用 Google Chrome 浏览器、IE 8 浏览器的开发者工具中的网络监控来查看浏览器和 Web 服务器之间的通信内容。

下面逐步分析访问这个网页时 HTTP 的工作流程。

1.1.1　建立连接

在浏览器的地址栏键入 http://127.0.0.1/book/index.html，浏览器就打开了连接到 Web 服务器 HTTP 端口的一个 TCP/IP 连接，因为 TCP/IP 网络编程的最常用编程接口就是 socket 接口，所以建立 TCP/IP 连接就等同于建立一个 TCP socket 连接，socket 连接进行数据交换的实质是使用数据输入和输出流进行数据的读写。

如果用 Java 来编写浏览器，建立连接过程的代码如下。

```
Socket socket=new Socket("127.0.0.1",80);
InputStream in=socket.getInputStream();
```

……

OutputStream out=socket.getOutputStream();

1.1.2　发送请求信息

一旦建立了 TCP 连接，Web 浏览器就会向 Web 服务器发送请求，其实质就是将符合 HTTP 规范的请求内容写入 socket 连接的输出流中。HTTP 规范要求，一个 HTTP 请求，它必须包括请求行、请求头、消息体以及分隔请求头和消息体的一个空行，如图 1.2 所示就是一个 HTTP 请求数据。

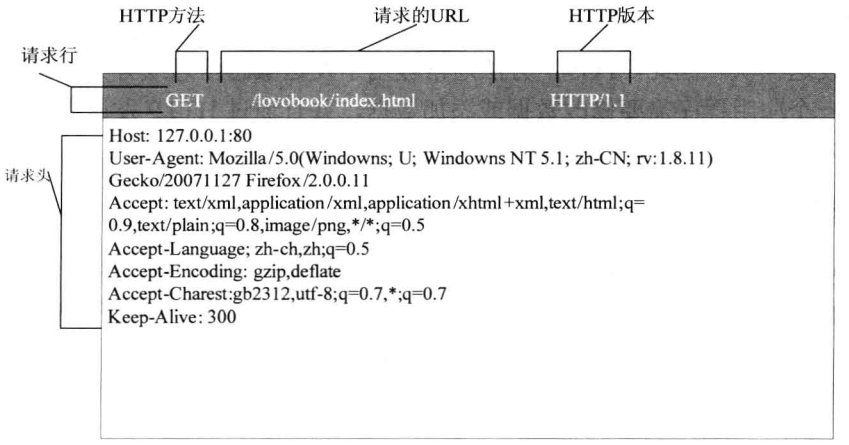


图 1.2　HTTP 请求示例

请求行是一个 ASCII 文本行，由三个标记组成：请求的 HTTP 方法、请求的 URL、HTTP 版本，中间用空格分开。例如：

GET/book/index.html HTTP/1.1

在 HTTP 1.1 版本中，定义了 8 种 HTTP 请求方法。

- GET：用于向服务器检索资源。
- POST：用于向服务器发送数据，并要求指定的 URI 处理。
- HEAD：与 GET 方法相同，服务器只返回状态行和头标，并不返回请求文档。
- PUT：请求服务器保存请求数据作为指定 URI 新内容。
- DELETE：请求服务器删除 URI 中命名的资源。
- OPTIONS：请求关于服务器支持的请求方法信息。
- TRACE：请求 Web 服务器反馈 HTTP 请求和其头标。
- CONNECT：已文档化但当前未实现的一个方法，预留做隧道处理。

在这些方法中，最常用的是 GET 和 POST 方法，其他几种方法对于 Web 开发者来说不是很常用。GET 方法用于向服务器检索信息，它可以以查询字符串的形式提供有限长度的数据。但是，GET 方法却不能发送大量信息。很多 Web 服务器会限制完整 URL 的长度为 255 个字符，超过这个长度的信息通常被忽略了。因此，GET 方法适合发送我们不介意在 URL 中可见的少量信息。此外，GET 方法只能用在 Web 应用程序中执行查询，而不能用于执行更新。

以下方式的 HTTP 请求为 GET 方法。

- 在浏览器地址栏输入 URL。
- 在当前网页上点击 HTML 链接。
- 在 HTML 表单中设置表单的 method='get'，并提交表单。
- 在 HTML 表单中没有设置表单的 method 属性，提交表单（如果不设置 method 属性，默认为 get 方法）。

