教　学　活　动　首　页

基　本　内　容
第 7 章 Java Servlet
教学目的与要求：通过本章的学习让学生了解如何用 servlet 读写文件，用 servlet 访问数据库；理解 servlet 工作原理，servlet 共享变量的使用；掌握编译和安装 servlet，通过 JSP 页面调用 servlet，HttpServlet 类，掌握会话管理。
教学内容： 7.1 servlet 工作原理 7.2 编译和安装 servlet 7.3 通过 JSP 页面调用 servlet 7.4 servlet 共享变量 7.5 HttpServlet 类 7.6 用 servlet 读写文件 7.7 用 servlet 访问数据库 7.8 会话管理
教学基本要求： 了解：用 servlet 读写文件，用 servlet 访问数据库 理解：servlet 工作原理，servlet 共享变量 掌握：编译和安装 servlet，通过 JSP 页面调用 servlet，HttpServlet 类，会话管理
教学重点教学难点： servlet 工作原理，编译和安装 servlet，通过 JSP 页面调用 servlet，HttpServlet 类，会话管理
教学方法： **教学手段：**多媒体教学和计算机程序演示
教学小结：　（见教学进程）
作业与思考：见课后习题
课后记载：

教 学 进 程

第 7 章 Java Servlet

我们已经知道，SUN 公司以 Java Servlet 为基础，推出了 Java Server Page。JSP 提供了 Java Servlet 的几乎所有好处，当一个客户请求一个 JSP 页面时，JSP 引擎根据 JSP 页面生成一个 Java 文件，即一个 servlet。这一章，将对 servlet 做一个较详细的介绍，这不仅对于深刻理解 JSP 有一定的帮助，而且通过学习 servlet，还能使我们选择使用 JSP+javabeans+servlet 的模式来开发我们的 Web 应用程序。

我们已经知道，用 JSP 支持 JavaBeans 这一特点，可以有效的管理页面的静态部分和页面的动态部分。另外，我们也可以在一个 JSP 页面中调用一个 servlet 完成动态数据的处理，而让 JSP 页面本身处理静态的信息。因此，开发一个 Web 应用有两种模式可以选择：

(1) JSP+javabeans

(2) JSP+javabeans+servlet

7.1 Servlet 工作原理

servlet 由支持 servlet 的服务器：servlet 引擎，负责管理运行。当多个客户请求一个 servlet 时，引擎为每个客户启动一个线程而不是启动一个进程，这些线程由 servlet 引擎服务器来管理，与传统的 CGI 为每个客户启动一个进程相比较，效率要高的多。

7.1.1 Servlet 的生命周期

学习过 Java 语言的人对 Java Applet(Java 小应用程序)都很熟悉，一个 Java Applet 是 java.applet.Applet 类的子类，该子类的对象由客户端的浏览器负责初始化和运行。servlet 的运行机制和 Applet 类似，只不过它运行在服务器端。一个 servlet 是 javax.servlet 包中 HttpServlet 类的子类，由支持 servlet 的服务器完成该子类的对象，即 servlet 的初始化。

Servlet 的生命周期主要有下列三个过程组成：

(1)初始化 servlet。servlet 第一次被请求加载时，服务器初始化这个 servlet，即创建一个 servlet 对象，这对象调用 init 方法完成必要的初始化工作。

(2)诞生的 servlet 对象再调用 service 方法响应客户的请求。

(3)当服务器关闭时，调用 destroy 方法，消灭 servlet 对象。

init 方法只被调用一次，即在 servlet 第一次被请求加载时调用该方法。当后续的客户请求 servlet 服务时，Web 服务将启动一个新的线程，在该线程中，servlet 调用 service 方法响应客户的请求，也就是说，每个客户的每次请求都导致 service 方法被调用执行。

7.1.2 init 方法

该方法是 HttpServlet 类中的方法，我们可以在 servlet 中重写这个方法。

方法描述：

```
public void init(ServletConfig  config)  throws ServletException
```

servlet 第一次被请求加载时，服务器初始化一个 servlet，即创建一个 servlet 对象，这个对象调用 init 方法完成必要的初始化工作。该方法在执行时，servlet 引擎会把一个

SevletConfig 类型的对象传递给 init()方法，这个对象就被保存在 servlet 对象中，直到 servlet 对象被消灭，这个 ServletConfig 对象负责向 servlet 传递服务设置信息，如果传递失败就会发生 ServeletException，servlet 就不能正常工作。

我们已经知道，当多个客户请求一个 servlet 时，引擎为每个客户启动一个线程，那么 servlet 类的成员变量被所有的线程共享。

7.1.3 service 方法

该方法是 HttpServlet 类中的方法，我们可以在 servlet 中直接继承该方法或重写这个方法。

方法描述：

```
public void service(HttpServletRequest request  HttpServletResponse response)  throw
                        ServletException,IOException
```

当 servlet 成功创建和初始化之后，servlet 就调用 service 方法来处理用户的请求并返回响应。Servlet 引擎将两个参数传递给该方法，一个 HttpServletRequest 类型的对象，该对象封装了用户的请求信息，此对象调用相应的方法可以获取封装的信息，即使用这个对象可以获取用户提交的信息。另外一个参数对象是 HttpServletResponse 类型的对象，该对象用来响应用户的请求。和 init 方法不同的是，init 方法只被调用一次，而 service 方法可能被多次的调用，我们已经知道，当后续的客户请求 servlet 服务时，Servlet 引擎将启动一个新的线程，在该线程中，servlet 调用 service 方法响应客户的请求，也就是说，每个客户的每次请求都导致 service 方法被调用执行，调用过程运行在不同的线程中，互不干扰。

7.1.4 destroy 方法

该方法是 HttpServlet 类中的方法。servlet 可直接继承这个方法，一般不需要重写。

方法描述：

```
public destroy()
```

当 Servlet 引擎终止服务时,比如关闭服务器等,destroy()方法会被执行,消灭 servlet 对象。

7.2 编译和安装 servlet

7.2.1 简单的 servlet 例子

在下面的例子 1 中，Hello 扩展了 HttpServlet。

例子 1

servlet 源文件

Hello.java：

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

```
import javax.servlet.http.*;
public class Hello extends HttpServlet
{  public void init(ServletConfig config) throws ServletException
    {super.init(config);
    }
  public void service(HttpServletRequest reqest,HttpServletResponse response)
throws IOException
    { //获得一个向客户发送数据的输出流：
      PrintWriter out=response.getWriter();
      response.setContentType("text/html;charset=GB2312");//设置响应的 MIME
类型。
      out.println("<HTML> <BODY>");
      out.println("Simple servlet");
      out.println("</body> </html>");
    }
}
```

7.2.2 编译 servlet

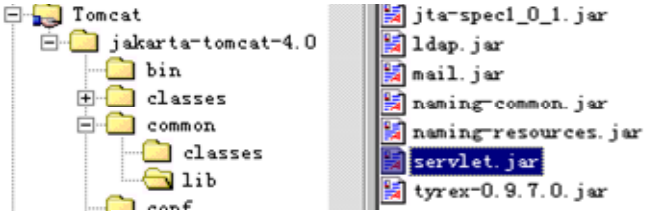

图 7.1　编译 servlet 所需要的 jar 文件

为了编译 servlet 源文件，需要 HttpServlet、HttpServletRequest 等类，JDK 内置包中并不包含这些类文件。为了能编译 servlet 源文件得到创建 servlet 用的字节码文件，需要在环境变量中包含 servlet.jar，这个 jar 文件在 TOMCAT 安装目录的 common\lib 文件下，如图 7.1 所示（也可以到 sun 公司网站下载编译 servlet 所需要的类）。

对于 window2000，用鼠标右键点击“我的电脑”，弹出菜单，然后选择属性，弹出“系统特性”对话筐，再单击该对话框中的高级选项，然后点击按钮“环境变量”，编辑 classpath，添加新的环境变量的值：

D:\Tomcat\jakarta-tomcat-4.0\common\lib\servlet.jar;

我们将上述 servlet 的源文件 Hello.java 保存到 F:\2000，然后编译生成字节码文件 Hello.class

7.2.3　存放 servlet 的目录

（1）所有 web 服务目录可使用的 servlet 的存放位置

如果让所有 web 服务目录都可以使用该 servlet，那么创建这个 servlet 的字节码文件需存放在 Tomcat 安装目录的 classes 目录中，例如，本书所用机器的目录就是：D:tomcat\Jakarta-tomcat-4.0\classes，如图 7.1 所示。

我们已经知道，servlet 第一次被请求加载时，服务器初始化一个 servlet，即创建一个 servlet 对象，这对象调用 init 方法完成必要的初始化工作。如果你对 servlet 的源文件进行了修改，并将新的字节码文件存放到 classes 中，如果服务器没有关闭的话，新的 servlet 不会被创建，因为，当后续客户请求 servlet 服务时，已初始化的 servlet 将调用 service 方法响应客户。

(2) 只对 examples 服务目录可用的 seclet 的存放目录

examples 是 TOMCAT 引擎的默认 web 服务目录之一。

如果想让某个 servlet 只对 examples 目录可用，那么创建该 servlet 的字节码文件只需存放在 webapps/example/Web-inf/classes 目录中。

存放在该目录中的 servlet 和存放在上面（1）中所述目录中的 servlet 有所不同，服务器引擎首先检查 webapps/example/Web-inf/classes 目录中的创建该 servlet 的字节码文件是否被修改过，如果重新修改过，就会用消灭 servlet，用新的字节码重新初始化 servlet。

如果经常调试 servlet，可以把 servlet 放在 webapps/example/Web-inf/classes。需要注意的是，当用户请求 servlet 服务时，由于服务器引擎每次都要检查字节码文件是否被修改过，导致服务器的运行效率降低。

