第十八章

Web服务器中的Java：Servlet与JSP

18.0. 简介

本章将讲述Java在Web服务器中的应用，但是不包括如何用Java编写CGI程序。虽然这完全可能，但不是很有效率。CGI程序的整个理念都有些过时了。每次调用CGI程序时，Web服务器都要创建新的重量级进程运行它，这样显然效率高不了。如果用Java编写，程序每次还要转换成机器代码，效率就更差了。

当今的大势所趋是把功能内置在Web服务器中：Microsoft ASP、PHP、Java Servlet和JSP（注1）都是例子。它们一般都不需要为每次请求创建单独的进程。基于Java的解决方案都运行在Web服务器中的线程（参见第二十四章）里，使用JIT（即时）运行时系统，Java字节码很长时间才需要转换成机器代码一次。自然了，本书要集中讨论的是Java解决方案。

本章我们将讲两个例子。任务将是显示一个网页，其中有5个随机选出的整数（彩票迷肯定喜欢）。所需Java代码非常简单：

```
// 文件netweb/servlets_jsp/FiveInts.java的一部分
Random r = new Random();
```

注1： 有人说Sun在拷贝什么东西时，总会进行改进，并在自己产品的名字中留有原作的痕迹。想一想Microsoft ODBC和Java JDBC；Microsoft ASP和Java JSP。大多数大公司都不会这样做。

```
for (int i=0; i<5; i++)
    System.out.println(r.nextInt());
```

但是当然了，我们不能只是运行一下，并把输出保存在HTML文件里就行了，应该让每个人看到的页面都有不同的数字。如果要把这些都放到网页中，就得写代码输出［用`Println()`］一些HTML了。这就要用到Java Servlet。

Servlet代码可能会有点乱，因为必须将字符串中的双引号转义。如果网站管理员还要改动HTML就更糟了，他得获得程序员的源代码，并恳求进行改动。想像一下，我们给网站管理员的页面包含HTML和Java代码，然后无论HTML何时改变，都要像变魔术似的把页面编译成Java，怎样才能做到呢？不用再想了，市场上已经有一种炙手可热的技术了，这就是JSP。

第二个例子是一部字典（术语列表）。我会用Servlet和JSP实现它。

我不想讲如何安装Servlet引擎，或者怎么安装Servlet。但如果你还没有安装，我推荐到*http://jakarta.apache.org*下载Tomcat。Tomcat是Sun指定的Servlet和JSP的官方参考实现。它也是（从URL可以猜到）广受欢迎的Apache Web服务器的官方Servlet引擎。

18.1. 第一个Servlet：生成HTML页面

问题

一个Servlet，向用户显示一些信息。

解决之道

覆盖`HttpServlet`方法`service()`或`doGet()/doPost()`。

讨论

抽象类`javax.servlet.Servlet`是为那些要围绕Servlet创建完整Web服务器的人设计的。例如，在Sun公司的Java Web服务器中，有一个Servlet子类处理普通

HTML 页面，另一个子类处理 CGI 程序，等等。除非你要自己写一个 Web 服务器，否则用不着从这个类扩展，用得着的是它的子类 `HttpServlet`，在 `javax.servlet.http` 中。该子类有如下方法：

```
public void service(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException;
```

`service()` 方法有两个参数：`request`（请求）和 `response`（响应）。request 中包含所有来自浏览器的请求的有关信息，包括输入流（如果要读数据）。response 包含要返回给浏览器的响应信息，包括写回给用户的输出流。

但 Web 中将数据传给网页有几种 HTTP 方法。这对普通 HTML 并不重要，但在处理表单（也就是需要用户填入信息或进行选择的网页）时，这种区别就值得注意了。简言之，HTTP 的 GET 方法将所有的表单数据都附加在 URL 中传送。比如，GET 的 URL 可能如下：

```
http://www.acmewidgets.com/cgi-bin/ordercgi?productId=123456
```

其优点是用户可以添加书签（bookmark），以免多次填同一表单。但 URL 仅 lKB 的总长度是有限的。因为必须是一个字符串，要进行编码把空格、制表符、冒号及其他字符都用两个十六进制位表示：`%20` 是十六进制的 20，或者表示 ASCII 空格符。相比之下，POST 方法把所有参数都作为套接字连接的输入，放在 HTTP 首部之后。

`service()` 方法的默认实现会指明用哪种方法来调用 Servlet。由它再分派到正确的方法：如果是 GET 请求，就用 `doGet()`；如果是 POST 请求，就用 `doPost()`，同时传递 `request` 和 `response` 参数。因此，虽然理论上可以覆盖 `service()` 方法，但覆盖 `doGet()`, `doPost()` 或两个同时覆盖，更加常见（也是 Sun 公司的推荐方式）。

最简单的 `HttpServlet` 如例 18-1 所示。

例 18-1：HelloServlet. Java

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

/** 简单的 Hello World Servlet
 */
public class HelloServlet extends HttpServlet{
```

```
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        PrintWriter out = response.getWriter();
        response.setContentType("text/html");
        out.println("<H1>Hello from a Servlet</H1>");
        out.println("<P>This servlet ran at ");
        out.println(new Date().toString());
        out.println("<P>Courtesy of HelloServlet.java 1.2 ");
    }
}
```

此程序输出如图 18-1。

图 18-1　例 18-1 的输出

用 Servlet 当然能实现更多内容。假设要从一个 Servlet 输出字典（术语及其含义的列表）。代码与例 18-1 很像，只不过要用 `doGet()`方法代替 `doPost()`。如例 18-2 所示。

例 18-2：TermsServlet.java

```
/** 输出字典项的 Servlet
 */
public class TermsServlet extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse resp) throws
IOException {
        PrintWriter out = resp.getWriter();
        out.println("<HTML>");
        out.println("<TITLE>Ian Darwin's Computer Terms and Acronyms</TITLE>");
```

```
            out.println("<BODY>");
            out.println("<H1>Ian Darwin's Computer Terms and Acronyms</H1>");
            out.println("<TABLE BORDER=2>");
            out.println("<TR><TH>Term<TH>Meaning</TR>");

            // 此部分Servlet生成许多行，如
            //    <TR> <TD>JSP <TD>Java Server Pages, a neat tool for ...
            TermsAccessor tax = new TermsAccessor("terms.txt");
            Iterator e = tax.iterator();
            while (e.hasNext()) {
                Term t = (Term)e.next();
                out.print("<TR><TD>");
                out.print(t.term);
                out.print("<TD>");
                out.print(t.definition);
                out.println("</TR>");
            }
            out.println("</TABLE>");
            out.println("<HR></HR>");
            out.println("<A HREF="servlet/TermsServletPDF
                 ">Printer-friendly (Acrobat PDF) version</A>");
            out.println("<HR></HR>");
            out.println("<A HREF="mailto:compquest@darwinsys.com/subject=Question
                 ">Ask about another term</A>");
            out.println("<HR></HR>");
            out.println("<A HREF="index.html">Back to HS</A> <A HREF="../
                 ">Back to DarwinSys</A>");
            out.println("<HR></HR>");
            out.println("<H6>Produced by $Id: TermsServlet.java,v 1.1 2000/04/06
                 ian Exp $");
            out.print(" using ");
            out.print(tax.ident);
            out.println("</H6>");
        }
    }
```

Servlet 的调试技巧

许多 Servlet 引擎（如 Allaire JRun）会在许多不同目录下生成大量很小的日志文件。花一些时间研究你自己的引擎堆栈轨迹记录、标准错误和输出以及其他信息，是非常值得的。

另外可以参见实例 16.5，其中叙述了 Servlet 或其他服务器组件如何与基于网络的日志记录工具通信。

18.2. Servlet：处理表单参数

问题

要处理 Servlet 中一个 HTML 表单的数据。

解决之道

使用 `request` 对象的 `getParameter()`方法。

讨论

HTML 页面上 FORM 里每个惟一命名的 INPUT 元素，都在 `request` 对象的参数列表中有对应的一项。可以作为一个枚举获取，但只请求其中一个更常见。图 18-2 所示为一个简单的表单，询问如何生成随机数，以及生成多少个？

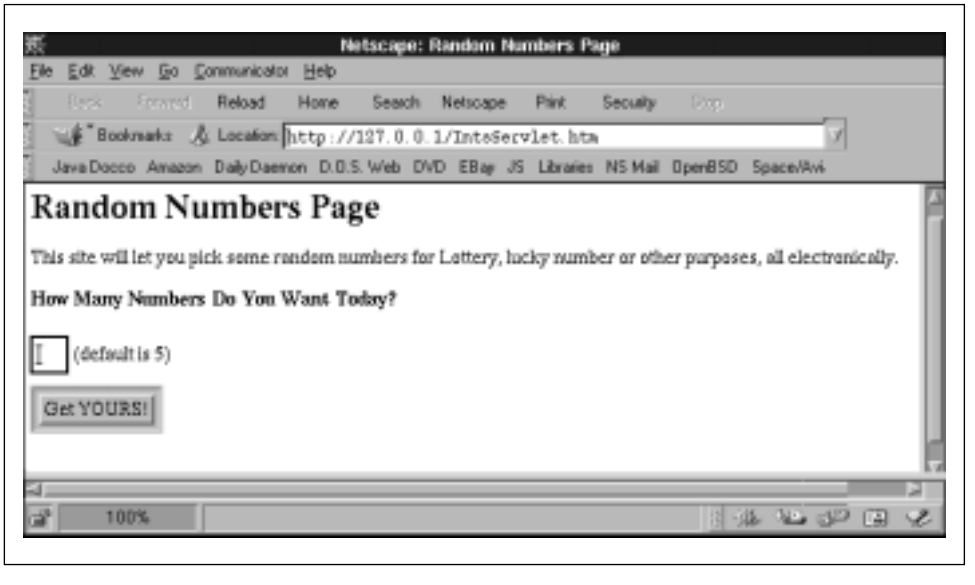

图 18-2　随机数 HTML 页面

在输入区域中输入 8，并按下“ Get Yours！ ”按钮时，可以看到图 18-3 所示屏幕图。

这里怎么工作的呢？程序显然由一个 HTML 页面和一个 Java Servlet 组成。HTML 页面如例 18-3 所示。注意 FORM 项和 INPUT 区域。

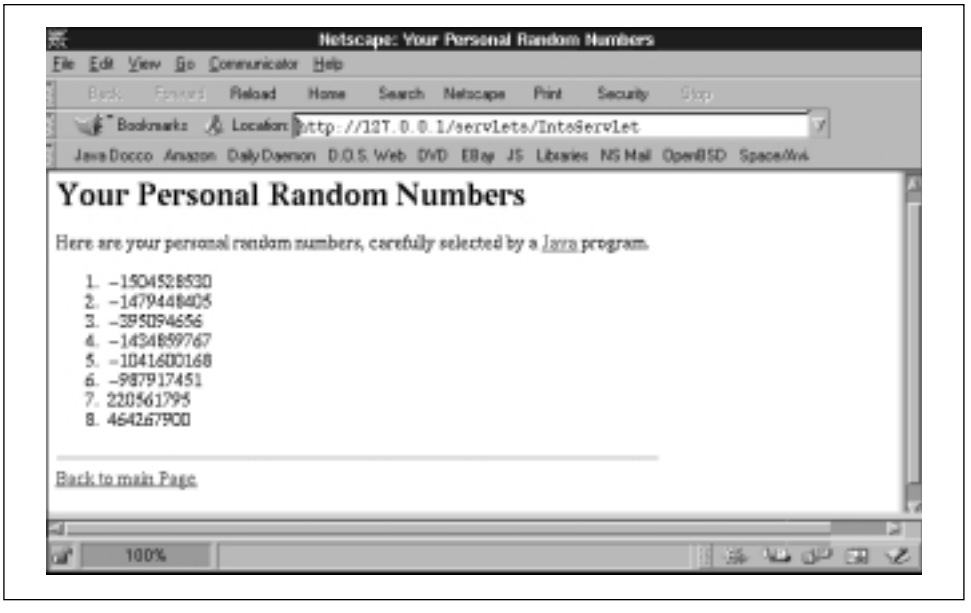

图 18-3　随机数 Servlet 的输出

例 18-3：IntsServlet.htm