注释：查询字符串是一个以问号“？”开始，后跟名称/值对的字符串列表。多个名称/值对之间用“&”分隔，例如：/index.html?name=admin&password=admin。

POST 方法用于向服务器发送数据，并要求指定的 URI 处理。与 GET 方法相似，POST 可以有查询字符串，但是 POST 方法用完全不同的机制传递信息。POST 方法可以将无限制数量的数据作为 HTTP 请求的一部分，通过套接字连接发送。数据不会作为 URL 的一部分出现，而且只发送一次。因此，POST 方法通常被用于发送敏感信息或者大量信息，或者上传文件。如果一个应用程序需要修改数据或添加数据，并且通过 HTTP 发送请求，那么就应该使用 POST 方法。

1. 请求头

HTTP 使用 HTTP 头来传递请求的元信息。HTTP 头是一个用冒号分隔的名称/值对，冒号前面是 HTTP 头的名称，后面是 HTTP 头的值。常见的请求头由用户代理资料、可以接收的格式、语言以及内容编码等组成，这些信息告诉服务器客户端是什么，客户端想要得到什么格式的回馈信息等。

2. 空行

发送回车符和退行，通知服务器下面不再有请求头。

3. 消息体

HTTP 请求中带有查询字符串时，如果是 GET 方法，查询字符串或表单数据附加在请求中，那么消息体中就没有内容；如果是 POST 方法，查询字符串或表单数据就添加在消息体中。

1.1.3 回送响应信息

Web 服务器解析请求，定位并读取指定的资源 http://127.0.0.1/book/index.html，将文件以及其他信息组成 HTTP 响应返回到客户端。HTTP 响应包括状态行、响应头、消息体以及分割消息头和响应头的一个空行，如图 1.3 所示是 HTTP 分析器捕获的 HTTP 响应数据。

1. 状态行

每个 HTTP 响应以一个状态行开头。状态行由 HTTP 版本、响应状态码和响应描述组成，三者之间用空格分隔。

响应状态码是一个三位的数字，它分为如下几个组。

- 1xx：信息，请求收到，继续处理。
- 2xx：成功，行为被成功地接收、理解和采纳。

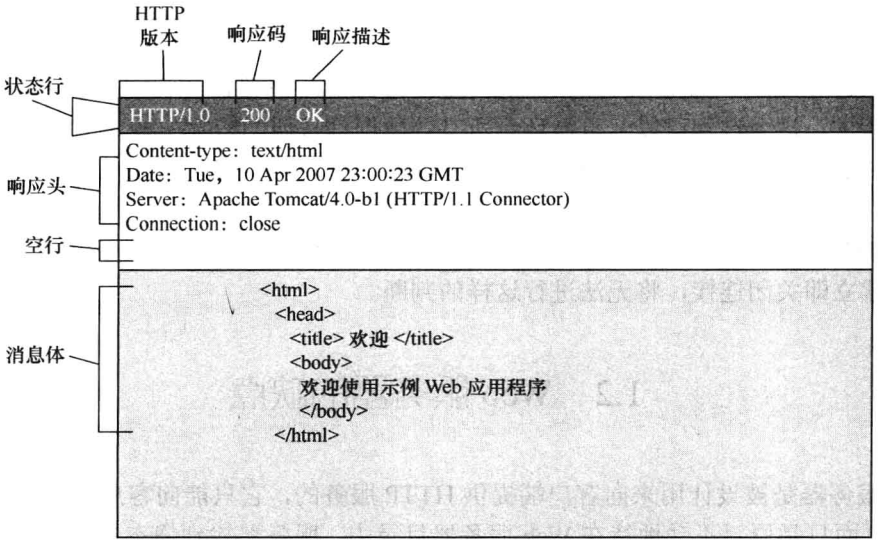


图 1.3　HTTP 响应示例

- 3xx：重定向，为了完成请求，必须进一步执行的动作。
- 4xx：客户端错误，请求包含语法错误或者请求无法实现。
- 5xx：服务器错误，服务器不能实现一种明显无效的请求。