7.2.4 运行 servlet

如果一个 servlet 对所有的 web 服务目录可用，那么只要在服务器引擎启动后，在浏览器地址栏键入：

http://localhost:8080/web 服务目录/servlet/创建 servlet 类的名字

即可，例如，对于用上述 Hello 创建的 servlet，

(1) Root 服务目录

http://localhost:8080/servlet/Hello

(2) friend 目录（我们自定义的一个 web 服务目录）

http://localhost:8080/friend/servlet/Hello

如果是只对 examples 服务目录可用的 servelt，那么只要在服务器引擎启动后，在浏览器地址栏键入：

http://localhost:8080/examples/servlet/创建 servlet 类的名字

我们将 Hello.class 文件保存到 Tomcat 引擎的 classes 文件夹中。图 7.2 和 7.3 是在不同的 web 目录下运行 servlet 的效果。

图 7.2 用 Web 根目录访问 servlet

图 7.3 用 friend 目录访问

7.2.5 带包名的 servlet

在写一个 servlet 的 java 文件时，可以使用 package 语句给 servlet 一个包名。包名可以是一个合法的标识符，也可以是若干个标识符加“.”分割而成，如：

```
package gping;
package tom.jiafei;
```

程序如果使用了包语句，例如

```
package tom.jiafei;
```

那么在 classes 目录下需有如下的子目录，例如，在 D:\Tomcat\jakarta-tomcat-4.0\classes 下再建立如下的目录结构。

\tom\jiafei

并将 servlet 的字节码文件存在该目录中，如图 7.4 所示。

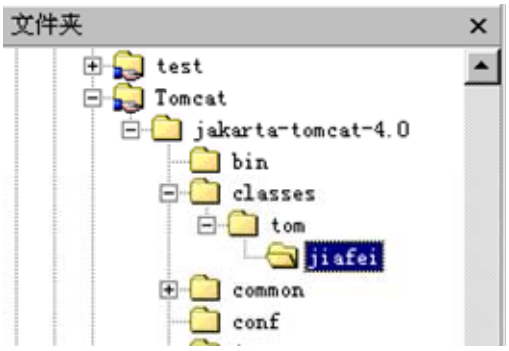

图 7.4 存放带包名的 servlet

如果 servlet 有包名，比如，Hello 的包名是 tom.jiafei，那么调用该 servlet 的 URL 是：

http://localhost:8080/web 服务目录/servlet/tom.jiafei.Hello

因为起了包名，Hello 的全名是 tom.jiafei.Hello（就好比大连的全名是：中国.辽宁.大连）。

7.3 通过 JSP 页面调用 servlet

7.3.1 通过表单向 servlet 提交数据

任何一个 Web 服务目录下的 JSP 页面都可以通过表单或超链接访问某个 servlet。通过 JSP 页面访问 servlet 的好处是，JSP 页面可以负责页面的静态信息处理，动态信息处理交给 servlet 去完成。

在下面的例子中，JSP 页面通过表单向 servlet 提交一个正实数，servlet 负责计算这个数的平方根返回给客户。

为了方便地调试 servlet，本书中，servlet 的字节码文件存放在 D:\Tomcat\jakarta-tomcat-4.0\webapps\example\Web-inf\classes 中，那么在 JSP 页面中调用 servlet 时，servlet 的 URL 是：

/examples/servlet/servletName

在下面的例子 2 中，JSP 页面通过表单提交一个正数，servlet 负责计算这个数的平方根。

例子 2

调用 servlet 的页面(该页面存放在 web 服务的根目录 Root 中)

givenumber.jsp(效果如图 7.5 所示)

```
<%@ page contentType="text/html;charset=GB2312" %>
<HTML>
<BODY bgcolor=cyan><Font size=1>
<P>输入一个数，servlet 求这个数的平方根：
<FORM action="examples/servlet/Sqrt" method=get>
  <Input Type=text name=number>
  <Input Type=submit value="提交">
</FORM>
</BODY>
</HTML>
```

图 7.5　提交数字的 JSP 页面

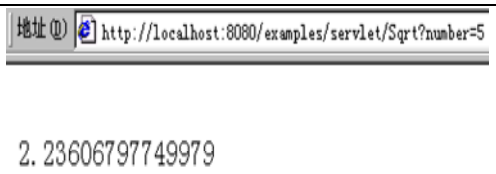

图 7.6　负责计算平方根的 servlet

servlet 源文件(效果如图 7.6 所示)

Sqrt.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Sqrt extends HttpServlet
{   public void init(ServletConfig config) throws ServletException
    {super.init(config);
    }
  public void service(HttpServletRequest request,HttpServletResponse
response) throws IOException
    { //获得一个向客户发送数据的输出流：
       PrintWriter out=response.getWriter();
       response.setContentType("text/html;charset=GB2312");//设置响应的 MIME
类型。
       out.println("<HTML> <BODY>");
       String number=request.getParameter("number");    //获取客户提交的信息。
       double n=0;
        try{ n=Double.parseDouble(number);
             out.print("<BR>"+Math.sqrt(n));
           }
        catch(NumberFormatException e)
           { out.print("<H1>input number letter please! </H1>");
           }
       out.println("</body> </html>");
    }
}
```

7.3.2　通过超链接访问 servlet

我们可以在 JSP 页面中，点击一个超链接，访问 servlet。

例子 3

connection.jsp:

```
<%@ page contentType="text/html;charset=GB2312" %>
<HTML>
<BODY bgcolor=cyan><Font size=1>
<A Href="/servlet/Hello" >加载 servlet<A>
</BODY>
</HTML>
```

7.4 servlet 的共享变量

我们已经知道，在 servlet 被加载之后，当后续的客户请求 servlet 服务时，引擎将启动一个新的线程，在该线程中，servlet 调用 service 方法响应客户的请求，而且 servlet 类中定义的成员变量，被所有的客户线程共享。在下面的例子 4 中，利用共享变量实现了一个计数器。

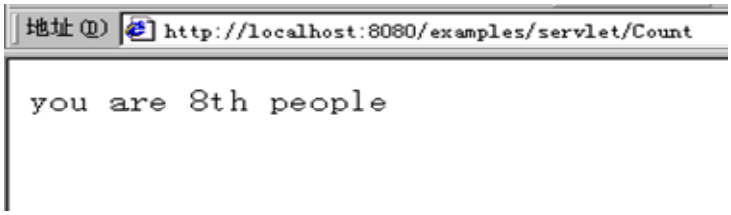

图 7.7　计数器

例子 4(效果如图 7.7 所示)

Count.java：

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Count extends HttpServlet
{ int count;
  public void init(ServletConfig config) throws ServletException
    {super.init(config);
     count=0;
    }
  public synchronized void service(HttpServletRequest
request,HttpServletResponse response)
                                throws IOException
    { //获得一个向客户发送数据的输出流：
       PrintWriter out=response.getWriter();
       response.setContentType("text/html;charset=GB2312");//设置响应的 MIME
类型。
       out.println("<HTML> <BODY>");
```

```
            count++;                              //增加计数。
            out.println("you are "+count+"th"+" people");
            out.println("</body> </html>");
        }
    }
```

注:在处理多线程问题时，我们必须注意这样一个问题：当两个或多个线程同时访问同一个变量，并且一个线程需要修改这个变量。我们应对这样的问题作出处理，否则可能发生混乱.所以上述例子中的 service 方法是一个 synchronized 方法。

数学上有一个计算π的公式：

π/4=1-1/3+1/5-1/7+1/9-1/11…　…

下面的例子中利用成员变量被所有客户共享这一特性实现客户帮助计算 的值，即每当客户请求访问 servlet 时都参与了一次　的计算。

客户通过点击一个 JSP 页面的超链接访问一个计算 的 servlet

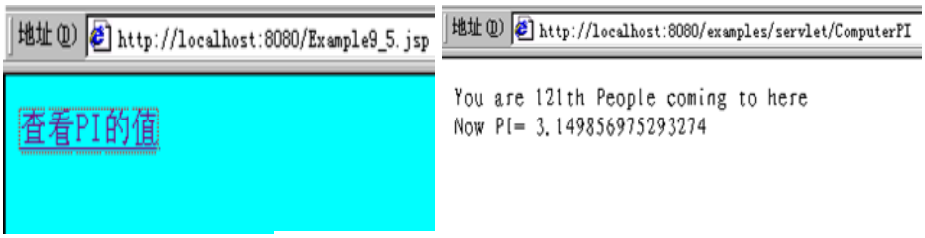

图 7.8　用 servlet 计算 PI

例子 5(效果如图 7.8 所示)

JSP 页面

Example7_5.jsp：

```
<%@ page contentType="text/html;charset=GB2312" %>
<HTML>
<BODY bgcolor=cyan>
<A Href="examples/servlet/ComputerPI" >查看 PI 的值<A>
</BODY>
</HTML>
```

servlet 源文件

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ComputerPI extends HttpServlet
{ double sum=0,i=1,j=1;
```