```
<HTML>
<HEAD><TITLE>Random Numbers Page</TITLE></HEAD>
<BODY BGCOLOR="white">
<H1>Random Numbers Page</H1>
<P>This site will let you pick some random numbers for Lottery, lucky number
or other purposes, all electronically.</P>
<FORM METHOD=POST ACTION="/servlets/IntsServlet">
<H4>How Many Numbers Do You Want Today?</H4>
<INPUT NAME=howmany SIZE=2> (default is 5)
<BR>
<INPUT TYPE="SUBMIT" VALUE="Get YOURS!">
</FORM>
</BODY></HTML>
```

例 18-4 为 Java Servlet。注意 `getParameter()`的使用。

例 18-4：IntsServlet.java

```
import java.io.*;
import java.util.Random;
```

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

public class IntsServlet extends HttpServlet {
    protected final int DEFAULT_NUMBER = 5;
    /** 当用户填入表单时调用 */
    public void doPost(HttpServletRequest req, HttpServletResponse resp)
    throws IOException {
        resp.setContentType("text/html");
        PrintWriter out = resp.getWriter();

        // 常规的 HTML 设置
        out.println("<HTML>");
        out.println("<HEAD>");
        out.println("<BODY BGCOLOR=\"white\">");

        // 此页的HTML
        out.println("<TITLE>Your Personal Random Numbers</TITLE>");
        out.println("<H1>Your Personal Random Numbers</H1>");
        out.println("<P>Here are your personal random numbers,");
        out.println("carefully selected by a");
        out.println("<A HREF=\"http://java.sun.com\">Java</A> program.");
        out.println("<OL>");

        // 求取要输出多少个数字
        int n = DEFAULT_NUMBER;
        String num=req.getParameter("howmany");
        if (num != null && num.length() != 0) {
            try {
                n = Integer.parseInt(num);
            } catch (NumberFormatException e) {
                out.println("<P>I didn't think much of ");
                out.println(num);
                out.println(" as a number.</P>");
            }
        }

        // 产生随机数
        Random r = new Random();
        for (int i=0; i<n; i++) {
            out.print("<LI>");
            out.println(r.nextInt(49));// for Lotto 6/49
        }
        out.println("</OL>");

        // 输出一条分隔线和一个返回链接
        out.println("<HR></HR>");
        out.println("<A HREF=\"index.html\">Back to main Page</A>");
        out.println("</HTML>");
    }
}
```

参考

在线资源中的 `OrderServlet`，一个更长一些的例子。

18.3. cookie

问题

要让客户端（浏览器）记住一些信息。

解决之道

做一个 cookie，用它使客户端能够记忆我们的响应。

讨论

cookie 最初是 Netscape 公司作为一种调试技术发明的，但现在却已经无所不在：所有现代浏览器，包括 MSIE，以及 Lynx 这样的文本浏览器都可以接受并保存它。cookie 本质上是一小条服务器端生成并发给客户的文本信息（一个名－值对）。浏览器能记住它们。（非暂时的 cookie 将存在硬盘中，例如 Netscape 创造的称为 *cookie* 或 *cookie.txt* 的文件。）然后浏览器在以后再次访问同一网站的页面时将它们发回。`Cookie` 类是 `javax.servlet.http` 包的一部分，因此 Servlet 实现产品中都有。构造方法的参数是一个名称及其值，但也可以设置其他参数。最重要的是终止期限（expiry time），即第一次发送后以秒计的时间。默认为 `-1`。如果值为负，cookie 不存盘，而以暂时形式存在，浏览器一经退出，就会消失。对于存盘的 cookie，终止期限转化为 1970 年 1 月 1 日为基准的值，这个时间是 Unix 时代的开始，也是现代计算机时代的开始。

当浏览器访问已发送过一个或多个 cookie 的网站时，它会把它们都作为 HTTP 首部的一部分返回。用 `getCookies()` 方法，可以将其全部获得（作为一个数组），然后可以进行遍历找到所需的一个。

```
for (int i=0; i<mySiteCookies.length; i++) {
    Cookie c = mySiteCookies[i];
    if (c.getName().equals(name-you're-looking-for)) {
```

```
                someString = c.getValue();
                break;
        }
    }
```

假设要让用户为 Servlet 选择一种喜爱的颜色，作为此后页面的背景色。我们用 `prefs.bgcolor` 作为颜色 cookie 的名字。主 Servlet 是 `Cookie Servlet`，用来检测此cookie。如果以前来设置颜色，它会跳到一个HTML 页中，然后通过另一Servlet 最终返回。如果以前设置过，`Cookie Servlet`（如例 18-5 所示）就会用用户选的颜色显示欢迎页面。

例 18-5：CookieServlet. java

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

/** 简单的基于 Cookie 的页面颜色显示 Servlet 演示程序
 */
public class CookieServlet extends HttpServlet {
    /** 喜爱的 cookie 名字 */
    protected final static String PREFS_BGCOLOR = "prefs.bgcolor";
    /** 如未指定去哪里 */
    protected final static String CUSTOMIZER = "/ColorCustomize.html";
    /** 用户选择的颜色 */
    protected String faveColor = null;

    public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

        // 遍历所有 Cookie，录找 faveColor
        Cookie[] mySiteCookies = request.getCookies();
        for (int i=0; i<mySiteCookies.length; i++) {
            Cookie c = mySiteCookies[i];
            if (c.getName().equals(PREFS_BGCOLOR)) {
                faveColor = c.getValue();
                break;
            }
        }

        // 如果未发现 faveColor
        // 转到定制 Servlet
        if (faveColor == null) {
            ServletContext sc = getServletContext();

            // 需要 Servlet API 2.1 或更高版本
            // RequestDispatcher rd =
            //  sc.getRequestDispatcher(CUSTOMIZER");
            //rd.forward(request, response);
```

```
            // 较旧版本的做法
            response.sendRedirect(CUSTOMIZER);
        }

        // 已有颜色了，生成页面
        PrintWriter out = response.getWriter();
        response.setContentType("text/html");

        out.println("<html><title>A Custom-Colored Page</title>");
        out.print("<body bgcolor=\"");
        out.print(faveColor);
        out.println("\">");
        out.println("<P>Welcome! We hope you like your colored page!</P>");
        out.println("</body></html>");
        out.flush();
    }
}
```

如果用户还未设置颜色定制 cookie，`CookieServlet` 将控制权传给如下 HTML 页面。（通过在老 API 中发送一个 HTTP 重定向，或用 Servlet API 1.2 或更高版本中的 `ServletDispatcher`）。

```
<BODY BGCOLOR="pink">
<H1>Please choose a color</H1>
<FORM ACTION="/servlet/ColorCustServlet" METHOD=GET>
<SELECT NAME="color_name">
     <OPTION VALUE="green">Green</>
     <OPTION VALUE="white" SELECTED>White</>
     <OPTION VALUE="gray">Grey</>
</SELECT>
<INPUT TYPE="submit" VALUE="OK">
</FORM>
```

最后，HTML 将跳到定制 Servlet（例 18-6），后者包含如下代码，可将用户选择存为 cookie，然后通过发送一个 HTTP 重定向返回 `CookieServlet`，这样会使浏览器装入指挥的替代页面。

例 18-6：Color Cust Servlet.java

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

/** 颜色定制Servlet */
public class ColorCustServlet extends HttpServlet {
    protected final static String DEFAULT_COLOR = "white";
    protected String faveColor = DEFAULT_COLOR;

    public void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
        throws IOException {
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();

            String cand=request.getParameter("color_name");
            if (cand != null) {
                faveColor = cand;
                Cookie c = new Cookie(CookieServlet.PREFS_BGCOLOR, faveColor);
                c.setMaxAge(60*60*24*365);
                response.addCookie(c);
            }
            response.sendRedirect("/servlet/CookieServlet");
        }
    }
```

当然，使用cookie时有些问题需要考虑。有些用户会因为害怕网站用cookie收集超出正常范围的信息而禁用cookie。这种情况下，我们的Servlet会返回定制页面。也可以输出警告说“必须启用cookie浏览此网站。”或使用其他技术，如会话跟踪（见实例18.4）。

实际中可能想保存的用户选项常常不止一条。比如，除了设置屏幕背景之外，还需要设置文字的颜色。可能将选项存入服务器端的数据库中，并设置一个记号来确认用户（可能是数据库主键）。但即使是这种时候，也要记住cookie是会改变的。参见实例18.11中的程序，它可以修改保存在硬盘上的cookie。

18.4. 会话跟踪

问题

在同一次浏览器会话里调用的多个Servlet中跟踪同一用户。

解决之道

使用`HttpSession`对象。

讨论

HTTP本身被设计成一种无状态协议，也就是说，当我们与服务器连接，下载一份实验室报告后，这次会话就结束了。后来人们越来越多地将HTTP用于交互式应用，

例如网上商店中的购物篮，在线测试中跟踪答案，或是网络游戏中进行移动。这时 HTTP 会话的概念就发展成对某个浏览器进行跟踪。我们可以用 cookie（见实例 18.3）或在 URL 中添加会话标识符来标识会话。无论是什么情况，会话都会在用户的浏览器程序退出时结束，但可以保持很长时间（这里可能是拒绝服务攻击的藏身之所，请小心了）。

在 Servlet API 中使用会话很简单。可以向传入 `service()`或 `doGet()/doPost()` 方法的 `HttpRequest` 请求 `HttpSession` 对象。该对象与 `Hashtable`（参见实例 7.6）很像，只不过方法名成了 `putValue()`和 `getValue()`。我们可以借此将会话中任意数月的对象保存起来，以后再取出来用。

此程序用一个 `HttpSession`在 Java 考试中跟踪用户的响应。考试分 20 种类别，选择类别后，就可以回答该主题下所有选择题了。第一个问题如图 18-4 所示。