每一个响应状态码都有相关联的字符串响应描述。我们经常看到的响应状态码是 200，它表明成功执行，我们得到了一个有效的响应。

2. 响应头

响应头与请求头一样，也是一个用冒号分隔的名称/值对，冒号前面是 HTTP 头的名称，后面是 HTTP 头的值。典型的响应头包括内容类型描述、内容长度、时间邮戳、服务器信息、内容最后更改的时间。这些信息帮助客户端判断发送过来的内容是什么、内容的大小、数据是否比前一次响应更新。

在响应头中，最重要的 HTTP 头是 Content-Type，它指定 MIME 类型。MIME 类型告诉浏览器要接收的数据是什么类型，这样浏览器才知道如何显示这些数据。这个值通常与 HTTP 请求头中的 Accept 相关。

3. 空白行

最后一个响应头之后是一个空行，发送回车符和退行，表明下面不再有响应头。

4. 消息体

要发送给客户端的 HTML 文档或其他要显示的内容等。Web 服务器把要发送给客户端的文档信息放在消息体中。

1.1.4　关闭连接

HTTP 响应到达客户端后，浏览器先解析 HTTP 响应中的状态行，查看请求是否成

功的状态代码。然后解析每一个响应头，读取响应消息体，将消息体显示在浏览器页面上。

一个 HTML 文档可能包含其他的需要被载入的资源，浏览器会识别，并对这些资源再进行额外的请求，这个过程可以一直循环到所有的数据都按照响应头中规定的格式显示到页面中。数据传送完毕，服务器端关闭连接，随着连接的关闭与连接关联在一起的状态也就消失了。如果不采用特定的状态保持技术，Web 无法判断前后两次请求是否来自同一个浏览器，因为采用 TCP/IP 协议通信判断是否是同一个客户端的方法就是看是不是同一个连接，每次请求应答后就立即关闭连接，将无法进行这样的判断。

1.2 Web 服务器的缺点

Web 服务器是被设计用来向客户端提供 HTTP 服务的，它只能向客户端提供静态网页内容。静态页面只是原封不动地待在 Web 服务器目录中，服务器找到静态页面，并把它原样传回到客户端。每个客户端看到的页面都是一样的。

假如用户需要动态页面（在发出请求之前还不存在的动态创建的页面），而且还希望把通过表单或查询字符串提交的数据保存到服务器上（即写到一个文件或数据库中），怎么办呢？解决方案是在运行 Web 服务器软件（如 Apache Web 服务器软件）的 Web 服务器主机上增加一个辅助应用，这个辅助应用负责生成动态页面，并且能与 Web 服务器软件通信。

当用户在客户端浏览器上点击一个链接，它的 URL 指向一个辅助应用，而不是一个静态页面。Web 服务器应用（例如 Apache Web 服务器软件）看出这个请求是给一个辅助程序的，所以 Web 服务器启动并运行这个程序。Web 服务器应用还把 GET 或 POST 请求提供的参数一并交给这个辅助应用。辅助应用创建一个全新的动态页面，然后把这个 HTML 返回给 Web 服务器。那么，对于 Web 服务器来说，来自辅助应用的 HTML 是一个静态页面。辅助应用关闭，客户端得到一个 HTML 页面，整个过程如图 1.4 所示。

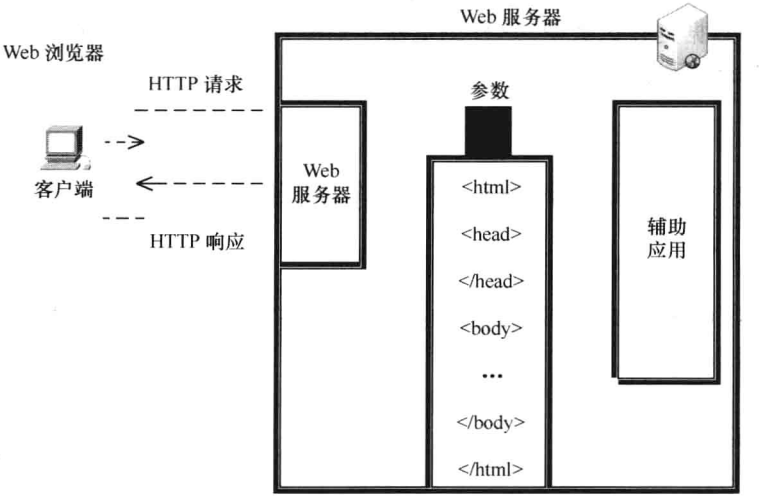


图 1.4 Web 服务器与辅助应用协同创建动态页面

1.3　服务器端网页编程

Web 服务器创建动态服务器端内容的过程被称为服务器端网页编程。服务器端网页编程的技术有多种，包括最早的 CGI 技术、微软的 ASP 和 ASP.NET 技术、开源的 PHP、基于 Java 的 Servlet/JSP 技术等。