```
       int number=0;
       public void init(ServletConfig config) throws ServletException
         {super.init(config);
         }
       public synchronized void  service(HttpServletRequest
request,HttpServletResponse response)
                              throws IOException
         { //获得一个向客户发送数据的输出流：
            PrintWriter out=response.getWriter();
            response.setContentType("text/plain");//设置响应的 MIME 类型为纯文本。
              number++;
              sum=sum+i/j;
              j=j+2;
              i=-i;
            out.println("You are "+number+"th People coming to here");
            out.println("Now PI= "+4*sum);
         }
     }
```

7.5 HttpServlet 类

7.5.1 doGet 方法和 doPost 方法

HttpServlet 除了 init、service、destroy 方法外，该类还有两个很重要的方法：doGet 和 doPost，用来处理客户的请求并作出响应。

当服务器引擎第一次接受到一个 servlet 请求时，会使用 init 方法初始化一个 servlet，以后每当服务器再接受到一个 servlet 请求时，就会产生一个新线程，并在这个线程中调用 service 方法检查 HTTP 请求类型（Get 、Post 等），并在 service 方法中根据用户的请求方式，对应地再调用 doGet 或 doPost 方法。因此，在 servlet 类中，我们不必重写 service 方法来响应客户，直接继承 service 方法即可。我们可以在 servlet 类中重写 doPost 或 doGet 方法来响应用户的请求，这样可以增加响应的灵活性，并降低服务器的负担。

如果不论用户请求类型是 Post 还是 Get，服务器的处理过程完全相同，那么我们可以只在 doPost 方法中编写处理过程，而在 doGet 方法中再调用 doPost 方法即可，或只在 doGet 方法中编写处理过程，而在 doPost 方法中再调用 doGet 方法（见例子 6）。

如果根据请求的类型进行不同的处理，就需在两个方法中编写不同的处理过程（见例子 7）。

在下面的例子 6 中，用户可以通过两个表单向 servlet 提交一个正数，其中一个表单的提交方式是 post，另一个表单的方式是 get。无论用户用那种方式，服务器的 servlet 都计算这个数的全部因数，返回给用户。而在下面的例子 7 中，如果使用 post 方式提交正数，servlet 计算这个数的全部因数，如果使用 get 方式，servlet 求出小于这个数的全部素数。

例子 6(效果如图 7.9 所示)

提交正数的 JSP 页面

Example7_6.jsp：

```
<%@ page contentType="text/html;charset=GB2312" %>
<HTML>
<BODY bgcolor=cyan><Font size=1>
<P>输入一个数，提交给 servlet(Post 方式)：
<FORM action="examples/servlet/ComputerFactor" method=post>
  <Input Type=text name=number>
  <Input Type=submit value="提交">
</FORM>
<P>输入一个数，提交给 servlet(Get 方式)：
<FORM action="examples/servlet/ComputerFactor" method=get>
  <Input Type=text name=number>
  <Input Type=submit value="提交">
</FORM>
</BODY>
</HTML>
```

sevlet 源文件

ComputerFacor.java：

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ComputerFactor extends HttpServlet
{
  public void init(ServletConfig config) throws ServletException
    {super.init(config);
    }
  public  void  doPost(HttpServletRequest request,HttpServletResponse
response)
                          throws ServletException,IOException
    {  //获得一个向客户发送数据的输出流：
       PrintWriter out=response.getWriter();
       response.setContentType("text/html;charset=GB2312");//设置响应的 MIME
类型。
       out.println("<HTML>");
       out.println("<BODY>");
```

```
            String number=request.getParameter("number");  //获取客户提交的信息。
            double n=0;
             try{ n=Double.parseDouble(number);
                  out.println("<H1> factors of "+n+" :</H1>");
                  //求 n 的全部因数：
                  for(int i=1;i<=n;i++)
                      { if(n%i==0)
                        out.println(i);
                      }
                }
             catch(NumberFormatException e)
                { out.print("<H1>input number letter please! </H1>");
                }
         }
       public  void  doGet(HttpServletRequest request,HttpServletResponse
response)
                              throws ServletException,IOException
         {
            doPost(request,response);
         }
     }
```

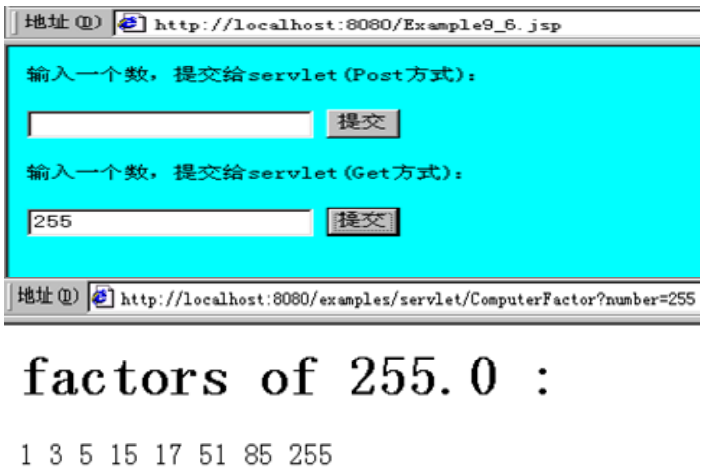

图 7.9　Post、Get 处理方式相同

例子 7(效果如图 7.10 所示)

提交正数的 JSP 页面

Example7_7.jsp：

```
<%@ page contentType="text/html;charset=GB2312" %>
<HTML>
```

```
<BODY bgcolor=cyan><Font size=1>
<P>输入一个数，提交给 servlet(Post 方式)：
<FORM action="examples/servlet/ComputerFactorandPrimNumber" method=post>
  <Input Type=text name=number>
  <Input Type=submit value="提交">
</FORM>
<P>输入一个数，提交给 servlet(Get 方式)：
<FORM action="examples/servlet/ComputerFactorandPrimNumber" method=get>
  <Input Type=text name=number>
  <Input Type=submit value="提交">
</FORM>
</BODY>
</HTML>
```

sevlet 源文件

ComputerFacorandPrimNumber.java：

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ComputerFactorandPrimNumber  extends HttpServlet
{
  public void init(ServletConfig config) throws ServletException
    {super.init(config);
    }
  public  void  doPost(HttpServletRequest request,HttpServletResponse
response)
                          throws ServletException,IOException
    {  //获得一个向客户发送数据的输出流：
       PrintWriter out=response.getWriter();
       response.setContentType("text/html;charset=GB2312");//设置响应的 MIME
类型。
       out.println("<HTML>");
       out.println("<BODY>");

       String number=request.getParameter("number");  //获取客户提交的信息。
       double n=0;
        try{ n=Double.parseDouble(number);
             out.println("<H1> factors of "+n+" :</H1>");
```

```
                    //求 n 的全部因数：
                    for(int i=1;i<=n;i++)
                        { if(n%i==0)
                          out.println(i);
                        }
                   }
                catch(NumberFormatException e)
                   { out.print("<H1>input number letter please! </H1>");
                   }
            }
          public  void  doGet(HttpServletRequest request,HttpServletResponse
response)
                               throws ServletException,IOException
            {  PrintWriter out=response.getWriter();
               response.setContentType("text/html;charset=GB2312");//设置响应的 MIME
类型。
               out.println("<HTML>");
               out.println("<BODY>");

               String number=request.getParameter("number");  //获取客户提交的信息。
               double n=0;
                try{ n=Double.parseDouble(number);
                     out.println("<H1> Primnumbers less "+n+" :</H1>");
                     //求小于 n 的全部素数：
                     int j=1;
                     for(int i=1;i<n;i++)
                         { for(j=2;j<i;j++)
                            {if(i%j==0)
                                break;
                            }
                           if(j>=i)
                            { out.println(i);
                            }
                         }
                   }
                catch(NumberFormatException e)
                   { out.print("<H1>input number letter please! </H1>");
                   }
```

```
        }
    }
```

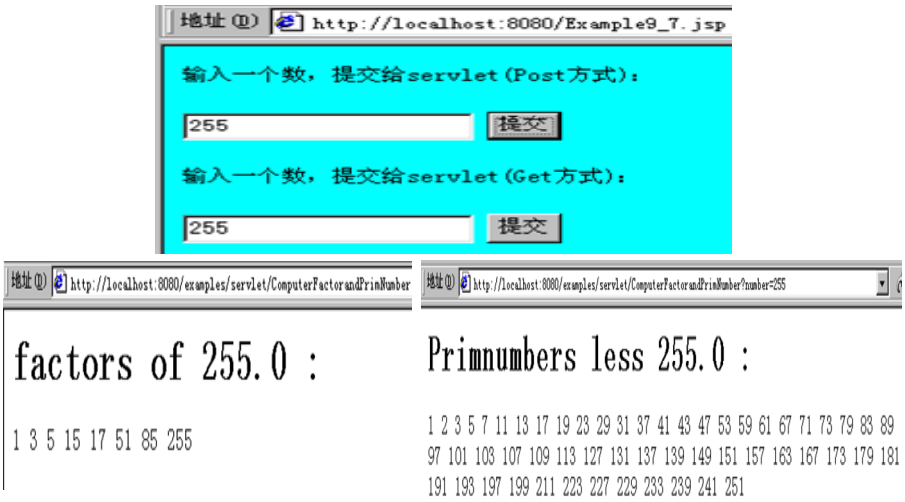

图 7.10　Post、Get 处理方式不相同

7.5.2　处理 HTTP 请求头及表单信息

有关 HTTP 请求头的和表单的介绍，可参见第 3 章。

在下面的例子 8 中，servlet 显示请求的 HTTP 头的值和表单提交的信息（可参考对比第 3 章例子 4）。

例子 8

提交信息的 JSP 页面

Example7_8.jsp：

```
<HTML>
<BODY bgcolor=cyan><FONT size=1>
<%@ page contentType="text/html;charset=GB2312" %>
   <FORM action="examples/servlet/GetMessages" method=post name=form>
       <INPUT type="text" name="boy">
       <INPUT TYPE="submit" value="enter" name="submit">
   </FORM>
</FONT>
</BODY>
</HTML>
```

处理 HTTP 请求头的 sevlet 源文件

GetMessages.java：

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