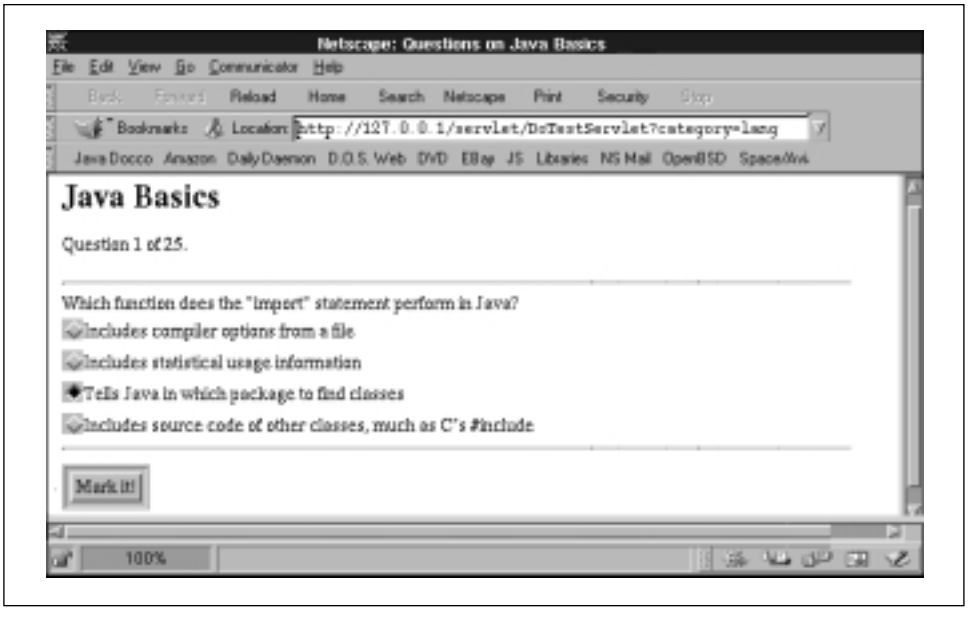

图 18-4　考试 Servlet 的开始页面

在回答了几个问题后，如图 18-5 所示。

考试结束后，可以看到答对了的题目总数。

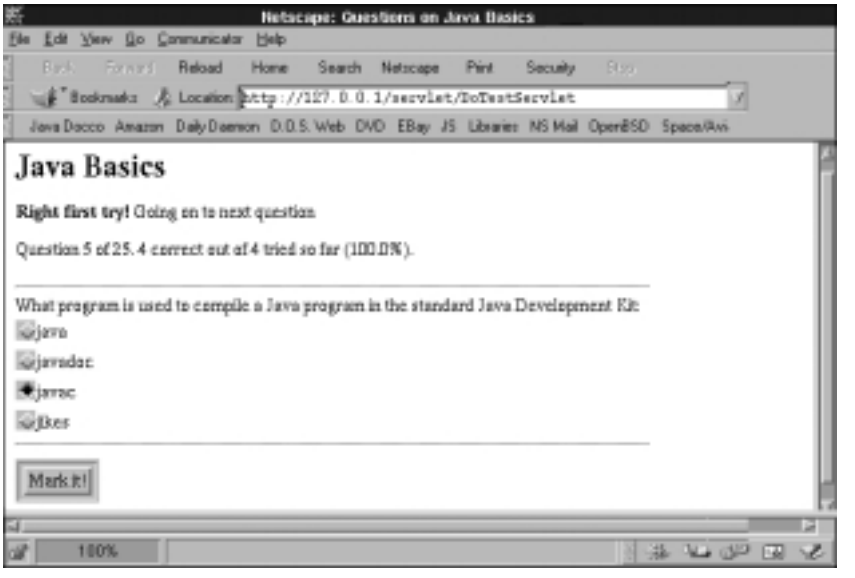

图 18-5　考试 Servlet 在几个问题之后的页面

Exam 对象（含有所有问题和答案以及答对题数的一个对象）是用一个 XamData-Accessor载入的(这两个类的代码没有在书中出现),而且Exam是保存在Progress对象中的。Progress 是一个 Servlet 中的内部类，是用来监控考试进程的一个数据结构。当改变主题时，Progress 将销毁并创建一个新的。代码如例 18-7 所示，主要是检测和跟踪答案,并生成HTML显示已答过了的题目的结果,以及当前的题目。

例 18-7：DoTestServlet.java

```
/** 管理网上考试的 Java Servlet
 *  将考试和状态会话对象保存，以避免重载入
 *  同时也要在会话中保存!
 */
public class DoTestServlet extends HttpServlet {

    /** 考试存放的路径 */
    protected final static String DIRECTORY =
         "/home/ian/webs/daroadweb/quizzes-";
    /** 背景色 */
    protected final static String BGCOLOR = "white";

    /** 跟踪学生进程的内部类 */
    class Progress {
        Exam exam;   // 正在进行的考试
```

```
    boolean done;// 已完成的考试
    String category;// 考试的名字
    int  curQuest;// 题号，从 0 开始
    int  correct;// 答对的题号
}

/** 为每个请求提供的服务 */
public void service(HttpServletRequest request,
    HttpServletResponse response) throws IOException, ServletException {

    PrintWriter out = response.getWriter();
    HttpSession session;
    Progress progress;
    String reqCategory;

    // 将响应设置为 HTML，输出 HTML 标签
    response.setContentType("text/html");
    out.println("<HTML>");

    // 寻找所请求的类别
    reqCategory = request.getParameter("category");
    reqSubject  = request.getParameter("subject"); // Unix 或 Java

    // 请求用户的会话，如为新则创建一个
    session = request.getSession(true);
    if (session.isNew()) {
        // out.println("<B>NEW SESSION</B>");
        progress = new Progress();
        progress.category = reqCategory;
        session.putValue("progress", progress);
    } else {
        progress = (Progress) session.getValue("progress");
    }
    if (reqCategory != null && progress.category != null &&
        !reqCategory.equals(progress.category)) {

        // 改变类别
        // out.println("<B>NEW PROGRESS CUZ " +
        //     reqCategory + " != " +progress.category + "</B>");
        progress = new Progress();
        progress.category = reqCategory;
        session.putValue("progress", progress);
    }
    if (progress.exam == null) {
        XamDataAccessor ls = new XamDataAccessor();
        try {
            progress.exam = ls.load(DIRECTORY + subject + "/" +
                progress.category + ".xam");
        } catch (IOException ex) {
            eHandler(out, ex, "We had some problems loading that exam!");
        } catch (NullPointerException ex) {
            eHandler(out, ex, "Hmmm, that exam file seems to be corrupt!");
        }
```

```
}

// 考试类别定下来了，获取标题
out.print("<TITLE>Questions on ");
out.print(progress.exam.getCourseTitle()); out.println("</TITLE>");
out.print("<BODY BGCOLOR=\""); out.print(BGCOLOR); out.println("\">");
out.print("<H1>");
out.print(progress.exam.getCourseTitle());
out.println("</H1>");

// 防止重载前一页
if (progress.done) {
    out.println("<HR><a href=\"/quizzes/\">Another Quiz?</a>");
    out.flush();
    return;
}

// 是回答问题还是进行标记？
out.println("<P>");
String answer =request.getParameter("answer");
int theirAnswer = -1;
if (answer != null) {
    // 标记
    Q q = progress.exam.getQuestion(progress.curQuest);
    theirAnswer = Integer.parseInt(answer);
    if (theirAnswer == q.getAns()) {
        // 答对了，好啊！
        if (!q.tried) {
            out.println("<P><B>Right first try!</B>");
            progress.correct++;
        } else
            out.println("<P><B>Right. Knew you'd get it.</B>");
        q.tried = true;// 答过了

        if (++progress.curQuest >= progress.exam.getNumQuestions()) {
            out.print("<P>END OF EXAM.");
            if (progress.correct == progress.curQuest) {
                out.println("<P><B>Awesome!</B> You got 100% right.");
            } else {
                out.print("You got ");
                out.print(progress.correct);
                out.print(" correct out of ");
                out.print(progress.curQuest);
                out.println(".");
            }
            out.println("<HR><a href=\"/quizzes/\">Another Quiz?</a>");

            // 用户重试时验证
            progress.done = true;

            // 返回，这样不会输出下一道题！
            return;
```

```
                } else {
                    out.print("Going on to next question");
                    theirAnswer = -1;
                }
            } else {
                out.print("<B>Wrong answer</B>. Please try again.");
                q.tried = true;
            }
        }

        // 继续吗？
        out.print("<P>Question ");
        out.print(progress.curQuest+1);
        out.print(" of ");
        out.print(progress.exam.getNumQuestions());
        out.print(". ");
        if (progress.curQuest >= 2) {
            out.print(progress.correct);
            out.print(" correct out of ");
            out.print(progress.curQuest);
            out.print(" tried so far (");
            double pct = 100.0 * progress.correct  / progress.curQuest;
            out.print((int) pct);
            out.println("%).");
        }

        // 现在为下一个(或同一)问题生成一个表单
        out.print("<FORM ACTION=/servlet/DoTestServlet METHOD=POST>");
        out.print("<INPUT TYPE=hidden NAME=category VALUE=");
            out.print(progress.category); out.println(">");
        out.println("<HR>");

        Q q = progress.exam.getQuestion(progress.curQuest);
        out.println(q.getQText());

        for (int j=0; j<q.getNumAnswers(); j++) {
                out.print("<BR><INPUT TYPE=radio NAME=answer VALUE=\"");
                out.print(j);
                out.print("\"");
                if (j==theirAnswer)
                    out.print(" CHECKED");
                out.print(">");
                out.print(q.getAnsText(j));
                out.println("</INPUT>");
            }
        out.println("<HR>");

        out.println("<INPUT TYPE=SUBMIT VALUE=\"Mark it!\"");
        out.println("</FORM>");
        out.println("</HTML>");
        out.close();
    }
    void eHandler(PrintWriter out, Exception ex, String msg) {
```

```
            out.println("<H1>Error!</H1>");
            out.print("<B>");
            out.print(msg);
            out.println("</B>");
            out.println("<pre>");
            ex.printStackTrace(out);
            out.flush();
            out.close();
        }
    }
```

使用了 HttpSession 的 Servlet 的调试技巧

许多对象（如上例中的`Exam`和`Progress`）都会在服务器中保存，直至会话结束。如果改变了这种类，使它与旧版本不再兼容，将出现让人难于捉摸的“类强制转换错误”。这时，应该关闭浏览器，再创建一个新的会话对象。另外可以参见实例9.17，那里讲述了避免`ClassCastException`错误的另一种方式。

18.5. 从 Servlet 中生成 PDF

问题

要用 Acrobat PDF 这样的格式创建一个打印美观的文档。

解决之道

用`response.setContentType("application/pdf")`,以及可以生成PDF的第三方 Java API。

讨论

PDF（Portable Document Format, 可移植文档格式）是 Adobe 公司发明的一种文件格式。它使我们可以完全控制文档的外观，比 HTML、XML 甚至 Java 的打印程序（第十二章）。都要强大得多。Adobe Acrobat 是读写 PDF 的程序集。Adobe 公司本身并没有发布从头创建 PDF 的 Java API, 但它发布了这种文件格式的规范（Adobe PDF Specification），并且明确宣布了编写生成和处理 PDF 文件的软件无需专门授