1.3.1　最早的解决方案 CGI

CGI，即通用网关接口（Common Gateway Interface），是最早用于创建动态服务器端内容的一种技术。使用 CGI，Web 服务器可以将客户端的请求传递给一个外部程序，这个外部程序可以执行、创建内容，并且将响应传递给客户端，如图 1.5 所示。

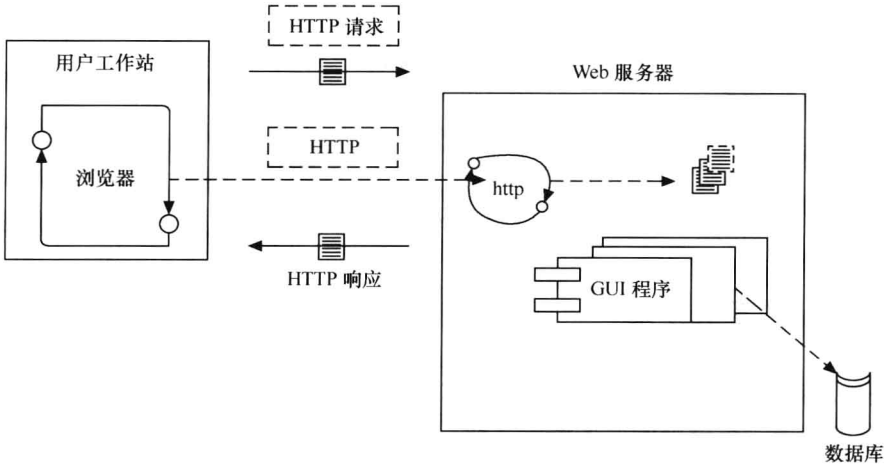


图 1.5　CGI 程序在 Web 服务器上的部署

CGI 最初出现的时候，它的功能对于静态页面来说有了极大的提升，为 Web 开发者提供了更多的功能。因此，CGI 开始迅速流行起来，并且成为创建动态网页的标准方法。CGI 的最大优点是它可以用 Shell、Perl、C、PHP、Python 等编写。但是，CGI 并不是完美的。

CGI 最初是被设计用来作为 Web 服务器与外部应用程序进行通信的标准方法，用于产生动态网页的功能实际上是设计目标的次要结果。使用 CGI 作为 Web 服务器的辅助应用时，每次请求一个 CGI 资源，将在服务器上创建一个新的进程（Process），并且通过标准输入和环境变量将信息传递给该进程。图 1.6 和图 1.7 分别是 CGI 处理一个和多个请求时的示意图。

从图中可以看出，在 CGI 运行时，每次请求就会打开一个 CGI 进程的方式严重消耗服务器资源，极大地限制了一个服务器可以支持的并发 CGI 用户数量。假设对操作系统和进程不熟悉，那么一个好的方法就是，每次用户创建一个请求就要启动和停止一个 Web 服务器。启动和停止一个进程需要大量时间，一个更好的解决方案是启动服务器进程一次，处理所有请求，然后在 Web 服务器不再需要时将其停止。启动和停止一个 Web 服务器时，如果 CGI 对此处理不好，经常会使 Web 服务器挂起。