```
import javax.servlet.*;
import javax.servlet.http.*;
public class GetMessages  extends HttpServlet
{
  public void init(ServletConfig config) throws ServletException
    {super.init(config);
    }
  public  void  doPost(HttpServletRequest request,HttpServletResponse
response)
                          throws ServletException,IOException
    {  //获得一个向客户发送数据的输出流：
       PrintWriter out=response.getWriter();
       response.setContentType("text/html;charset=GB2312");//设置响应的 MIME
类型。
       out.println("<HTML>");
       out.println("<BODY>");
       // 客户使用的协议是:
        out.println("<BR>Protocol:");
        String protocol=request.getProtocol();
        out.println(protocol);
        //获取接受客户提交信息的 servlet：
        out.println("<BR>Accept servlet:");
        String path=request.getServletPath();
        out.println(path);
        //客户提交的信息的长度:
        out.println("<BR>message Length:");
        int length=request.getContentLength();
        out.println(length);
       // 客户提交信息的方式：
        out.print("<BR> Method:");
        String method=request.getMethod();
        out.println(method);
        //获取 HTTP 头文件中 User-Agent 的值：
        out.println("<BR> User-Agent:");
        String header1=request.getHeader("User-Agent");
        out.println(header1);
       //获取 HTTP 头文件中 accept 的值：
         out.println("<BR> accept:");
```

```
        String header2=request.getHeader("accept");
        out.println(header2);
       // 获取 HTTP 头文件中 Host 的值：
         out.println("<BR> Host:");
        String header3=request.getHeader("Host");
        out.println(header3);
       //获取 HTTP 头文件中 accept-encoding 的值:
        out.println("<BR> accept-encoding:");
        String header4=request.getHeader("accept-encoding");
        out.println(header4);
      //获取客户的 IP 地址：
        out.println("<BR> client IP:");
        String  IP=request.getRemoteAddr();
        out.println(IP);
      // 获取客户机的名称：
        out.println("<BR> client name:");
        String clientName=request.getRemoteHost();
        out.println(clientName);
      // 获取服务器的名称：
        out.println("<BR> server name:");
        String serverName=request.getServerName();
        out.println(serverName);
      // 获取服务器的端口号：
        out.println("<BR> ServerPort:");
        int serverPort=request.getServerPort();
        out.println(serverPort);
       //获取客户端提交的所有参数的名字：
        out.println("<BR>Parameter Names");
        Enumeration enum=request.getParameterNames();
             while(enum.hasMoreElements())
              {String s=(String)enum.nextElement();
                out.println(s);
              }
     //  文本框 text 提交的信息:
         out.println("<BR> text:");
         String str=request.getParameter("boy");
                out.println(str);
      out.println("</BODY>");
```

```
            out.println("</HTML>");
          }
        public  void  doGet(HttpServletRequest request,HttpServletResponse
response)
                              throws ServletException,IOException
          {
            doPost(request,response);
          }
      }
```

下面的例子 9 用 servlet 实现用户注册。用户通过一个 JSP 页面提交姓名和 email 地址实现注册。当 servlet 获取这些信息后，首先检查散列表对象中是否已经存在这个名字，该散列表存储了已经注册的用户的名字。如果目前准备注册的用户提交的名字在散列表中已经存在，就提示客户更换名字，否则将检查客户是否提供了书写正确的 email 地址，如果提供了书写正确 email 地址将允许注册（仅仅要求 email 地址中不允许出现空格）。

例子 9(效果如图 7.11 所示)

提交注册名字的 JSP 页面

Example7_9.jsp：

```
<%@ page contentType="text/html;charset=GB2312" %>
<HTML>
<BODY bgcolor=cyan><Font size=1 >
  <FORM action="examples/servlet/LoginByServlet" method=post >
     <P>输入你的姓名:
     <INPUT type="text" name="name" value="abc">
     <BR>
     <P>输入你的 e-mail 地址:
      <INPUT type="text" name="address" value="ookk@sina.com">
     <P>点击送出按钮：
     <BR>
     <INPUT TYPE="submit" value="送出" name=submit>
  </FORM>
</FONT>
</BODY>
</HTML>
```

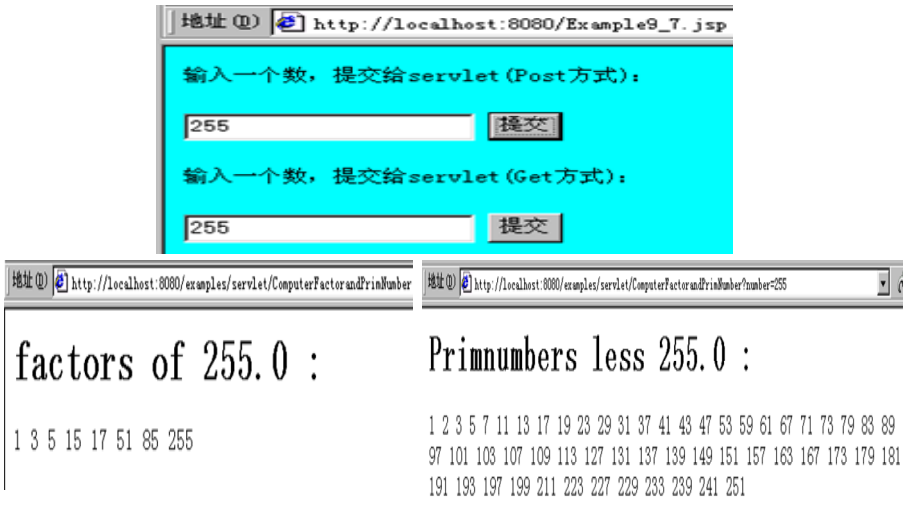

图 7.10　Post、Get 处理方式不相同

sevlet 源文件

LoginByServlet.java：

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class LoginByServlet  extends HttpServlet
{  Hashtable hashtable=new Hashtable();

  public void init(ServletConfig config) throws ServletException
    {super.init(config);
    }
  public synchronized void doPost(HttpServletRequest
request,HttpServletResponse response)
                          throws ServletException,IOException
    {  //获得一个向客户发送数据的输出流：
       PrintWriter out=response.getWriter();
       response.setContentType("text/html;charset=GB2312");OutputStream();
       response.setContentType("text/html");//设置响应的 MIME 类型。
       out.println("<HTML>");
       out.println("<BODY>");
       //获取用户提交的名字：
       String person_name=request.getParameter("name"),
              name_found=null;
         if(person_name==null)
```

```
                {person_name="";
                }
            //在散列表查找是否已存在该名字：
            name_found=(String)hashtable.get(person_name);
           if(name_found==null)
              { String person_email=request.getParameter("address");
                  if(person_email==null)
                      {person_email="";
                      }
                StringTokenizer fenxi=new StringTokenizer(person_email,"  @");
                int n=fenxi.countTokens();
                 if(n>=3)
                     {out.print("<BR>"+"there are exists illegal letters in your
email");
                      }
                 else
                    { hashtable.put(person_name,person_name);
                      out.print("<BR>"+"login success!");
                      out.print("<BR>"+"your name is "+person_name);
                    }
              }
             else
              {out.print("<BR>"+"This name has exist ");
              }
           out.println("</BODY>");
           out.println("</HTML>");
        }


      public synchronized void  doGet(HttpServletRequest
request,HttpServletResponse response)
                          throws ServletException,IOException
        {
          doPost(request,response);
        }
    }
```

7.5.3 设置响应的 HTTP 头

有关响应的 HTTP 头介绍，可参见第 3 章。

在下面的例子 10 中，servlet 设置响应头：Refresh 的头值是 2，那么该 servlet 在 2

秒钟后自动刷新，即 servlet 在 2 秒钟后重新调用 service 方法响应用户。

例子 10(效果如图 7.12 所示)

调用 servlet 的 JSP 页面

Example7_10：

```
<%@ page contentType="text/html;charset=GB2312" %>
<HTML>
<BODY bgcolor=cyan>
<A Href="examples/servlet/DateServlet" >查看时间<A>
</BODY>
</HTML>
```

图 7.12　用 servlet 显示时间

sevlet 源文件

DateServlet.java：

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class DateServlet  extends HttpServlet
{  public void init(ServletConfig config) throws ServletException
    {super.init(config);
    }
  public  void doPost(HttpServletRequest request,HttpServletResponse
response)
                        throws ServletException,IOException
    {  //获得一个向客户发送数据的输出流：
       PrintWriter out=response.getWriter();
       response.setContentType("text/html;charset=GB2312");//设置响应的 MIME
类型。
       out.println("<HTML>");
       out.println("<BODY>");
       response.setHeader("Refresh","2");  //设置 Refresh 的值。
       out.println("Now Time:");
```

```
            out.println("<BR>"+new Date());
            out.println("</BODY>");
            out.println("</HTML>");
          }
       public  void  doGet(HttpServletRequest request,HttpServletResponse
response)
                              throws ServletException,IOException
          {
            doPost(request,response);
          }
     }
```

下面例子 11 实现 servlet 的重定向，客户访问 servlet：Day；如果访问的时间是在 22 点之后，就被重定向到 servlet：Night，提醒用户休息。Day 和 Night 被存放在 examples/Web-inf/classes 中。

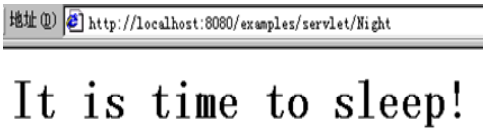

图 7.13　servlet 重定向

例子 11(效果如图 7.13 所示)

Day.java：

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Day extends HttpServlet
{
  public void init(ServletConfig config) throws ServletException
    {super.init(config);
    }
  public  void doPost(HttpServletRequest request,HttpServletResponse
response)
                              throws ServletException,IOException
    {  //获得一个向客户发送数据的输出流：
       PrintWriter out=response.getWriter();
       response.setContentType("text/html;charset=GB2312");//设置响应的 MIME
```