权。PDF用Java这样的面向对象语言处理极为般配，因为它本身就是一种基于对象的文本格式。因此，PDF 的 Java API 很多，商业的和自由的都有。

- Sitraka/KL Group（*http://www.klg.com*）出品了 PDF API、图表制作及其他工具，还有 JProbe，领先的调校工具。
- StyleWriterEE（*http://www.InetSoftCorp.com*）。
- PDFLib GmbH（*http://www.pdflib.com/pdflib/*）出品的 PDFLib。PDFLib 大部分用 C 写成，带有一个 Java 封装。还有其他语言的绑定。源码随附，非商业用途免费，商业用途需缴纳少量费用。
- ReportLab（*http://www.reportlab.org*）不是针对 Java 的，完全用 Python 写成，可能在 JPython（参见实例 26.3）中使用。请关注未来版本。
- 因为我无法做出选择，因此自己写了一个，这就是 SPDF。
- 访问 *http://www.pdfzone.com* 在 Toolbox 中寻找其他产品。

与 Perl 一样，SPDF 有好几种全称。褒义的 Perl 全称，是 Practical Extraction and Report Language,贬义的呢是Purely Eclectic Rubbish Lister。真正的狂热者（geek）会喜欢这种怪异的趣味。我的SPDF既可以代表Simple PDF API, 也可以说是Stupid PDF API。恐怕更多地称为后者更合适。例 18-8 是一个简单的 Servlet，从图 18-6 的 HTML 表单中获取用户名，然后用客户信息及惟一的序列号（为了方便这里用 `Date` 对象代替了）生成一张定做的购物券，序列号可以防止一张购物券多次使用。我估计如果网上商店和大型零售折扣商店联合，有许多机会要用到这种购物券。可我因为忙于埋头写这本书，没有时间去抓住这巨大的商机，因此我准备把 SPDF 的源代码公开了。如果你因此而致富，可别忘了寄给我一份！

我没有把 SPDF 的代码放在书中，因为它目前的版本太粗陋了：没有字体支持，没有图形，只支持单页文档。但它会发布的，如果感兴趣请访问*http ://www.darwinsys.com/freeware/spdf.html*。在你真正要使用一个 PDF API 时，可以把SPDF 看作备用 PDF API。

点击“ Get Yours ”按钮，Servlet 将运行，生成一个 PDF 文件，并把它发回给浏览器。因为我没有安装 Acrobat，我的 Unix 版本的 Netscape 会将它存在硬盘中。文件名 *MyCoupon.pdf* 是 response 对象中添加的 `Content-disposition` 首部所提供的。如图 18-7。

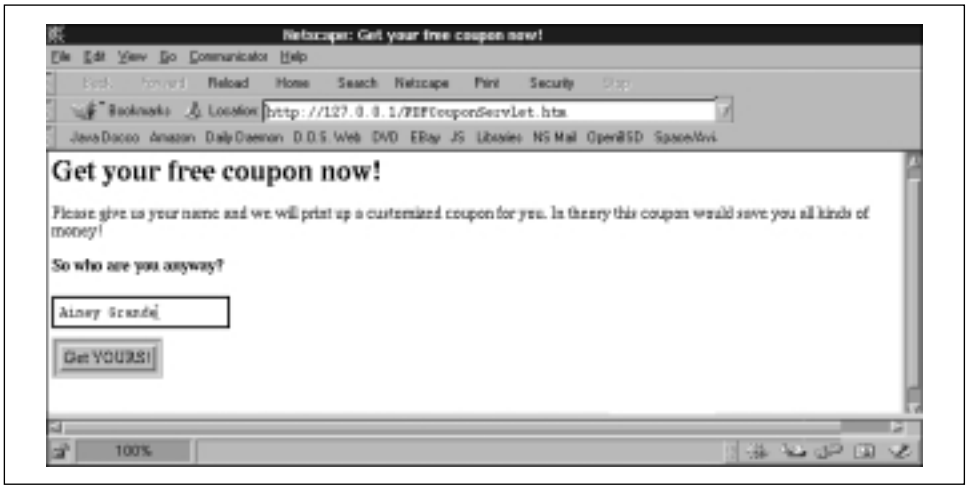

图 18-6 PDF 购物券 Servlet

图 18-7 PDF 购物券的保存对话框

我的MS-Windows系统中Netscape已安装了Acrobat,所以将作为插件运行并显示生成的购物券，如图 18-8。

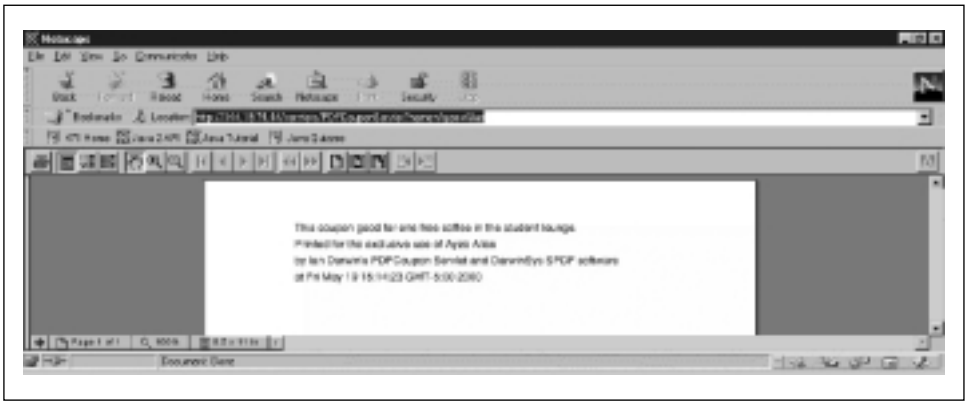

图 18-8　Acrobat Reader 中显示的 PDF 购物券

基本的 SPDF API 用一个 PDF 对象代表 PDF 文件。PDF 对象有进行设置、添加页。（page对象也有添加文本字符串、moveTO操作及其他功能的方法），以及写文件的方法。例 18-8 是响应图 18-6 中购物券请求的 Servlet。

例 18-8：PDFCouponServlet.java

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import com.darwinsys.spdf.*;

/** 简单的基于 PDF 的购物类输出 Servlet
 */
public class PDFCouponServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response) throws IOException {

        PrintWriter out = response.getWriter();
        response.setContentType("application/pdf");

        // 浏览器内嵌显示，如不能
        // 存为指定名字的文件
        response.setHeader("Content-disposition",
            "inline; filename=\"MyCoupon.pdf\"");
        PDF p = new PDF(out);
        Page p1 = new Page(p);
        p1.add(new MoveTo(p, 100, 600));
        p1.add(new Text(p,
            "This coupon good for one free coffee in the student lounge."));
        String name = request.getParameter("name");
        if (name == null)
            name = "unknown user";
```

```
            p1.add(new Text(p,
                "Printed for the exclusive use of " + name));
            p1.add(new Text(p,
                "by Ian Darwin's PDFCoupon Servlet and DarwinSys SPDF software"));
            p1.add(new Text(p, "at " + new Date().toString()));
            p.add(p1);
            p.setAuthor("Ian F. Darwin");

            // 写 PDF 文件页
            p.writePDF();
        }
    }
```

大多数Java PDF API基本上是很相似的。例18-9是用PDFLib重写的Terms Servlet，可以用 PDFLib 生成更美观的 PDF 格式的字典。

例 18-9：TermsServletPDF.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import com.pdflib.*;

/** 以更美观的 PDF 格式输出字典
 * 这个版本使用了 "PDFlib", 来自 PDFLib.GmbH (www.pdflib.com)
 */
public class TermsServletPDF extends HttpServlet {
    /** 获取响应的对象 */
    PrintWriter out;

    /** 处理 get 请求 */
    public void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException {

        try {

            out = new PrintWriter(response.getOutputStream());

            int font;
            pdflib p = new pdflib();

            if (p.open_file("") == -1) {
                warning(response, "Couldn't create in-memory PDF file", null);
                return;
            }

            p.set_info("Title", "Dictionary Project");
            p.set_info("Author", "Ian F. Darwin, ian@darwinsys.com");
            p.set_info("Creator", "www.darwinsys.com/dictionary");
```

```
            p.begin_page(595, 842);

            font = p.findfont("Helvetica", "host", 0);

            p.setfont(font, 14);

            // 目前只使用一条术语
            Iterator e = new TermsAccessor("terms.txt").iterator();
            Term t = (Term)e.next();
            p.set_text_pos(50, 700);
            p.show("Term: ");
            p.continueText(t.term);
            p.set_text_pos(70, 666);
            p.show("Definition: ");
            p.continueText(t.definition);
            p.end_page();

            p.close();

            byte[] data = p.get_buffer();

            response.setContentType("application/pdf");
            response.getOutputStream().write(data);
        } catch (IOException e) {
            warning(response, "pdflib IO error:", e);
            return;
        } catch (Exception e) {
            warning(response, "pdflib error:", e);
            return;
        }
    }
```

例 18-10 中 Servlet 的末尾说明了一种方式，可以提供出现问题时用户友好的异常轨迹。`warning()`方法当仅传入 `null` 作为异常参数时，还可以输出不带轨迹的通用错误信息。

例 18-10：TermsServletPDF 错误处理

```
    /** 通用错误处理，必须在 "out" 使用之前调用 */
    protected void warning(HttpServletResponse response,
        String error, Exception e) {
        response.setContentType("text/html");
        try {
            PrintWriter out = response.getWriter();
        } catch (IOException exc) {
            // 呀，无法与用户正确通信
            System.err.println("EGAD! IO error " + exc +
                  " trying to tell user about " + error + " " + e);
            return;
        }
        out.println("<H1>Error</H1>");
```

```
            out.print("<P>Oh dear. You seem to have run across an error in ");
            out.print("our dictionary formatter. We apologize for the inconvenience");
            out.print("<P>Error message is ");
            out.println(error);

            if (e != null) {
                out.print("<P>Exception is: ");
                out.println(e.toString());
                out.print("Traceback is: ");
                out.print("<PRE>");
                e.printStackTrace(out);
                out.print("</PRE>");
            }
            System.out.print("DictionaryServletPDF: ");
            System.out.println(error);
            if (e != null) {
                System.out.println(e.toString());
            }
        }
    }
```

18.6. 当 HTML 遇上 Java:JSP

问题

需要在网页中使用 Java。

解决之道

使用 JSP 方法混用 HTML 的 Java。

讨论

JSP（Java Server Pages，Java 服务器页面）与 Microsoft 的 ASP 和自由软件 PHP 都有很多通用的语法。它们都可以用来将 HTML 和程序代码结合。代码在服务器端执行，而把 HTML 再加上代码结果一起输出到客户端。由于 Java 的移植性极好，加之 JSP 可以访问整个 Java API，JSP 可以说是自 cookie 以来最激动人心的 Web 技术。例 18-11 是用 JSP 实现的随机数代码。

