第3章　JSP内置对象

有些对象不用声明就可以在JSP页面的Java程序片和表达式部分使用，这就是JSP的内置对象。

JSP的常用内置对象有resquest、response、session、application和out。

response和request对象是JSP内置对象中较重要的两个，这两个对象提供了对服务器和浏览器通信方法的控制。直接讨论这两个对象前，要先对HTTP协议——Word Wide Web底层协议作简单介绍。

Word Wide Web是怎样运行的呢？在浏览器地址栏输入一个正确的网址后，若一切顺利，网页就出现了。例如，在浏览器地址栏中输入http://www.renren.com，“人人网”的主页就出现在浏览器窗口。这背后是什么在起作用？

使用浏览器从网站获取HTML页面时，实际上是在使用Hypertext Transfer Protocol(HTTP)。HTTP协议规定了信息在Internet上的传输方法，特别是规定了浏览器与服务器的交互方法。

从网站获取页面时，浏览器在网站上打开了一个针对网络服务器的连接，并发出请求。服务器收到请求后回应，所以HTTP协议被称作“请求和响应”协议。

浏览器请求有某种结构，HTTP请求包括请求行、头域和可能的信息体。最普通的请求类型是对页面的一个简单请求，如下例：

```
GET/hello.htm HTTP/1.1
Host:www.sina.com.cn
```

这是对网站www.sina.com.cn上页面hello.htm的HTTP请求的例子。首行是请求行，规定了请求的方法、请求的资源及使用的HTTP协议的版本。请求的方法是“get”方法，此方法获取特定的资源。此处的get方法用来获取名为hello.htm的网页。其他请求方法包括post、head、delete、trace及put方法等。

此例中的第2行是头(header)。Host头给出了网站上hello.htm文件的Internet地址。此例中，主机地址是www.sina.com.cn。

一个典型请求通常包含许多头，称为请求的HTTP头。头提供了关于信息体的附加信息及请求的来源。其中有些头是标准的，有些则和特定的浏览器有关。

一个请求还可能包含信息体。例如，信息体可以包含HTML表单的内容。在HTML表单上单击submit按钮(“提交”按钮)时，该表单内容就由post方法或get方法在请求的信息体中发送。

服务器在收到请求时，返回HTTP响应。响应也有某种结构，每个响应都由状态行开始，可以包含几个头及可能的信息体，称为响应的HTTP头和响应信息体。这些头和信息体由服务器发送给用户的浏览器，信息体就是用户请求的网页的运行结果，对于JSP页面，就是网页的静态信息。状态行说明了正在使用的协议、状态代码及文本信息。例如，若服务

器认为请求出错,则状态行返回错误及对错误的描述,比如“HTTP/1.1 404 Object Not Found”。若服务器成功地响应了对网页的请求,返回包含“200 OK”的状态行。

本章在 webapps 目录下新建一个 Web 服务目录 ch3,除非特别约定,本章例子中涉及的 JSP 页面均保存在 ch3 中。

3.1 request 对象

HTTP 通信协议是用户与服务器之间一种提交(请求)信息与响应信息(request/response)的通信协议。在 JSP 中,内置对象 request 封装了用户提交的信息,那么该对象调用相应的方法可以获取封装的信息,即使用该对象可以获取用户提交的信息。

内置对象 request 是实现了 ServletRequest 接口类的一个实例,可以在 Tomcat 服务器的 webapps\tomcat-docs\servletapi 中查找 ServletRequest 接口的方法。

用户通常使用 HTML 表单向服务器的某个 JSP 页面提交信息,表单的一般格式是:

```
<form  action = "提交信息的目的地页面"  method = get | post  >
    提交手段
</form>
```

其中<form>是表单标记,method 取值 get 或 post。get 方法和 post 方法的主要区别是:使用 get 方法提交的信息会在提交的过程中显示在浏览器的地址栏中,而 post 方法提交的信息不会显示在地址栏中。提交手段包括文本框、列表、文本区等,例如:

```
<form action = "tom.jsp" method = "post" >
      <input type = "text" name = "boy" value = "ok" >
      <input type = "submit" value = "送出" name = "submit">
</form>
```