类型。

```
            out.println("<HTML>");
            out.println("<BODY>");
            Calendar calendar=Calendar.getInstance(); //创建一个日历对象。
            calendar.setTime(new Date());//用当前时间初始化日历时间。
            int hour=calendar.get(Calendar.HOUR_OF_DAY),
            minute=calendar.get(Calendar.MINUTE),
            second=calendar.get(Calendar.SECOND);
            if(hour>=22)
                {response.sendRedirect("Night");   //重定向。
                }
             else
                { out.print("Now time :");
                  out.print(hour+":"+minute+":"+second);
                }
          out.println("</BODY>");
          out.println("</HTML>");
        }

      public void doGet(HttpServletRequest request,HttpServletResponse response)
                           throws ServletException,IOException
        {
          doPost(request,response);
        }
    }
```

Night.java：

```
    import java.io.*;
    import java.util.*;
    import javax.servlet.*;
    import javax.servlet.http.*;
    public class Night extends HttpServlet
    {
      public void init(ServletConfig config) throws ServletException
        {super.init(config);
        }
      public  void doPost(HttpServletRequest request,HttpServletResponse
response)
```

```
                              throws ServletException,IOException
        {  //获得一个向客户发送数据的输出流：
           PrintWriter out=response.getWriter();
           response.setContentType("text/html;charset=GB2312");//设置响应的 MIME
类型。
           out.println("<HTML>");
           out.println("<BODY>");
           out.println("<H1> It is time to sleep");
           out.println("</BODY>");
           out.println("</HTML>");
        }
      public void doGet(HttpServletRequest request,HttpServletResponse response)
                              throws ServletException,IOException
        {
          doPost(request,response);
        }
    }
```

7.6 用 servlet 读写文件

这节内容涉及到的文件操作及输入、输出流的内容可参见第 4 章。

7.6.1 读取文件的内容

在下面的例子 12 中，通过一个 JSP 页面显示给用户一些 JSP 文件的名字，该 JSP 文件存放在 Root 服务目录下。用户可以通过 Post 或 Get 方式将文件的名字提交给一个 servlet，该 servelt 存放在服务目录 examples 下的 Web-inf/classes 中。这个 servlet 将根据提交方式的不同，分别读取 JSP 文件的源代码给客户，或显示该 JSP 文件的运行效果给客户。

例子 12(效果如图 7.14、7.15、7.16 所示)

提交文件名字的 JSP 页面

read.jsp:

```
<%@ page contentType="text/html;charset=GB2312" %>
<%@ page import ="java.io.*" %>
<%! class FileJSP implements FilenameFilter
     { String str=null;
         FileJSP(String s)
         {str="."+s;
         }
      public  boolean accept(File dir,String name)
         { return name.endsWith(str);
         }
```

```
        }
    %>
    <P>下面列出了服务器上的一些 jsp 文件
     <% File dir=new File("d:/Tomcat/Jakarta-tomcat-4.0/webapps/root/");
       FileJSP file_jsp=new FileJSP("jsp");
       String file_name[]=dir.list(file_jsp);
        for(int i=0;i<5;i++)
               {out.print("<BR>"+file_name[i]);
               }
     %>
     <BR>输入文件的名字读取 JSP 文件的源代码内容：
      <FORM action="examples/servlet/ReadFileServlet" method=post>
         <Input type="text" name="name">
         <Input type=submit value="提交">
      </FORM>
    <BR>输入文件的名字显示该 JSP 文件的运行效果：
      <FORM action="examples/servlet/ReadFileServlet" method=get>
         <Input type="text" name="name">
         <Input type=submit value="提交">
      </FORM>
```

读取文件的 servlet 源文件

ReadFileServlet:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ReadFileServlet extends HttpServlet
{
  public void init(ServletConfig config) throws ServletException
    {super.init(config);
    }
  //doPost 方法使用了回压流来读取 JSP 文件的源代码：
  public  void doPost(HttpServletRequest request,HttpServletResponse
response)
                          throws ServletException,IOException
    {  //获取提交的文件的名字：
       String name=request.getParameter("name");
        //获得一个向客户发送数据的输出流：
```

```
            PrintWriter out=response.getWriter();
            response.setContentType("text/html;charset=GB2312");//设置响应的 MIME
类型。
            out.println("<HTML>");
            out.println("<BODY>");
            File f=new File("d:/Tomcat/Jakarta-tomcat-4.0/webapps/root",name);
            try{ FileReader in=new FileReader(f) ;
                 PushbackReader push=new PushbackReader(in);
                 int c;
                 char b[]=new char[1];
                 while ( (c=push.read(b,0,1))!=-1)//读取 1 个字符放入字符数组 b。
                  { String s=new String(b);
                     if(s.equals("<"))        //回压的条件
                     {  push.unread('&');
                        push.read(b,0,1); //push 读出被回压的字符字节,放入数组 b.
                        out.print(new String(b));
                        push.unread('L');
                        push.read(b,0,1); //push 读出被回压的字符字节,放入数组 b.
                        out.print(new String(b));
                        push.unread('T');
                        push.read(b,0,1); //push 读出被回压的字符字节,放入数组 b.
                        out.print(new String(b));
                     }
                 else if(s.equals(">"))        //回压的条件
                    {   push.unread('&');
                        push.read(b,0,1); //push 读出被回压的字符字节,放入数组 b.
                        out.print(new String(b));
                        push.unread('G');
                        push.read(b,0,1); //push 读出被回压的字符字节,放入数组 b.
                        out.print(new String(b));
                        push.unread('T');
                        push.read(b,0,1); //push 读出被回压的字符字节,放入数组 b.
                        out.print(new String(b));

                    }
                 else if(s.equals("\n"))
                   { out.print("<BR>");
                   }
```

```
                else
                    {out.print(new String(b));
                    }
                }
               push.close();
             }
           catch(IOException e){}
            out.println("</BODY>");
            out.println("</HTML>");
        }
      //doGet 方法将显示 JSP 源文件运行的效果
      public void doGet(HttpServletRequest request,HttpServletResponse response)
                            throws ServletException,IOException
        {  String name=request.getParameter("name");
            //获得一个向客户发送数据的输出流：
           PrintWriter out=response.getWriter();
           response.setContentType("text/html;charset=GB2312");//设置响应的 MIME
类型。
           out.println("<HTML>");
           out.println("<BODY>");
           File f=new File("d:/Tomcat/Jakarta-tomcat-4.0/webapps/root",name);
           try{ FileReader in=new FileReader(f) ;
                BufferedReader bufferin=new BufferedReader(in);
                String str=null;
                while((str=bufferin.readLine())!=null)
                    {out.print("<BR>"+str);
                    }
                bufferin.close();
                in.close();
               }
           catch(IOException e){}
            out.println("</BODY>");
            out.println("</HTML>");

        }
    }
```

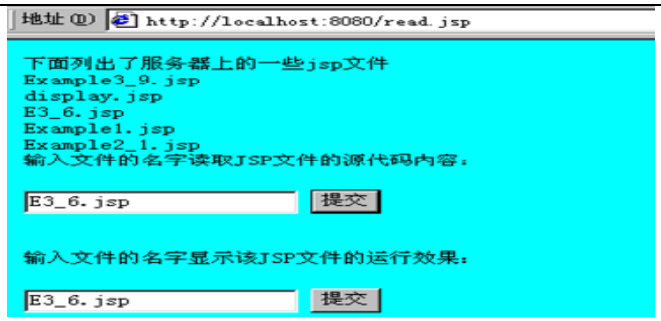

图 7.14　提交文件名字的 JSP 页面

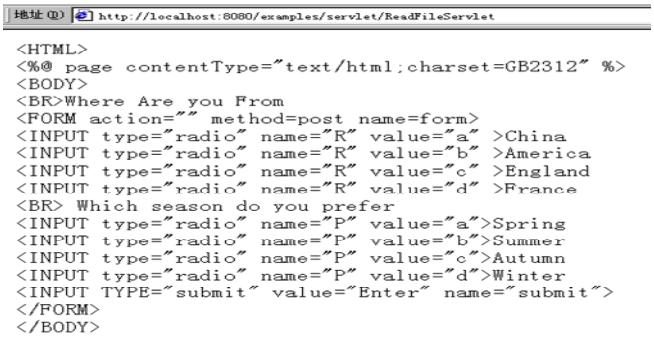

图 7.15　用 servlet 读取 JSP 文件源代码

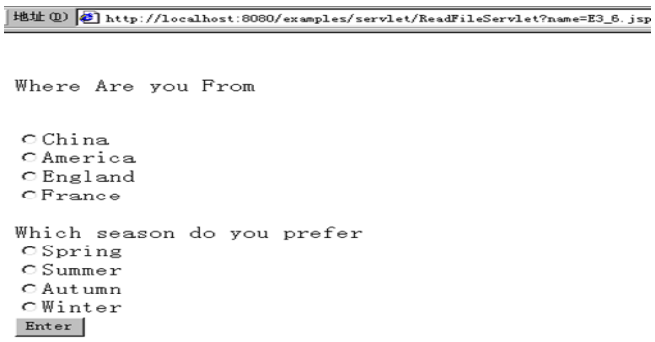

图 7.16　用 servlet 显示 JSP 文件运行效果

7.6.2　写文件

在这节，我们通过一个 servlet 实现小说内容的续写来说明 servlet 在写文件中的技巧。

在下面的例子 13 中，通过一个 JSP 页面显示给用户一个小说文件的已有内容，小说文件存放在服务器的 F:/2000 下，文件名字是 story.txt。JSP 文件存放在 Root 服务目录下。用户可以通过 Post 方式将小说的新内容提交给一个 servlet，该 servelt 存放在服务目录 examples 下的 Web-inf/classes 中。这个 servlet 将用户提交的内容写入小说文件的尾部。

例子 13(效果如图 7.17 所示)

提交小说内容的 JSP 页面

story.jsp:

```
<%@ page contentType="text/html;charset=GB2312" %>
<%@ page import ="java.io.*" %>
<HTML>
<BODY>
```