例 18-11：fireints.jsp

```
<HTML>
```

```
<HEAD>
<TITLE>Your Personal Random Numbers</TITLE>
<H1>Your Personal Random Numbers</H1>
<P>Here are your personal random numbers,
carefully selected by a
<A HREF=\"http://java.sun.com\">Java</A> program.
<OL>
    <%
    java.util.Random r = new java.util.Random();
    for (int i=0; i<5; i++) {
       out.print("<LI>");
       out.println(r.nextInt());
    }
    %>
</OL>
<HR></HR>
<A HREF=\"index.html\">Back to main Page</A>
```

请注意，与实例 18-1 中的 Servlet 版本相比，这个例子是多么浓缩。你可能已经猜到，JSP 实际上将编译成 Servlet，因此大部分我们所知的 Servlet 知识也适用于 JSP。让我们再看一个例子，它将生一个 HTML 表单，并在被激活时自我回调，其中还包含一个显示当月的 HTML 表格。图 18-9 和例 18-12 是实例 6.11 中 `CalendarPage` 程序的 JSP 版本。

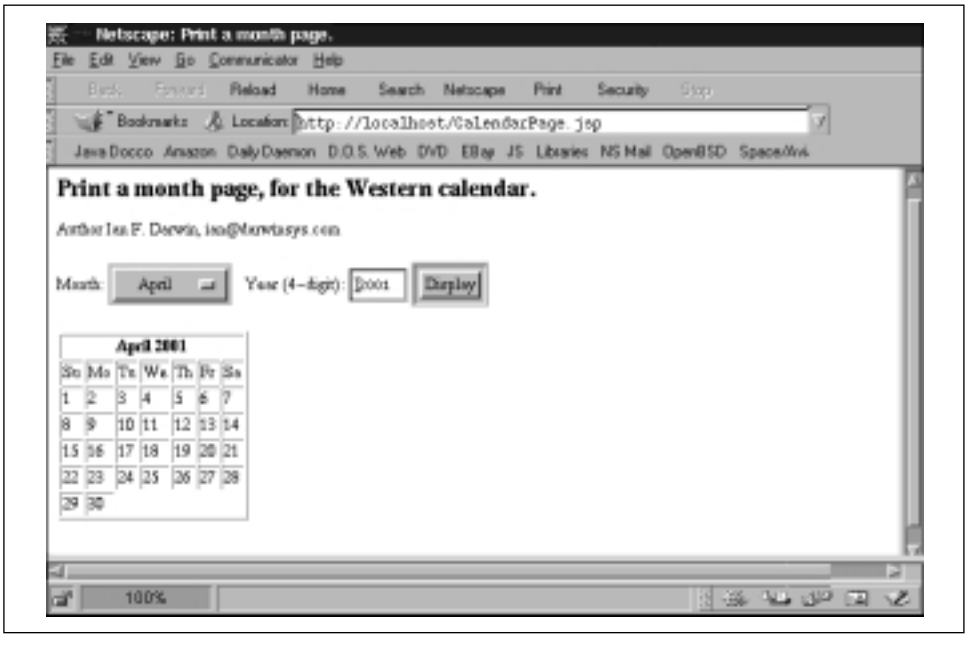

图 18-9　执行中的 CalenderPage.jsp

例 18-12：CalendarPage.jsp

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

<head>
    <title>Print a month page.</title>
    <meta name="version"
</head>
<body bgcolor="white">
<h1>Print a month page, for the Western calendar.</h1>
<P>Author Ian F. Darwin, ian@darwinsys.com

<%  // 首先从表单获取月份和年份
    boolean yyok = false;// -1 是非法的年份, 用布尔值
    int yy = 0, mm = 0;
    String yyString = request.getParameter("year");
    if (yyString != null && yyString.length() > 0) {
        try {
            yy = Integer.parseInt(yyString);
            yyok = true;
        } catch (NumberFormatException e) {
            out.println("Year " + yyString + " invalid");
        }
    }
    Calendar c = Calendar.getInstance();
    if (!yyok)
        yy = c.get(Calendar.YEAR);

    String mmString = request.getParameter("month");
    if (mmString == null) {
        mm = c.get(Calendar.MONTH);
    } else {
        for (int i=0; i<months.length; i++)
            if (months[i].equals(mmString)) {
                mm = i;
                break;
            }
    }
%>

<form method=post action="CalendarPage.jsp">
    Month: <select name=month>
    <% for (int i=0; i<months.length; i++) {
        if (i==mm)
            out.print("<option selected>");
        else
            out.print("<option>");
        out.print(months[i]);
        out.println("</option>");
    }
    %>
    </select>
    Year (4-digit):
```

```
        <input type="text" size="5" name="year"
            value="<%= yy %>"></input>
    <input type=submit value="Display">
</form>
<%!
    /** 每月的天数 */
    String[] months = {
        "January", "February", "March", "April",
        "May", "June", "July", "August",
        "September", "October", "November", "December"
    };

    /** 月份名 */
    int dom[] = {
            31, 28, 31, 30,    /* jan feb mar apr */
            31, 30, 31, 31,    /* may jun jul aug */
            30, 31, 30, 31     /* sep oct nov dec */
    };
%>

<%
    /** 本月开始应空白的天数 */
    int leadGap = 0;
%>
<table border=1>
<tr><th colspan=7><%= months[mm] %>  <%= yy %></tr>

<%      GregorianCalendar calendar = new GregorianCalendar(yy, mm, 1);%>

<tr><td>Su<td>Mo<td>Tu<td>We<td>Th<td>Fr<td>Sa</tr>

<%
        // 计算第一天前要空几个
        // getDay() 对 Sunday 返回 0, 刚好
        leadGap = calendar.get(Calendar.DAY_OF_WEEK)-1;

        int daysInMonth = dom[mm];
        if (calendar.isLeapYear(calendar.get(Calendar.YEAR)) && mm == 1)
            ++daysInMonth;

        out.print("<tr>");

        // 在每月第一天前留空白
        for (int i = 0; i < leadGap; i++) {
            out.print("<td> ");
        }
        // 填入本月的日期
        for (int i = 1; i <= daysInMonth; i++) {

            out.print("<td>");
            out.print(i);
            out.print("</td>");
```

```
                if ((leadGap + i) % 7 == 0) {// 如到行末转行
                    out.println("</tr>");
                    out.print("<tr>");
                }
            }
    %>
    </tr>
    </table>
```

另一个例子，例 18-13 将用 JSP 显示实例 18-1 中的术语和定义列表。

例 18-13：terms.jsp

```
<HTML>
<HEAD>
    <TITLE>Ian Darwin's Computer Terms and Acronyms</TITLE>
    <%@ page import="java.io.*" %>
</HEAD>
<BODY BGCOLOR=white>
<H1>Ian Darwin's Computer Terms and Acronyms</H1>
<TABLE BORDER=2>
<TR><TH>Term<TH>Meaning</TR>
    <%
    // 部分 Servlet , 生成几行, 如
    // <TR> <TD>JSP <TD>Java Server Pages, a neat tool for ...

    // 这种文件名不能做参数
    // 这样会使黑客读取系统中的文件!!
    // 实际程序中应从 Properties 文件中读取
    String TERMSFILE = "/var/www/htdocs/hs/terms.txt";

    TermsAccessor tax = new TermsAccessor(TERMSFILE);
    Iterator it = tax.iterator();
    while (it.hasNext()) {
        Term t = it.next();
        out.print("<TR><TD>");
        out.print(t.term);
        out.print("</TD><TD>");
        out.print(t.definition);
        out.println("</TD></TR>");
    }
    %>
</TABLE>
<HR></HR>
<A HREF="/servlet/TermsServletPDF">Printer-friendly (Acrobat PDF)
version</A>
<HR></HR>
<A HREF="mailto:compquest@darwinsys.com?subject=Question">Ask about
another term</A>
<HR></HR>
<A HREF="index.html">Back to HS</A> <A HREF="../">Back to DarwinSys</A>
<HR></HR>
```

18.7. JSP 的嵌入和控制转移

问题

要写一个页面合成的 JSP，包含其他页面或可将控制转给另一页面。

解决之道

使用 `<jsp:include>` 或 `<jsp:forward>`。

讨论

假设我们有一些公用 HTML 代码，要在每个页面上显示，比如导航条和或标题。当然可以将它复制到每个 HTML 和 JSP 文件中，但改起来就惨了，必须把所有文件都找一遍，再逐个更新。如果只用一份，在需要时嵌入进来，就会容易多了。大多数 Web 服务器都有这种机制(如服务器端嵌入，SSI)。但是，使用 JSP 有很多优点，比如可以在请求中添加对象，我们在实例 18.8 中探讨。

基本机制是很简单的，就是用 `<jsp:include>`，并用 page 属性命名要嵌入的页面。并以 `</jsp:include>`。为方便起见。可以在开始标签末尾放一个“ / ”，省略结束标签。这种语法很大程度是借鉴了 XML 的名字空间（参见第二十一章）。flush 属性也是必需的，它的值必须是 TRUE。这提醒我们，一旦用了嵌入，输出的内容都将写出。所以，就不用我们自己为发送 HTTP 首部而费心了，比如改变内容类型，或用 HTTP 重定向请求转移控制。因此完整的 JSP 嵌入如下：

```
<H2>News of the day</H2>
<jsp:include page="./news.jsp" flush="true" />
```

`jsp:forward`请求与`jsp:include`类似，但控制不能反向转移。属性`flush="true"`在一些 JSP 引擎（还包括本书付印时的 Tomcat）上是必需的，以提醒我们一旦用于嵌入，必须提交输出(在嵌入前，输出还可以都放在缓冲区中)。所以与刚才说的一样，不用再操心什么首部的生成了。包括`setContentType()`，`sendRedirect()`等等。

另一种替换的嵌入机制是 `<%@include file="filename"%>`。这种机制效率更好

（嵌入是 JSP 在编译时完成的），但仅限于文本文件（读取的是文件，而不是处理 HTTP URL。因此如果要嵌入一个 CGI 脚本，CGI 脚本的内容将出现在 JSP 输出，这一点儿用都没有）。而 `<jsp:include>` 可以嵌入任何类型的 URL（HTML，Servlet，JSP，CGI，甚至是 PHP 或 ASP）。

18.8. 使用 Servlet 的 JSP

问题

似乎 Servlet和JSP是互斥的，但事实上它们可以协作得很好。通过把控制从 Servlet 转移到 JSP，可以减少 JSP 中 Java 代码的数量。

解决之道

使用 MVC（Mode-View-Controller，模型－视图－控制器）设计模式（译注 1），用 `ServletDispatcher().forward()`。

讨论

MVC 是用来创建良好的用户交互程序的一种设计模式。其中 M（Model, 模型）是表示数据的对象或集合；V（View, 视图）是用户所能看到的内容，而 C（Controller, 控制器）则对用户请求做出响应。考虑一个幻灯片演示程序：有一个文本视图，一个幻灯片视图和一个排序视图。当改变任何视图中的数据时，所有其他视图也需要立即更新。这是因为 MVC 中，多个视图都指向同一个模型。MVC 为大多数设计良好的 GUI 程序提供了设计基础。

使用 MVC 模式，Servlet 可以是控制器，JSP 可以是视图。例如，Servlet 可以接收来自表单的原始请求，按请求查询数据库，构造与用查询匹配的对象集合，并将它转发给一个 JSP 去显示（Servlet 可以将找到的数据加入请求中）。一个很好的例子是搜索页面，它可以只有几个（甚至一个）表单参数，因此用一个带 JavaBean 的 JSP

译注 1： MVC 实际上是 Observer 设计模式的一种实例，最早在 Smalltalk 语言中出现，参见《Design Patterns》一书。

来接收结果就有些大材小用了，更好的设计是让一个Servlet获取表单参数，并联系搜索 API 或数据库。从中 Servlet 可以获取与查询匹配的页面列表。Servlet 还可以将列表封装到一个 `Vector` 或 `ArrayList` 中，并将它加入请求，转发给 JSP 去做格式化。

基本语法是：