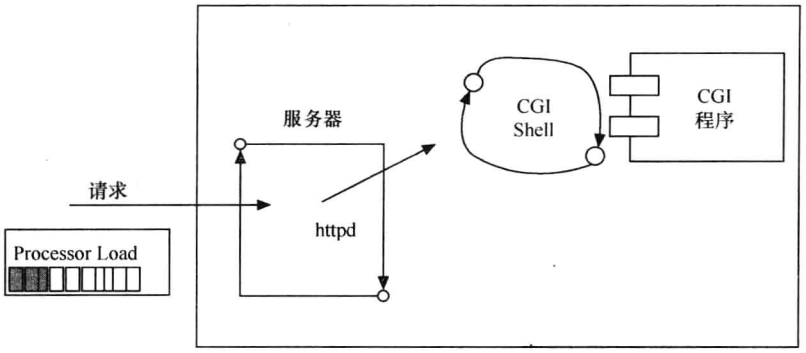


图 1.6　CGI 处理一个请求

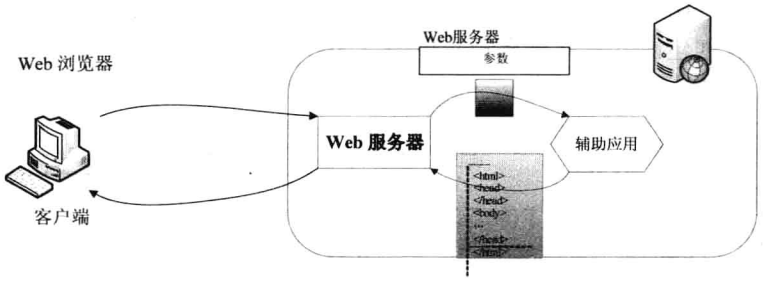


图 1.7　CGI 处理多个请求

1.3.2　Java 的解决方案

在 Java EE 标准中，Servlet 以及 Web 容器被设计用来解决动态创建 HTML 内容的问题，为 Web 开发者创建一个健壮的服务器端环境。Servlet 与 Web 容器在 Web 服务器上的部署如图 1.8 所示。

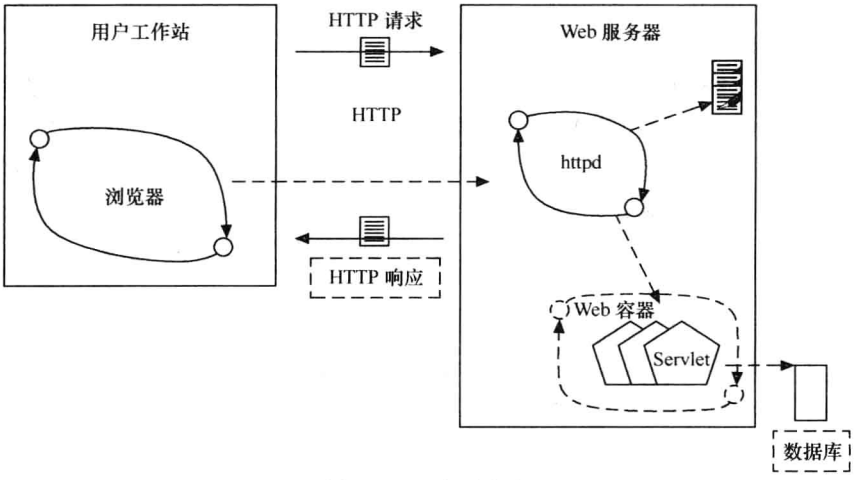


图 1.8　Java 解决方案

1. Servlet

Servlet 是一个在 Web 服务器端或者应用服务器端运行的 Java 程序，主要用于在服务器端产生动态内容。与我们已经学习过的其他 Java 技术一样，Servlet 是与平台无关的 Java 类，能够编译成与平台无关的字节码，从而被基于 Java 技术的 Web 服务器动态装载和运行。