```
 <H4>小说已有内容：</H4>
<Font size=1 Color=blue>
 <% File f=new File("F:/2000","story.txt");
    //列出小说的内容：
        try{ RandomAccessFile file=
             new RandomAccessFile(f,"r");
             String temp=null;
             while((temp=file.readUTF())!=null)
                {  byte  d[]=temp.getBytes("ISO-8859-1");
                  temp=new String(d);
                  out.print("<BR>"+temp);
                }
             file.close();
           }
        catch(IOException e){}
 %>
    <P>请输入续写的新内容：
    <Form action="examples/servlet/Write" method=post name=form>
      <TEXTAREA name="content" ROWs="12" COLS=80 WRAP="physical">
      </TEXTAREA>
      <BR>
      <INPUT type="submit" value="提交内容" name="submit">
     </FORM>
</BODY>
</HTML>
```

续写文件的 servlet 源文件：

Write.java:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Write extends HttpServlet
{  //声明一个共享的文件和共享字符串：
    File f=null;
    String use="yes" ;
  public void init(ServletConfig config) throws ServletException
    {super.init(config);
    }
```

```
       public  void doPost(HttpServletRequest request,HttpServletResponse
response)
                              throws ServletException,IOException
         {   //获取提交的文件内容：
            String content=request.getParameter("content");
             //获得一个向客户发送数据的输出流：
            PrintWriter out=response.getWriter();
            response.setContentType("text/html;charset=GB2312");//设置响应的 MIME
类型。
            out.println("<HTML>");
            out.println("<BODY>");
            f=new File("F:/2000","story.txt");
           //把对文件的操作放入一个同步块中，并通知
           //其它用户该文件正在被操作中：
           if(use.startsWith("yes"))
             { synchronized(f)
                 {  use="using";
                   try{
                       RandomAccessFile file=new RandomAccessFile(f,"rw");
                       file.seek(file.length()); //定位到文件的末尾。
                       file.writeUTF(content);
                       file.close();
                       use="yes";
                       out.print("<BR>"+"contents have been Write to file");
                      }
                   catch(IOException e){}
                 }
              }
            //如果该小说正在被续写，就通知客户等待：
           else
               {out.print("file is writing,wait please");
               }
            out.println("</BODY>");
            out.println("</HTML>");
         }
       public void doGet(HttpServletRequest request,HttpServletResponse response)
                              throws ServletException,IOException
         {
```

```
            doPost(request,response);
        }
    }
```

图 7.17　使用 servlet 续写文件

7.7 用 servlet 访问数据库

有关数据库连接的一些知识可参见第 5 章。本节通过例子说明 servlet 在数据库方面的应用。我们仍然使用第 5 章的数据源 sun ，该数据源为 Server 服务器上的 pubs 数据库，该库有一个表：students。

7.7.1　数据库记录查询

在下面的例子 14 中，客户通过 condition.jsp 页面输入查询条件，例如，查询某个姓名的成绩、查询成绩在某个分数段范围内的学生成绩等等。用户通过 Post 方式提交姓名给 servlet；分数区间通过 Get 方式提交给 servlet。该 servlet 根据不同的提交方式采取相应的查询方法。

例子 14(效果如图 7.18 所示)

提交查询条件的 JSP 页面

condition.jsp:

```
<%@ page contentType="text/html;charset=GB2312" %>
<HTML>
<BODY bgcolor=cyan>
<Font size=1>
<FORM action="examples/servlet/Inquire" Method="post">
```

```
<P>成绩查询
<P>输入姓名:
 <Input type=text name="name">
 <Input type=submit name="g" value="提交">
</Form>
<FORM action="examples/servlet/Inquire" Method="get" >
<P>根据分数查询名单:<BR>
 英语分数在
<Input type=text name="englishmin" value=1>
 和
<Input type=text name="englishmax" value=100>
 之间
<BR> 数学分数在
<Input type=text name="mathmin" value=1>
 和
<Input type=text name="mathmax" value=100>
 之间 <BR>
 <Input type=submit  value="提交">
</Form>
</BODY>
</HTML>
```

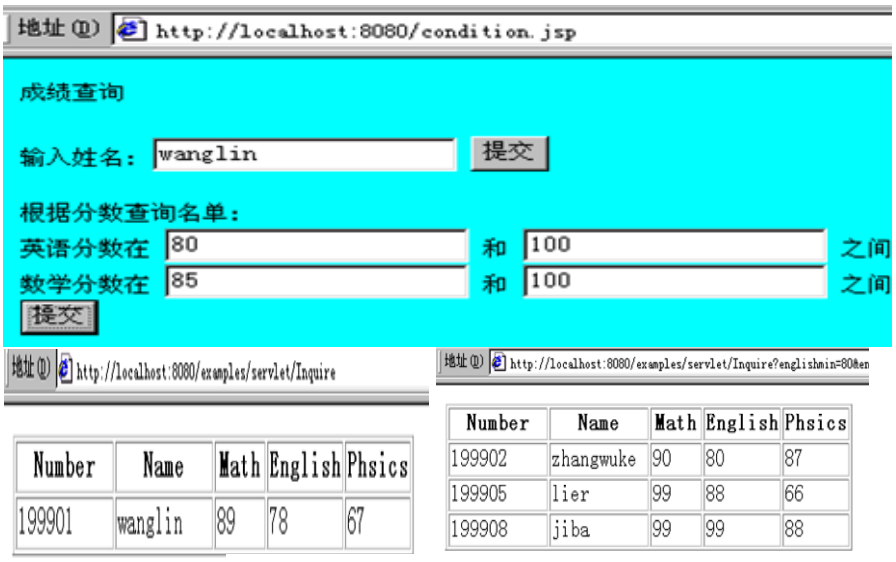

图 7.18　使用 servlet 查询数据库

负责查询的 servlet 源文件：

Inquire.java:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
public class Inquire extends HttpServlet
{  public void init(ServletConfig config) throws ServletException
    {super.init(config);
    }
   //通过 Post 方法按名字查询记录：
  public  void doPost(HttpServletRequest request,HttpServletResponse
response)
                          throws ServletException,IOException
  { PrintWriter out=response.getWriter();
    response.setContentType("text/html;charset=GB2312");//设置响应的 MIME 类
型。
    out.println("<HTML>");
    out.println("<BODY>");
  //获取提交的姓名：
    String name=request.getParameter("name");
    String number,xingming;
    Connection con=null;
    Statement sql=null;
    ResultSet rs=null;
    int math,english,physics;
       try{Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
          }
       catch(ClassNotFoundException e){}
       try
      {    con=DriverManager.getConnection("jdbc:odbc:sun","sa","");
           sql=con.createStatement();
           String condition="SELECT * FROM students WHERE 姓名 =
"+"'"+name+"'";
           rs=sql.executeQuery(condition);
        out.print("<Table Border>");
            out.print("<TR>");
            out.print("<TH width=100>"+"Number");
            out.print("<TH width=100>"+"Name");
            out.print("<TH width=50>"+"Math");
```

```
                out.print("<TH width=50>"+"English");
                out.print("<TH width=50>"+"Phsics");
                out.print("</TR>");
           while(rs.next())
               { out.print("<TR>");
                 number=rs.getString(1);
                 out.print("<TD >"+number+"</TD>");
                 xingming=rs.getString(2);
                 out.print("<TD >"+xingming+"</TD>");
                 math=rs.getInt("数学成绩");
                 out.print("<TD >"+math+"</TD>");
                 english=rs.getInt("英语成绩");
                 out.print("<TD >"+english+"</TD>");
                 physics=rs.getInt("物理成绩");
                 out.print("<TD >"+physics+"</TD>");
                out.print("</TR>") ;
                }
            out.print("</Table>");
           con.close();
          }
         catch(SQLException e)
               {
               }
           out.println("</BODY>");
           out.println("</HTML>");
       }
       //通过 Get 方法按成绩查询记录：
       public void doGet(HttpServletRequest request,HttpServletResponse response)
                            throws ServletException,IOException
       { PrintWriter out=response.getWriter();
         response.setContentType("text/html;charset=GB2312");//设置响应的 MIME 类
型。
         out.println("<HTML>");
         out.println("<BODY>");
        //获取提交的分数的最大值和最小值：
         String englishmax=request.getParameter("englishmax");
         String englishmin=request.getParameter("englishmin");
         String mathmax=request.getParameter("mathmax");
```

```
        String mathmin=request.getParameter("mathmin");
        String number,xingming;
        Connection con=null;
        Statement sql=null;
        ResultSet rs=null;
        int math,english,physics;
           try{Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
              }
           catch(ClassNotFoundException e){}
           try
          {    con=DriverManager.getConnection("jdbc:odbc:sun","sa","");
               sql=con.createStatement();
               String eCondition="英语成绩 <= "+englishmax+" AND "+"英语成绩 >=
"+englishmin;
               String mCondition="数学成绩 <= "+mathmax+" AND "+"数学成绩 >=
"+mathmin;

               String condition="SELECT * FROM students WHERE "+mCondition+" and
"+eCondition;
               rs=sql.executeQuery(condition);
           out.print("<Table Border>");
                out.print("<TR>");
                out.print("<TH width=100>"+"Number");
                out.print("<TH width=100>"+"Name");
                out.print("<TH width=50>"+"Math");
                out.print("<TH width=50>"+"English");
                out.print("<TH width=50>"+"Phsics");
                out.print("</TR>");
           while(rs.next())
               { out.print("<TR>");
                 number=rs.getString(1);
                 out.print("<TD >"+number+"</TD>");
                 xingming=rs.getString(2);
                 out.print("<TD >"+xingming+"</TD>");
                 math=rs.getInt("数学成绩");
                 out.print("<TD >"+math+"</TD>");
                 english=rs.getInt("英语成绩");
                 out.print("<TD >"+english+"</TD>");
```