```
ArrayList searchResultsList = // 从查询获取
RequestDispatcher disp;
disp = getServletContext().getRequestDispatcher("searchresults.jsp");
request.setAttribute("my.search.results", searchResultsList);
disp.forward(request, response);
```

这将使 Servlet 将搜索结果传给 JSP。JSP 可以用以下代码获取结果集：

```
ArrayList myList = (ArrayList) request.getAttribute("my.search.results");
```

然后可以用 `for` 循环输出搜索请求的内容。请注意，`getRequestDispatcher()`调用中的URL必须是对同一个Web服务器的调用，不能是另一端口或机器上的服务器。

18.9. 用 JavaBean 组件减少 JSP 中的 Java 代码量

问题

使用 JavaBean 组件减少 JSP 中 Java 编码量。

解决之道

使用 `<jsp:useBean>`。

讨论

JavaBeans（译注2）是 Java 的组件技术，与 MS-Windows 上的 COM 组件类似。实

译注 2： JavaBeans是Java组件技术的名字，而具体的组件称为JavaBean或Bean，请读者注意。

例 23.7 和 23.8 中有一个将 Java 类封装为 JavaBean 的方案。虽然 JavaBeans 最初是一种客户端的方便 GUI 创建的组件，但 JavaBeans 规范中对于其应用并没有仅限制在客户端或 GUI。事实上，JavaBean 组件用于 JSP 非常常见。而且这非常容易也很有用，让我们看看怎么使用。

最低要求下的JavaBean是有一个公共无参数构造方法并遵守set/get模式的对象。这意味着，get/set 型方法是一种常规。比如有一个类，其实例代表一个需要登录的网站上的用户帐号。例如，对于名字，如下方法：

```
public void setName(String name);
public String getName();
```

可以使其他类得以完全控制类中的“ name ”字段, 同时又保持了一定程度的封装性，也就是说，程序不必知道字段的实际名字（可以是 name，myName 或其他）（译注 3）。其他程序甚至可以用内省（introspection, 参见实例 25.2）获取多个 get/set 型方法。例 18-14 是完整的类文件。可以看到，它主要都是一些 get/set 型方法。

例 18-14：User.java，一个可重用的类，一个 JavaBean

```
/** 代表一个登录了的用户
 */
public class User {

    protected String name;
    protected String passwd;
    protected String fullName;
    protected String email;
    protected String city;
    protected String prov;
    protected String country;

    protected boolean editPrivs = false;
    protected boolean adminPrivs = false;

    /** 构造一个无数据的用户 —— 必须是一个无参数的
     * 用于 jsp:useBean 的构造方法
     */
    public User() {
    }

    /** 构造一个只有名字的用户 */
    public User(String n) {
```

译注 3： C#中为此目的专门引进了一个新概念property(性质), 这样类的用户可以像访问字段一样使用 get/set 方法。

```
   name = n;
}

/** 返回别名 */
public String getName() {
   return name;
}

public void setName(String nick) {
   name = nick;
}

// 密码不是公开的，没有getPassword

/** 验证密码是否匹配 */
public boolean checkPassword(String userInput) {
    return passwd.equals(userInput);
}

/** 设置密码 */
public void setPassword(String passwd) {
    this.passwd = passwd;
}

/** 获取Email */
public String getEmail() {
    return email;
}

/** 设置Email */
public void setEmail(String email) {
    this.email = email;
}

 // 省略了许多相似的与字符串有关的set/get方法

/** 获取管理权限 */
public boolean isAdminPrivileged() {
    return adminPrivs;
}

/** 设置管理权限 */
public void setAdminPrivileged(boolean adminPrivs) {
    this.adminPrivs = adminPrivs;
}

/** 返回一个字符串表示. */
public String toString() {
    return new StringBuffer("User[").append(name)
         .append(',').append(fullName).append(')').toString();
}
```

```
    /** 检查是否所有必需字段都已设置 */
    public boolean isComplete() {
        if (name == null || name.length()==0 ||
            email == null || email.length()==0 ||
            fullName == null || fullName.length()==0 )
              return false;
        return true;
    }
}
```

惟一与 set/get 无关的方法是 `toString()`和 `isCompcete()`(后者在 JavaBean 中所有必需字段都设置好时返回 true)。如果你已经猜到这是在验证 HTML 表单中的必需字段，好好犒劳一下自己吧。

我们只需写如下一行代码即可在基于 JSP 的网页中使用 JavaBean:

```
<jsp:useBean id="myUserBean" scope="request" class="User">
```

这就创建了一个名为`myUserBean`的类实例。但是，现在它是空的，字段均未设置。要填充字段，可以在脚本段（scriptlet）中直接引用 JavaBean,或者采用更方便的`<jsp:setProperty>`将 HTML 表单的值直接传给 JavaBean。这可以省却大量的编码工作。

如果所有的名字（如表单中 HTML 参数名）和 JavaBean 中的 `setName(String)`方法匹配了，整个 HTML 表单都可以用 `property="*"` 传给一个 JavaBean！

```
<jsp:setProperty name="myUserBean" property="*"/>
</jsp:useBean>
```

现在 JavaBean 已经生成了，可以通过调用其 `iscomplete()`方法检测它是否完整。如果是完整的，输出响应；如果不完整，将用户引导回去，再次填充必需的字段：

```
<% // 现在看看表单是否填好 ...
    if (!myUserBean.isComplete()) {
%>
        <TITLE>Welcome New User - Please fill in this form.</TITLE>
        <BODY BGCOLOR=White>
        <H1>Welcome New User - Please fill in this form.</H1>
            <FORM ACTION="name_of_this_page.jsp" METHOD=post>
            // 再次输出表单，供二次填充
            </FORM>
        <%
        } else {
        String nick = newUserBean.getName();
        String fullname = newUserBean.getFullName();
```

```
// etc...
        // 欢迎语
        out.println("Welcome " + fullname);
```

可以在实例 18.12 中看到这个 JSP 页面的完整版本。

参考

我们可以通过使用 Java 自定义标签（custom tag）更多地从 JSP 中提取 Java 代码，使其更像纯粹的 HTML 页面。自定义标签（也称自定义行为）是减少 JSP 中必需的 Java 代码量的一种新机制。其进一步的优势在于，它的语法上看起来很像从 XML 名字空间（参见实例 21.0）中生成的元素。这使其对于编辑 HTML 的软件和编辑 HTML 的人来说都非常舒服。自定义标签的劣势在于，编写起来比 Servle 与 JSP 的时间都多。但是我们并不是非要自己编写自定义标签。已经有很多优秀的自定义标签库了。一个来自 Tomcat（*http://jakarta apache.org*），另一个来自 JRun（*http://jrun.allaire.com*）。Sun 也在制订通用标签库的标准。JSP 标签都是编译好的类，好像是 Applet 或 Servlet，因此任何厂商的任何标签库都可以用于任何兼容的 JSP 引擎。在实例 18.12 中的 JabaDot 程序的源目录中有许多 JSP 自定义标签。

18.10. JSP 语法总结

问题

我们记不住所有 JSP 的语法。

解决之道

查表。

讨论

表 18-1 总结了 JSP 的语法。顾名思义，这里只有基本语法。更复杂的语法请到 *http://java.sun.com/products/jsp* 下载。

表 18-1 JSP 基本语法

项目	语法	示例
Scriptlet	`<% code;%>`	`<% mountain.setHeight(1000); %>`
Expression (to print)	`<%= expr %>`	`<%=mountain.getHeight() %>`
Declaration	`<%! decls; %>`	`<%! int height = 0; %>`
Include	`<jsp:include page="URL" flush=true />`	`<jsp:include page="./mountain-list.html"flush=true />`
Forward	`<jsp:forward page="url"/>`	`<jsp:forward page="./last-resort.html"/>`
Use bean	`<jsp:useBean .../>`	`<jsp:useBean class="x.Climb-Bean"id="myClimbBean" scope="page"/>`
Set property	`<jsp:setProperty... />`	`<jsp:setProperty name="myClimbBean"property="*" />`
Page directive	`<%@ page ... %>`	`<%@ page import="java.io.*" errorPage="catcher.jsp" %>`
Comment	`<!-- comment -->`	`<%!-- This comment appears in HTML -->`
Hidden comment	`<%-- comment --%>`	`<%-- This comment is local to JSP --%>`

18.11. 程序：Cookiecutter

`Cookiecutter` 是我写的一个小程序，可以用来显示、修改甚至删除 cookie。因为 Double Click这样的网络条幅广告跟踪企业可能会记录许多有关我们浏览习惯的信息，我们应该骗一骗他们。毕竟他们用我们的硬盘空间，每次点击都赚取利润，对我们却一点儿表示都没有（当然了，没有广告赞助商，也就不可能有现在这么多网站，图 18-10 中，我正在将一个 cookie 编辑成……不，应该是说是在把一个个人身份 cookie 更新为一个无效数字（许多的 9，位数也太多了）。往上几行，可以看到我把 `prefs.bgcolor` cookie 设为“ green ”了。

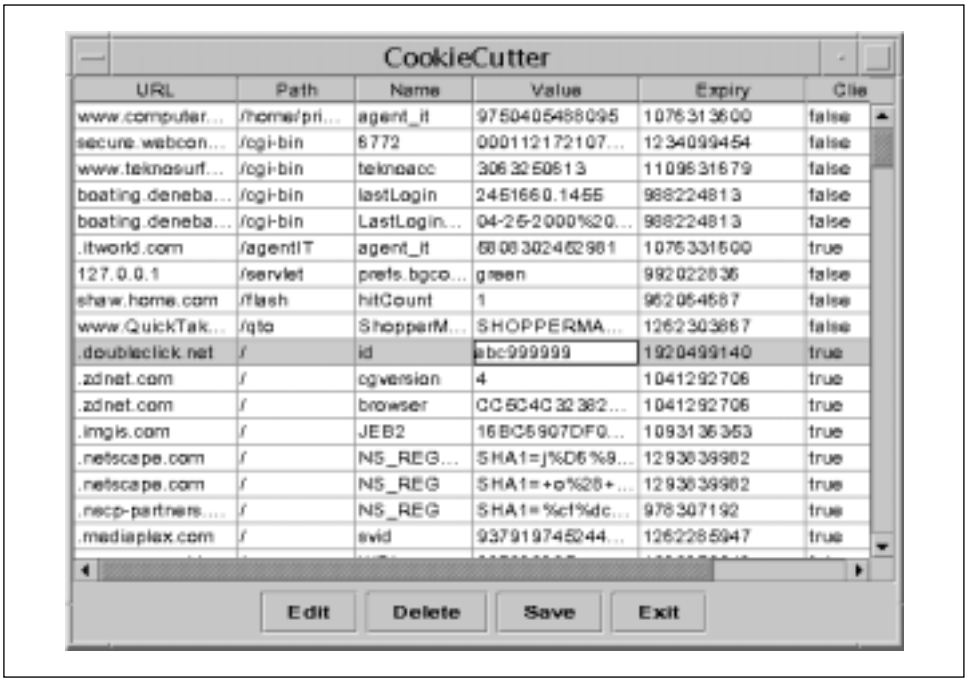

图 18-10 CookieCutter 的显示画面

这里我并不想列出 `CookieCutter` 的源代码，它其实并不是真的与 Web 技术有关（它只是一个客端程序），但在本书的源代码下载文件中可以找到。该程序假定 cookie 是以 Netscape 格式存储的；对于 Microsoft Explorer 格式，必须改变文件读取和文件写入代码。

18.12. 程序：JabaDot Web 新闻门户

这里可能要算本书中开发的最具雄心的一个程序了。它是完整的“新闻门户”网站（与*http://www.slashdot org*, *http://www.deadly.org*或*http://daily.dae monnews.org* 相似）的起点。但是正如你期望的，整个网站都是用 Java 开发的！可能我应该说成“用 Java 或由 Java 开发的”，因为有 JSP 机制（完全用 Java 写成），将 JSP 页面转成了这个网站上运行的 Java Servlet。网站如图 18-11 所示。

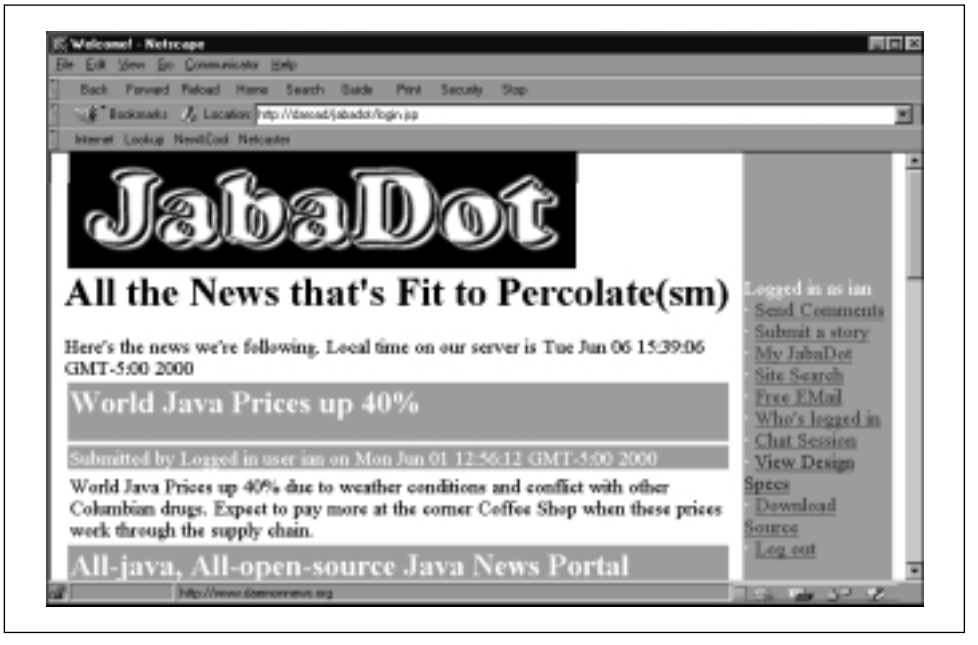

图 18-11 JabaDot 欢迎页面

与大多数门户网站一样，JabaDot 提供了许多无需登录的服务（如当前的新闻），当然还有无所不在的条幅广告，同时也提供需要登录的服务。图中我用自己的名字（ian）进行了登录，因此可以使用所有服务。支持此画面的页面是 *index.jsp*（例 18-15），其中包含 HTML 和 Java 代码的太杂烩。

例 18-15：index.jsp