下面是一个简单的 Servlet 代码示例。

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*

public class HelloWorld extends HttpServlet {
    public void doGet(HttpServletRequest request.
HttpServletResponse response)
        throws IOException,ServletException
{
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Hello World!</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Hello World!</h1>");
        out.println("</body>");
        out.println("</html>");
}
}
```

分析以上代码，可以得出以下结论。

（1）Servlet 是常规的 Java 代码。代码中用了一些新的 API，但是不涉及新的语法。

（2）Servlet 代码中有我们不熟悉的重要语句。Servlet 不属于 J2SE，它属于 Java EE 规范。

（3）Servlet 对标准类（HttpServlet）进行了扩展。

（4）Servlet 没有 main()方法。

Servlet 在服务器端的工作主要是执行如下任务。

- 读取客户端发送的显式和隐式数据。客户端发送的显式数据（包括最终用户在网页的 HTML 表单中输入的数据以及查询字符串），以及隐式数据（HTTP 信息）都被封装在 HTTP 请求中。
- 生成结果。在此过程中可能要访问数据库，执行 RMI 或 EJB 调用，调用 Web 服务，或者直接计算得出对应的响应。实际数据可能存储在关系型数据库中。该数据库可能不理解 HTTP，或者不能返回 HTML 形式的结果，所以 Web 浏览器不能直接与数据库进行会话，即使它能做到这点，从安全角度考虑，也不希望让它这样做。为此，需要 Web 中间层从 HTTP 流中提取输入数据，与应用程序会话，并将结果嵌入文档中。
- 向客户端发送显式数据（即文档）和隐式数据（HTTP 响应头中的数据）。这个文档中可以用各种格式发送，包括文本（HTML 或 XML）、二进制（GIF 图像），甚至可以是建立在其他底层格式上的压缩格式，例如 GZIP。但是，到目前为止，HTML 是最常用的格式，所以 Servlet 的重要任务之一就是将结果包装到 HTML 中。显式数据和隐式数据最终都被封装在 HTTP 响应中。

理解 Servlet，是掌握 Java Web 编程的核心，这部分知识将在第 2 章、3 章、4 章、5 章详细讲解。

2. Web 容器

从上面的代码中已经知道 Servlet 类中没有 main()方法。那么，Servlet 是如何运行的呢？实际上，Servlet 受控于另一个 Java 应用程序，这个 Java 应用程序称为 Web 容器（Container）。Web 容器负责管理和运行 Servlet，但是由此会带来一些额外的开销，这样值得吗？容器对 Servlet 的支持包括如下内容。

- 通信支持。利用容器提供的方法能轻松地让 Servlet 与 Web 服务器对话。不用我们自己建立 ServletSocket、监听某个端口、创建流等。容器知道自己与 Web 服务器之间的协议，所以我们的 Servlet 不用担心 Web 服务器和我们自己的 Web 代码之间的 API。要考虑的只是如何在 Servlet 中实现业务逻辑。
- 生命周期管理。容器控制着 Servlet 的生与死。它会负责加载类、实例化和初始化 Servlet、调用 Servlet 方法，以及使 Servlet 实例能够被垃圾回收。有了容器的控制，就不需要太多地考虑资源管理了。
- 多线程支持。容器会自动地为它接收的每个 Servlet 请求创建一个新的 Java 线程。针对客户的请求，如果 Servlet 已经运行完相应的 HTTP 服务方法，这个线程就会结束。由服务器创建和管理多个线程来处理多个请求，可以让我们省掉很多工作。
- JSP 支持。容器会自动把 JSP 翻译成为 Servlet Java 代码。
- 处理安全性。Java Web 应用程序通常需要实现安全性控制，限制用户可以访问的资源。SUN Servlet 规范中规定 Web 容器必须实现访问权限控制，这样我们在编写 Servlet 程序时，就不用考虑访问安全性问题，Web 容器会帮我们完成。

实际上，Web 容器主要是用于给容器中应用程序组件（Servlet、JSP）提供一个环境，使 Servlet、JSP 直接跟容器中的环境变量交互。这样，我们就可以把注意力放在用 Servlet 实现业务逻辑上，而不必为系统底层服务（例如线程管理、安全性和网络通信等）编写代码。

Sun 关于 Web 容器的官方定义在 Servlet 规范中都有详细的描述。Servlet 规范只定义了一个容器必须实现的标准功能。Servlet 容器有很多不同的实现，不同的厂商生产的容器有不同的价格、性能和功能。这就让 Servlet 开发者开发软件时有更多的选择。常用的 Web 容器实现有 Tomcat、Jboss、WebLogic、WebSphere、Oracle9i AS 等。其中 Tomcat 服务器是一种用得很广泛的 Web 容器，经常用于中小型企业的 J2EE 解决方案中，它是一个开放源代码的免费的中间件产品。现在我们已经了解了关于容器的一些知识，我们知道要运行 Servlet，必须安装一个容器。本书采用的是 Tomcat 5.0 以上版本作为示例容器，在后面的小节中将详细介绍 Tomcat。

3. Servlet 与 Web 容器配合处理请求和响应

与 CGI 类似，Servlet 允许一个请求被一个程序处理，并且使用同样的程序产生动态的响应。此外，Servlet 特别定义了一个有效的生命周期，使得用单个进程管理所有请求成为可能。它消除了 CGI 的多进程缺陷，允许主进程在多个 Servlet 和多个请求之间共享资源。图 1.9 和图 1.10 是 Servlet 处理一个和多个请求时的示意图。