```
                physics=rs.getInt("物理成绩");
                out.print("<TD >"+physics+"</TD>");
               out.print("</TR>") ;
               }
          out.print("</Table>");
         con.close();
        }
       catch(SQLException e)
        {
        }
       out.println("</BODY>");
       out.println("</HTML>");
     }
  }
```

7.7.2 使用共享连接

数据库操作中，建立连接是耗时最大的操作之一。如果客户访问的是同一数据库，那么，为每个客户都建立一个连接是不合理的。我们已经知道，servlet 的成员变量是被所有用户共享的。这样，我们可以把 Connection 对象作为一个成员变量被所有的客户共享，也就是说第一个访问数据库的客户负责建立连接，以后所有的客户共享这个连接，每个客户都不要关闭这个共享的连接。下面的 servlet 使用共享连接查询数据库的所有记录。

例子 15

使用共享连接的 servlet 源文件

ShareInquire.java:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
public class ShareInquire extends HttpServlet
{  Connection con=null; //共享连接。
  public void init(ServletConfig config) throws ServletException
    {super.init(config);
      //加载 JDBC-ODBC 桥接器：
       try{Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
          }
       catch(ClassNotFoundException e){}
    }
   //通过 Post 方法按名字查询记录：
```

```
       public  void doPost(HttpServletRequest request,HttpServletResponse
response)
                              throws ServletException,IOException
       { PrintWriter out=response.getWriter();
         response.setContentType("text/html;charset=GB2312");//设置响应的 MIME 类
型。
         out.println("<HTML>");
         out.println("<BODY>");
         Statement sql=null;
         ResultSet rs=null;
         if(con==null)
         { try
           {   //第一个用户负责建立连接 con。
               con=DriverManager.getConnection("jdbc:odbc:sun","sa","");
               sql=con.createStatement();
               String condition="SELECT * FROM students";
               rs=sql.executeQuery(condition);
            out.print("<Table Border>");
                out.print("<TR>");
                out.print("<TH width=100>"+"Number");
                out.print("<TH width=100>"+"Name");
                out.print("<TH width=50>"+"Math");
                out.print("<TH width=50>"+"English");
                out.print("<TH width=50>"+"Phsics");
                out.print("</TR>");
          while(rs.next())
              { out.print("<TR>");
                out.print("<TD >"+rs.getString(1)+"</TD>");
                out.print("<TD >"+rs.getString(2)+"</TD>");
                out.print("<TD >"+rs.getInt("数学成绩")+"</TD>");
                out.print("<TD >"+rs.getInt("英语成绩")+"</TD>");
                out.print("<TD >"+rs.getInt("物理成绩")+"</TD>");
                out.print("</TR>") ;
               }
            out.print("</Table>");
          }
         catch(SQLException e)
             {
```

```
            }
        }
       //其它客户通过同步块使用这个连接：
       else
        { synchronized(con)
          {try{ sql=con.createStatement();
                String condition="SELECT * FROM students";
                rs=sql.executeQuery(condition);
                out.print("<Table Border>");
                out.print("<TR>");
                out.print("<TH width=100>"+"Number");
                out.print("<TH width=100>"+"Name");
                out.print("<TH width=50>"+"Math");
                out.print("<TH width=50>"+"English");
                out.print("<TH width=50>"+"Phsics");
                out.print("</TR>");
                while(rs.next())
                 { out.print("<TR>");
                   out.print("<TD >"+rs.getString(1)+"</TD>");
                   out.print("<TD >"+rs.getString(2)+"</TD>");
                   out.print("<TD >"+rs.getInt("数学成绩")+"</TD>");
                   out.print("<TD >"+rs.getInt("英语成绩")+"</TD>");
                   out.print("<TD >"+rs.getInt("物理成绩")+"</TD>");
                   out.print("</TR>") ;
                 }
               out.print("</Table>");
              }
             catch(SQLException e)
               {
               }
           }
        }
       out.println("</BODY>");
       out.println("</HTML>");
     }
      public void doGet(HttpServletRequest request,HttpServletResponse response)
                           throws ServletException,IOException
     {  doPost(request,response);
```

```
    }
  }
```

7.8 会话管理

7.8.1 获取用户的会话

我们已经知道，HTTP 协议是一种无状态协议。一个客户向服务器发出请求（request）然后服务器返回响应（respons），连接就被关闭了。在服务器端不保留连接的有关信息，因此当下一次连接时，服务器已没有以前的连接信息了，无法判断这一次连接和以前的连接是否属于同一客户。因此，必须使用客户的会话，记录有关连接的信息。

一个 servlet 使用 HttpServletRequest 对象 request 调用 getSession 方法获取用户的会话对象：

```
HttpSession session=request.getSession(true);
```

一个用户在不同的 servlet 中获取的 session 对象是完全相同的，不同的用户的 session 对象互不相同。有关会话对象常用方法可参见第 4 章。

在下面的例子 16 中，有两个 servlet，Boy 和 Girl。客户访问 Boy 时，将一个字符串对象，存入自己的会话中，然后访问 Girl，在 Girl 中再输出自己的 session 对象中的字符串对象。

例子 16(效果如图 7.19 所示)

Boy.java：

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Boy extends HttpServlet
{  public void init(ServletConfig config) throws ServletException
    {super.init(config);
    }
   public  void doPost(HttpServletRequest request,HttpServletResponse
response)
                          throws ServletException,IOException
    {  //获得一个向客户发送数据的输出流：
       PrintWriter out=response.getWriter();
       response.setContentType("text/html;charset=GB2312");//设置响应的 MIME
类型。
       out.println("<HTML>");
       out.println("<BODY>");
       HttpSession session=request.getSession(true);  //获取客户的会话对象
```

```
                        session.setAttribute("name","Zhoumin");
            out.println(session.getId());                    //获取会话的 Id.
             out.println("</BODY>");
            out.println("</HTML>");
          }
       public void doGet(HttpServletRequest request,HttpServletResponse response)
                            throws ServletException,IOException
          { doPost(request,response);
          }
     }
```

图 7.19　在 servlet 中使用会话对象

Girl.java：

```
     import java.io.*;
     import javax.servlet.*;
     import javax.servlet.http.*;
     public class Girl extends HttpServlet
     {  public void init(ServletConfig config) throws ServletException
         {super.init(config);
         }
       public  void doPost(HttpServletRequest request,HttpServletResponse
response)
                            throws ServletException,IOException
         {  //获得一个向客户发送数据的输出流：
            PrintWriter out=response.getWriter();
            response.setContentType("text/html;charset=GB2312");//设置响应的 MIME
类型。
            out.println("<HTML>");
            out.println("<BODY>");
            HttpSession session=request.getSession(true);  //获取客户的会话对象
                        session.setAttribute("name","Zhoumin");
            out.println(session.getId());                    //获取会话的 Id.
            String s=(String)session.getAttribute("name"); //获取会话中存储的数
据。
```

```
            out.print("<BR>"+s);
            out.println("</BODY>");
            out.println("</HTML>");
         }
       public void doGet(HttpServletRequest request,HttpServletResponse response)
                              throws ServletException,IOException
         { doPost(request,response);
         }
    }
```

7.8.2 购物车

用户通过一个 JSP 页面：choice.jsp 选择商品，提交给 servlet：AddCar，该 servlet 负责将商品添加到用户的 session 对象中（相当于用户的一个购物车），并将 session 对象中的商品显示给用户。用户可以不断地从 choice.jsp 页面提交商品给 AddCar。用户通过 remove.jsp 页面选择要从购物车中删除的商品提交给 servlet：RemoveGoods，该 servlet 负责从用户的购物车（用户的 session 对象）删除商品。

例子 17(效果如图 7.20、7.21 所示)

负责选择商品的 JSP 页面

choice.jsp:

```
<%@ page contentType="text/html;charset=GB2312" %>
<%@ page import="java.util.*" %>
<%@ page import="Car1" %>
<HTML>
<BODY bgcolor=cyan><Font size=1>
<P>这里是第一百货商场，选择您要购买的商品添加到购物车：
<FORM action="examples/servlet/AddCar" method=post name=form>
       <Select name="item"  value="没选择">
          <Option value="TV">电视机
          <Option value="apple">苹果
          <Option value="coke">可口可乐
          <Option value="milk">牛奶
          <Option value="tea">茶叶
       </Select>
<P>输入购买的数量:
    <Input type=text name="mount">
<P>选择计量单位：
   <INPUT type="radio" name="unit" value="个">个
   <INPUT type="radio" name="unit" value="公斤">公斤
```

```
        <INPUT type="radio" name="unit" value="台">台
        <INPUT type="radio" name="unit" value="瓶">瓶
    <Input type=submit value="提交添加">
    </FONT>
    </BODY>
    </HTML>
```

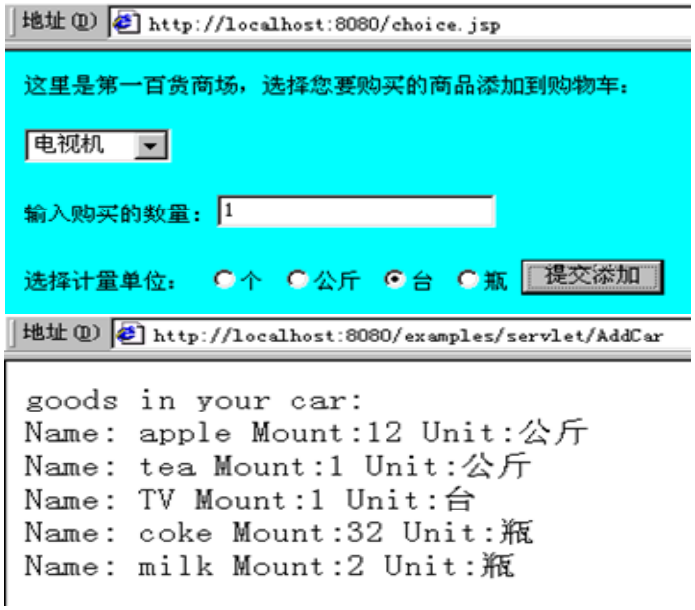

图 7.20　使用 servlet 添加商品

负责添加商品的 servlet

AddCar.java:

```
    import java.io.*;
    import java.util.*;
    import javax.servlet.*;
    import javax.servlet.http.*;
    public class AddCar extends HttpServlet
    {
       public void init(ServletConfig config) throws ServletException
        {super.init(config);
        }
      public  void doPost(HttpServletRequest request,HttpServletResponse
response)
                              throws ServletException,IOException
        {  //获得一个向客户发送数据的输出流：
           PrintWriter out=response.getWriter();
```

```
            response.setContentType("text/html;charset=GB2312");//设置响应的 MIME
类型。
            out.println("<HTML>");
            out.println("<BODY>");
            HttpSession session=request.getSession(true);  //获取客户的会话对象
            String item =request.getParameter("item"),      //获取客户选择的商品
名称。
                   mount=request.getParameter("mount"),    //获取客户购买的数量。
                   unit =request.getParameter("unit");     //获取商品的计量单位。
            //将客户的购买信息存入客户的 session 对象中。
             String str="Name: "+item+"  Mount:"+mount+"  Unit:"+unit;
                    session.setAttribute(item,str);
            //将购物车中的商品显示给客户：
            out.println(" goods in your car: ");
                 Enumeration enum=session.getAttributeNames();
                  while(enum.hasMoreElements())
                    { String name=(String)enum.nextElement();
                      out.print("<BR>"+(String)session.getAttribute(name));
                    }
          }
       public void doGet(HttpServletRequest request,HttpServletResponse response)
                             throws ServletException,IOException
         {
           doPost(request,response);
         }
     }
```

选择删除商品的 JSP 页面

remove.jsp：

```
<%@ page contentType="text/html;charset=GB2312" %>
<%@ page import="java.util.*" %>
<%@ page import="Car1" %>
<HTML>
<BODY bgcolor=cyan><Font size=1>
<P>选择要从购物车中删除的商品：
<FORM action="examples/servlet/RemoveGoods" method=post name=form>
       <Select name="item"  value="没选择">
          <Option value="TV">电视机
```

```
            <Option value="apple">苹果
            <Option value="coke">可口可乐
            <Option value="milk">牛奶
            <Option value="tea">茶叶
        </Select>
    <Input type=submit value="提交删除">
    </FONT>
    </BODY>
    </HTML>
```

图 7.21　使用 servlet 删除商品

负责删除商品的 servlet

RemoveGoods.java

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class RemoveGoods extends HttpServlet
{public void init(ServletConfig config) throws ServletException
    {super.init(config);
    }
public  void doPost(HttpServletRequest request,HttpServletResponse response)
                        throws ServletException,IOException
    {  //获得一个向客户发送数据的输出流：
       PrintWriter out=response.getWriter();
       response.setContentType("text/html;charset=GB2312");//设置响应的 MIME
```

类型。

```
            out.println("<HTML>");
            out.println("<BODY>");
            HttpSession session=request.getSession(true);  //获取客户的会话对象
            String item =request.getParameter("item");      //获取要删除的商品名
称。
                  session.removeAttribute(item);       //删除商品。
             //将购物车中的商品显示给客户：
            out.println("<H3>Now goods in your car:</H3> ");
                 Enumeration enum=session.getAttributeNames();
                  while(enum.hasMoreElements())
                    { String name=(String)enum.nextElement();
                      out.print("<BR>"+(String)session.getAttribute(name));
                    }
          }
     public void doGet(HttpServletRequest request,HttpServletResponse response)
                           throws ServletException,IOException
         { doPost(request,response);
         }
     }
```

7.8.3 猜数字

在第 3 章、第 6 章讲述 JSP 内置对象以及 javabeans 时，曾分别举过猜数字的例子。在这里，我们再使用 servlet 来实现猜数字这个小游戏，这样，我们就用 3 种方式实现了这个小游戏：直接由 JSP 页面来实现、通过 javabeans 来实现、通过 servlet 来实现。

当客户访问 servlet：GetNumber 时，随机分配给客户一个 1 到 100 之间的数，然后将这个数字存在客户的 session 对象中。客户在表单里输入一个数，来猜测分配给自己的那个数字。客户输入一个数字后，提交给 servlet：Result，该 servlet 负责判断这个数是否和客户的 session 对象中存放的那个数字相同，如果相同就连接到 servlet：Success；如果不相同就连接到 servlet：Large 或 Small。然后，客户在这些 servlet 中重新提交数字到 Result。

servlet 源文件

GetNumber.java：

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class GetNumber extends HttpServlet
{  public void init(ServletConfig config) throws ServletException
    {super.init(config);
```

```
        }
      public  void  doPost(HttpServletRequest request,HttpServletResponse
response)
                            throws ServletException,IOException
        {  response.setContentType("text/html");
           ServletOutputStream out=response.getOutputStream();
           out.print("A number between 1 and 100 to you,guess it out please! ");
           HttpSession session=request.getSession(true);
           session.setAttribute("count",new Integer(0));
           int number=(int)(Math.random()*100)+1;        //获取一个随机数。
           session.setAttribute("save",new Integer(number));
           out.print("<FORM action=Result  method=post name=form>");
           out.print("<INPUT type=text name=boy >");
           out.print("<INPUT type=submit value=Enter>");
           out.print("</FORM>");
        }
      public  void  doGet(HttpServletRequest request,HttpServletResponse
response)
                            throws ServletException,IOException
        { doPost(request,response);
        }
    }
```

Result.java：

```
    import java.io.*;
    import java.util.*;
    import javax.servlet.*;
    import javax.servlet.http.*;
    public class Result extends HttpServlet
    { public void init(ServletConfig config) throws ServletException
        {super.init(config);
        }
      public  void  doPost(HttpServletRequest request,HttpServletResponse
response)
                            throws ServletException,IOException
        {  response.setContentType("text/html");
           ServletOutputStream out=response.getOutputStream();
           HttpSession session=request.getSession(true);
```

```
            String str=request.getParameter("boy");
           if(str==null)
             {str="0";
             }
           int guessNumber=Integer.parseInt(str);
           Integer integer=(Integer)session.getAttribute("save");
           int realnumber=integer.intValue();
          if(guessNumber==realnumber)
             { int n=((Integer)session.getAttribute("count")).intValue();
               n=n+1;
               session.setAttribute("count",new Integer(n));
                response.sendRedirect("Success");
             }
            else if(guessNumber>realnumber)
             { int n=((Integer)session.getAttribute("count")).intValue();
               n=n+1;
               session.setAttribute("count",new Integer(n));
               response.sendRedirect("Larger");
             }
          else if(guessNumber<realnumber)
             { int n=((Integer)session.getAttribute("count")).intValue();
               n=n+1;
               session.setAttribute("count",new Integer(n));
               response.sendRedirect("Smaller");
             }
          }
        public  void  doGet(HttpServletRequest request,HttpServletResponse
response)
                              throws ServletException,IOException
          { doPost(request,response);
          }
      }
```

Larger.java：

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

```
public class Larger extends HttpServlet
{  public void init(ServletConfig config) throws ServletException
    {super.init(config);
    }
  public  void  doPost(HttpServletRequest request,HttpServletResponse
response)
                        throws ServletException,IOException
    {  response.setContentType("text/html");
       ServletOutputStream out=response.getOutputStream();
       out.print("Larger ,try again!"); //所猜的数比实际的数大，请再猜。
       out.print("<BR><FORM action=Result  method=post name=form>");
       out.print("<INPUT type=text name=boy >");
       out.print("<INPUT type=submit value=Enter>");
       out.print("</FORM>");
    }
  public  void  doGet(HttpServletRequest request,HttpServletResponse
response)
                        throws ServletException,IOException
    { doPost(request,response);
    }
}
```

Smaller.java

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Smaller extends HttpServlet
{
  public void init(ServletConfig config) throws ServletException
    {super.init(config);
    }
  public  void  doPost(HttpServletRequest request,HttpServletResponse
response)
                        throws ServletException,IOException
    {  response.setContentType("text/html");
       ServletOutputStream out=response.getOutputStream();
       out.print("Smaller ,try again!"); //所猜的数比实际的数小，请再猜。
```

```
            out.print("<BR><FORM action=Result  method=post name=form>");
            out.print("<INPUT type=text name=boy >");
            out.print("<INPUT type=submit value=Enter>");
            out.print("</FORM>");
         }
       public  void  doGet(HttpServletRequest request,HttpServletResponse
response)
                              throws ServletException,IOException
         { doPost(request,response);
         }
     }
```

Success.java

```
     import java.io.*;
     import java.util.*;
     import javax.servlet.*;
     import javax.servlet.http.*;
     public class Success extends HttpServlet
     {
       public void init(ServletConfig config) throws ServletException
         {super.init(config);
         }
       public  void  doPost(HttpServletRequest request,HttpServletResponse
response)
                              throws ServletException,IOException
         {  response.setContentType("text/html");
            ServletOutputStream out=response.getOutputStream();
            HttpSession session=request.getSession(true);
            int count=((Integer)session.getAttribute("count")).intValue();
            int num=((Integer)session.getAttribute("save")).intValue();
            long startTime=session.getCreationTime();
            long endTime=session.getLastAccessedTime();
            long spendTime=(endTime-startTime)/1000;
            out.println("Congratulatuon! You are right");
            out.println("afer just"+count+"tries") ;
            out.println("you spend"+spendTime+"Seconds");
            out.println("That Number is"+num);
         }
```

```
       public  void  doGet(HttpServletRequest request,HttpServletResponse
response)
                              throws ServletException,IOException
         { doPost(request,response);
         }
     }
```