```
<%@page errorPage="oops.jsp"%>
<HTML>
<TITLE>JabaDot - Java News For Ever(yone)</TITLE>
<P ALIGN=CENTER><jsp:include page="/servlet/AdRotator" flush="true"/></P>
<BODY BGCOLOR="#f0f0f0">
<%   HttpSession sess = request.getSession(true);
     User user = (User)sess.getValue("jabadot.login");
 %>
<TABLE>
<TD WIDTH=75% ALIGN="TOP">
     <!- 占据左边大部分页面 -->
     <IMG SRC="logo.gif" ALIGN="LEFT"></IMG>
     <BR CLEAR="ALL">
     <jsp:include page="./news.jsp" flush="true"/>
</TD>
<TD WIDTH=25% BGCOLOR="#00cc00" ALIGN="TOP">
```

```
        <!- 右边剩余页面 -->
        <FONT COLOR=WHITE NAME="Helvetica,Arial">
        <% if (user == null) { %>
        <FORM ACTION=login.jsp METHOD=POST>
             Name: <INPUT TYPE=TEXT SIZE=8 NAME=nick>
             <br>
             Password: <INPUT TYPE=PASSWORD SIZE=8 NAME=pass>
             <br>
             <INPUT TYPE=SUBMIT VALUE="Login" ALIGN=CENTER>
        </FORM>
        <jsp:include page="public_services.html" flush="true"/>
        <% } else { %>
        Logged in as <%= user.getName() %>
        <jsp:include page="./logged_in_services.html" flush="true"/>
        <li><a href="logout.jsp">Log out</a>
        <% } %>
        </FONT>
    </TD>
    </TABLE>
    </BODY>
    </HTML>
```

可以看到，代码的开始用“ page ”标签（%@page）指定错误处理页面（对错误页面只是输出堆栈轨迹，以及一句道歉）。然后，加入 AdRotator Servlet 的输出，程序还会随机地选择一个条幅广告，并以围绕一个 IMG 标签的一个 HTML 锚输出它。接下来获取 HttpSession 对象，并由此获取当前 User 对象，如果当前没有已登录用户，该对象为 null。User 类在讨论 JSP 中的 JavaBean 时已讲过了（实例 18.9）。它在大多数 JSP 中都只用作一个普通对象，但在 *newuser.jsp* 中是一个 JavaBean，此时用户已在“ Create an Account ”页面上输入了所有字段。

然后出现一个 HTML 表格，将剩下的页面分为两列。页面左边很宽，存放着新闻、标题提交者的名字，时间，（可选的）URL 和新闻正文。未来的版本将允许用户发送对新闻的评论。正如 Slashdot 所说，这是“ 社区建设 ”的重要组成部分，也是一种吸引网民不断浏览网站的策略，这样也可以向他们显示更多的广告。

右边导航部分的显示视用户是否登录而异。如果未登录，将以登录表单开始，然后以 HTML 锚形式列出公开提供的服务，而不能使用的服务用斜体表示。如果已登录，则列出完整的服务链接，最后是一个退出页面链接。

在登录之前，必须创建一个账号。这里的技巧是，要求用户给出一个有效 Email 地址，我们将其用于各种鉴别，而且可能以邮件形式发送每月的电子通讯。为了确保

用户提供的是有效Email地址，我们将按此地址发邮件告诉他下载密码的URL。图18-12是注册页面。表单是由 *newuser.jsp* 处理的。

图18-12 运行中newuser.jsp

例18-16是 *newuser.jsp* 的源代码。前面提到过，它所获取的`User`对象是一个JavaBean（见实例18-9）。

例18-16：newuser.jsp

```
<%@page errorPage="oops.jsp" import="jabadot.*, java.io.*" %>
<%! java.util.Random r = new java.util.Random(); %>
<jsp:useBean id="newUserBean" scope="request" class="jabadot.User">
    <jsp:setProperty name="newUserBean" property="*"/>
</jsp:useBean>
<jsp:useBean id="mailBean" scope="request" class="jabadot.Mailer"/>
<html>
<%@include file="header.html" %>
<%
    User user = (User)session.getAttribute("jabadot.login");
    if (user != null) {
 %>
<TITLE>You're already logged on!</TITLE>
```

```
<H1>You are logged on!</H1>
<P>Please <A href="logout.jsp">log out</A> before
trying to create a new account.  Thank you!
<%  return;
    }
 %>
<% //现在看看表单是否填好了 ...
    if (!newUserBean.isComplete()) {
        // out.println("<!-- in new -->");
 %>
        <TITLE>Welcome New User - Please fill in this form.</TITLE>
        <BODY BGCOLOR=White>
        <H1>Welcome New User - Please fill in this form.</H1>
        <TABLE>
        <TD>
        <FORM ACTION="newuser.jsp" METHOD=post>
            <table><!-内部表格, 使字段对齐 -->
            <tr><td>Nickname:</td>
                <td><INPUT TYPE=TEXT SIZE=10 NAME="name"> (required)</td></tr>
            <tr><td>Full name:</td>
                <td><INPUT TYPE=TEXT SIZE=10 NAME="fullName"> (required)</td></tr>
            <tr><td>E-mail:</td>
                <td><INPUT TYPE=TEXT SIZE=10 NAME="email"> (required)</td></tr>
            <tr><td>City:</td>
                <td><INPUT TYPE=TEXT SIZE=10 NAME="city"></td></tr>
            <tr><td>Province/State:</td>
                <td><INPUT TYPE=TEXT SIZE=10 NAME="prov"></td></tr>
            <tr><td>Country</td>
                <td><select name="location">
                <jsp:include page="country_select.html" flush="true"/>
                </select>
                </td></tr>
            <tr><td colspan=2 align="center">
                <INPUT TYPE=SUBMIT VALUE="Create My JabaDot!"></tr>
            </table>
       </FORM>
       <TD>
       <P>If you've done one of these before, you may be wondering where
       the "Password" field is. It's not there. Believing somewhat in
       security, we'll make up a fairly good password for you.
       We won't email it to you, but will email to you the location
       from which you can learn it, so watch your email closely
       after you complete the form. Thank you!
       </TABLE>
<%     return;
       }

       // out.println("<!-- in get -->");
       String nick = newUserBean.getName();
       if (UserDB.getInstance().getUser(nick) != null) {
 %>
            <P>It seems that that user name is already in use!
```

```
            Please go back and pick another name.
            <% return;
            } %>
<%
        String fullname = newUserBean.getFullName();
        String email = newUserBean.getEmail();
 %>
        <!-输出欢迎语 -->
        Welcome <%= fullname %>.
        We will mail you (at <%= email %>) with a URL
        from which you can download your initial password.

        <jsp:setProperty name="newUserBean"
            property="editPrivileged" value="false"/>
        <jsp:setProperty name="newUserBean"
            property="adminPrivileged" value="false"/>
<%
        // 生成随机密码, 保存在User中
        String newPass = Password.getNext().toString();
        newUserBean.setPassword(newPass);

        //将User保存到持久性数据库
        UserDB.getInstance().addUser(newUserBean);

        // 创建临时的HTML文件, 包含完整名字
        // 以及新的密码并可将URL寄给用户
        // 这可以确保用户提供的是有效地址
        // 千万不要同时显示网名和密码!
        String tempDir = JDConstants.getProperty("jabadot.tmp_links_dir");
        File tempLink = File.createTempFile(
            r.nextInt()+"$PW", ".html", new File(tempDir));
        PrintWriter pw = new PrintWriter(new FileWriter(tempLink));
        pw.print("<HTML><BODY>");
        pw.print("Greetings ");
        pw.print(newUserBean.getFullName());
        pw.print(". Your new password for accessing JabaDot is <B>");
        pw.print(newPass);
        pw.print("</B>. Please remember this, or better yet, ");
        pw.print("<a href=\"/jabadot/index.jsp\">");
        pw.print("login</a> now!");
        pw.print("You may want to visit \"My Jabadot\"");
        pw.print("and change this password after you log in.");
        pw.println("</HTML>");
        pw.close();

        // 现在把URL发送给用户
        mailBean.setFrom(JDConstants.getProperty("jabadot.mail_from"));
        mailBean.setSubject("Welcome to JabaDot!");
        mailBean.addTo(email);
        mailBean.setServer(JDConstants.getProperty("jabadot.mail.server.smtp"));

        // 获取URL, 删去 "newuser.jsp", 添加 "/tmp/"+tmpname
```

```
        StringBuffer getPW_URL = HttpUtils.getRequestURL(request);
        int end = getPW_URL.length();
        int start = end - "newuser.jsp".length();
        getPW_URL.delete(start,end).append("tmp/").append(tempLink.getName());
        mailBean.setBody("To receive your JabaDot password,\n" +
            "please visit the URL " + getPW_URL);

        //发送邮件
        mailBean.doSend();

        // 这里不要使用 sess.setAttribute()
        // 因为用户还没有验证密码!
%>
```

创建账户并读过含有密码的链接后，返回网站并正常登录。登录表单是由 *login.jsp* 处理的，如例 18-17 所示。

例 18-17：login.jsp

```
<%@page errorPage="oops.jsp" import="jabadot.*" %>
<HTML>
<%
    User user = (User)session.getAttribute("jabadot.login");
    if (user != null) {
        session.setAttribute("jabadot.message",
           "<H1>You're already logged on!</H1>"+
           "(as user " + user.getName() + "). Please" +
           "<a href=\"logout.jsp\">" +
           "logout</a> if you wish to log in as a different user.");
        response.sendRedirect("/jabadot/");
    }
    String nick = request.getParameter("nick");
    String pass = request.getParameter("pass");
    if (nick == null || nick.length() == 0 ||
        pass == null || pass.length() == 0) {
%>
        <!-这里必须使用 jsp include 而不是@ include
        ** 因为 Tomcat 看到第二个@ include 时会报错
        **不能只在开始嵌入，因为
        **成功时 JSP 最后要进行重导向
        -->
        <jsp:include page="./header.html" flush="true" />
        <TITLE>Missing name/password!</TITLE>
        <BODY BGCOLOR=WHITE>
        <H1>Missing name/password!</H1>
        <P>Please enter both a name and a password in the form.
<%      return;
    }

    User u = UserDB.getInstance().getUser(nick);
    if (u == null || !u.checkPassword(pass)) {
%>
```

```
            <jsp:include page="./header.html" flush="true" />
            <TITLE>Invalid name/password</TITLE>
            <BODY BGCOLOR=WHITE>
            <H1>Invalid name/password</H1>
            <P>We could not find that name and password combination.
            Please try again if you have an account, else go create one.
<%          return;
        }

        // 太好了，终于登录成功

        session.setAttribute("jabadot.login", u); // 登录标志
        //session.setAttribute("jabadot.ads", new AdServlet());
        session.setAttribute("jabadot.message",
            "<H1>Welcome back, " + u.getFullName() + "</H1>");

        // 对于非管理员登录，超时为 3 小时
        if (!u.isAdminPrivileged()) {
            session.setMaxInactiveInterval(3600*3);
        }

        // 重导向回首页，用户可以在 URL 中看到此页
        response.sendRedirect("/jabadot/");
%>
```

确认用户还未登录时，此页面从 HTML 表单获取用户名和密码，检测是否都存在，在密码数据库中查找名字，如果有则验证密码。如果有错，将报告（这是一种安全策略，可以避免给怀有恶意的用户更多信息，注2）。如果已登录，将代表用户的`User`对象放入`HttpSession`，设置欢迎语，再把控制权通过重定向转给主页面。

无论是否登录，都可以通过*submit.jsp*页面发一条通用的评论给系统管理员。这将生成一个 HTML 表单，如图 18-13 所示。

当按下“Submit Article”按钮时，此表单由例 18-18 中的*comments.jsp*处理。

例 18-18：comment.jsp

```
<%@page errorPage="oops.jsp" %>
<%@page import="jabadot.*, javax.mail.*" %>
<jsp:useBean id="mailBean" scope="request" class="jabadot.Mailer">
    <jsp:setProperty name="mailBean" property="*"/>
</jsp:useBean>
<%
    User user = (User)session.getAttribute("jabadot.login");
```

注 2： 这个古老的建议源自 Unix 的早期岁月，你可能会奇怪为什么至今还有许多网站并没有真正采纳之。

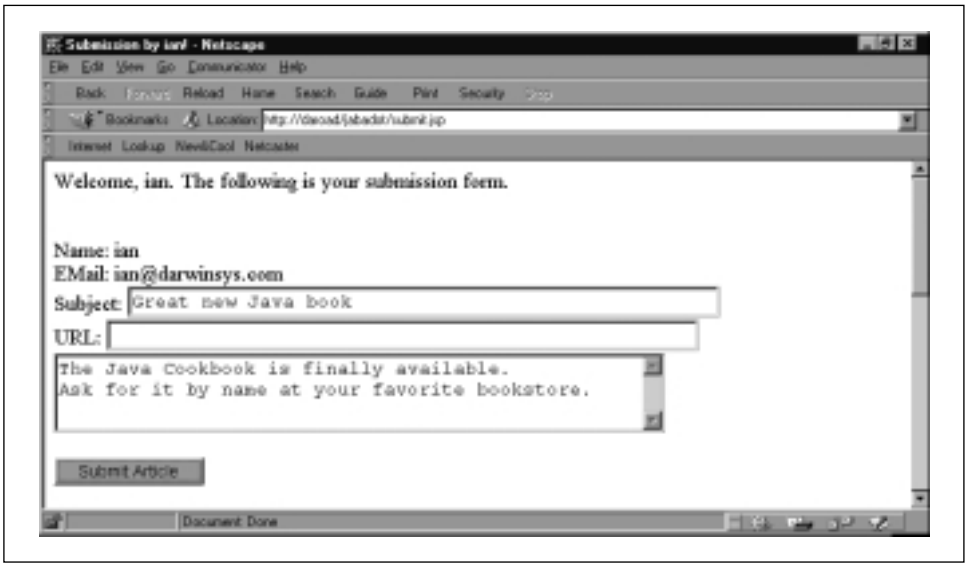

图 18-13 coments.jsp 的输入表单

```
        mailBean.setFrom(JDConstants.getProperty("jabadot.mail_from"));
        mailBean.setSubject("Comment from jabadot site");
        mailBean.addTo(JDConstants.getProperty("jabadot.mail_comments_to"));
       mailBean.setServer(JDConstants.getProperty("jabadot.mail.server.smtp"));

        String message = request.getParameter("message");
        if (message != null)
            mailBean.setBody(message);

        //看看是否已经填好了表单 ...
        if (mailBean.isComplete()) {
            try {
                mailBean.doSend();

                //将感谢语附上并发给索引页面
                session.setAttribute("jabadot.message",
                    "<H1>Mail sent</H1><p>Your commentary has been sent to our chief" +
                    " pickle.<b>Thank you.</b></p>");
                response.sendRedirect("/jabadot/");
                // 未从 sendRedirect 返回
            } catch (MessagingException ex) {
                throw new IllegalArgumentException(ex.toString());
            }
        }
        // 否则，mailBean 不完整，重显示表单
    %>
        <%@include file="header.html" %>
```

```
     <P ALIGN=CENTER><jsp:include page="/servlet/AdServlet" flush="true"/></P>
     <TITLE>Send Comments</TITLE>
     <BODY BGCOLOR="white">

     <DIV ALIGN="CENTER">
      <BLOCKQUOTE>

      <FORM METHOD="POST" ACTION="comments.jsp">

     <P>Please send us your feedback on this site.
<%   if (user != null) { %>
         <P>Since you are logged in, you can use
         <A href="mailto:<%=JDConstants.getProperty(
              // 这个不带引号的subject= 还可以工作，但感觉却不太好 :-)
              "jabadot.mail_comments_to")%>?subject=Comments about JabaDot">
         this <I>mailto</I> link</A>.</P>
<%   } %>
     <P>Name: <INPUT TYPE="TEXT" NAME="name" SIZE="15"
     VALUE="<%= user==null?"":user.getFullName()%>"> 
         Email:<INPUT TYPE="TEXT" NAME="from" SIZE="15"
         VALUE="<%= user==null?"":user.getEmail()%>"></P>
     <P>City:     <INPUT TYPE="TEXT" NAME="City" SIZE="15"
     VALUE="<%= user==null?"":user.getCity() %>"> 
         State: <INPUT TYPE="text" NAME="State" SIZE="15"
         VALUE="<%= user==null?"":user.getProv() %>"></P>
     <P>Country: <INPUT TYPE="text" NAME="country" size="15"
     VALUE="<%= user==null?"": user.getCountry() %>"></P>
     <TEXTAREA NAME="message" ROWS="8" COLS="50" WRAP="physical">
<%= message %>
     </TEXTAREA>
     <P><INPUT TYPE="submit" VALUE="Send Comments"></P>
      </FORM>

     </BLOCKQUOTE>
     </DIV>
     </BODY>
```

我非常喜欢在用户登录时显示mailto:URL的代码。SPAM发送器（见第十九章）因为只是为了找到mailto:URL而自动下载整个网站而臭名昭著。而这是防止鼠害进入的好办法。因为一般情况下用不着为了一个服务而登录，并提供真实的邮件地址，只是为了获得一个mailto:URL。更有趣的是，为每个用户生成惟一邮件地址。这样受到垃圾邮件攻击时，就知道是谁干的了。

参考

限于篇幅，关于Servlet和JSP只能讲这么多了。这些技术很好地将代码和功能分离

开来。JSP 主要与获取输入和显示结果有关。JSP 可以转成 Servlet，也可以嵌入或转向其他的本地 Web 资源，如一个音频文件。Servlet 和 JSP 是 J2EE 的主要组成部分，而且是一种日益重要的 Web 服务器技术。

关于 JSP 有一种相反的观点（即 JSP 是对错误问题的一种错误解决方案），可以访问 *http://www.servlets.com*。有关 Servlet 和 JSP 的更多信息，请参考 O'Reilly 出版的《Java Servlet Programming》和《Java Server Pages》（译注 4）。

译注 4： 中文版《Java Servlet 编程》和《JSP 设计》均已由中国电力出版社出版。