该表单使用 post 方法向页面 tom.jsp 提交信息,提交信息的手段是 text,即在文本框输入信息,其中默认信息是“ok”;然后单击“送出”按钮(submit)向服务器的 JSP 页面 tom.jsp 提交信息。

request 对象可以使用 getParameter(String s)方法获取该表单通过 text 提交的信息。例如:

```
request.getParameter("boy");
```

3.1.1 获取用户提交的信息

request 对象获取用户提交信息的最常用的方法是 getParameter(String s)。在下面的例子 3_1 中,example3_1.jsp 通过表单向 example3_1_computer.jsp 提交三角形三边的长度,example3_1_computer.jsp 负责计算并显示三角形的面积。example3_1.jsp 和 example3_1_computer.jsp 的效果如图 3.1(a)和图 3.1(b)所示。

(a) 使用表单提交信息　　(b) 获取并处理表单提交的信息

图 3.1　例子 3_1 效果图

例子 3_1

example3_1.jsp(效果如图 3.1(a)所示)

```
<%@ page contentType = "text/html;charset = gb2312" %>
<HTML><body bgcolor = cyan>
    <form action = "example3_1_computer.jsp" method = post >
        <input type = "text" name = "sizeA" value = 1 size = 6 >
        <input type = "text" name = "sizeB" value = 1 size = 6 >
        <input type = "text" name = "sizeC" value = 1 size = 6 >
        <input TYPE = "submit" value = "提交" name = "submit">
    </form>
</body></HTML>
```

example3_1_ computer.jsp(效果如图 3.1(b)所示)

```
<%@ page contentType = "text/html;charset = gb2312" %>
<HTML><bod<%@ page contentType = "text/html;charset = gb2312" %>
<HTML><body bgcolor = yellow>
  <% String sideA = request.getParameter("sizeA");
     String sideB = request.getParameter("sizeB");
     String sideC = request.getParameter("sizeC");
     try {   double a = Double.parseDouble(sideA);
             double b = Double.parseDouble(sideB);
             double c = Double.parseDouble(sideC);
             double p = (a + b + c)/2,area = 0;
             area = Math.sqrt(p * (p - a) * (p - b) * (p - c));
             out.println("<BR>三角形面积" + area);
     }
     catch(NumberFormatException ee){
            out.println("<BR>请输入数字字符");
     }
  %>
</body></HTML>
```

在下面的例子 3_2 中,example3_2.jsp 通过表单向自己提交一串用 # 号分隔的数字,然后计算这些数字的算术和。如果表单中的 action 请求的页面是当前页面,可以用双引号""代替当前页面,注意双引号中不能含有空格。example3_2.jsp 效果如图 3.2 所示。

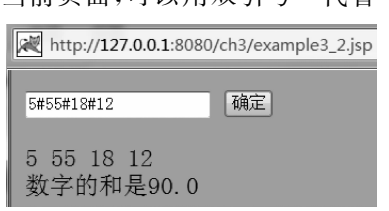


图 3.2　向当前页面提交信息

例子 3_2

example3_2.jsp(如图 3.2 所示)

```
<%@ page
contentType = "text/html;charset = gb2312" %>
```

```
<HTML>
<body bgcolor = cyan><FONT size = 4>
    <form  action = "" method = post name = form>
       <input type = "text" name = "girl">
       <input type = "submit" value = "确定" name = "submit">
    </form>
    <%   String textContent = request.getParameter("girl");
         if(textContent == null) {
            textContent = "0";
         }
         String []a = textContent.split("#");
         double sum = 0;
         try {
               for(String s:a) {
                  out.print(s + " ");
                  sum += Double.parseDouble(s);
               }
               out.print("<br>数字的和是" + sum);
         }
         catch(NumberFormatException e) {
            out.print("<BR>" + "请输入数字字符");
         }
     %>
</font></body></HTML>
```

注意:使用 request 对象获取当前页面提交的信息时要格外小心,在上面的例子 3_2 中,当用户在浏览器中输入页面地址请求页面时,用户还没有机会提交数据,那么页面在执行

```
String textContent = request.getParameter("girl");
```

时得到的 textContent 就是空对象。如果程序使用了空对象,Java 解释器就会提示出现了 NullPointerException 异常。因此,在上述例子 3_2 中,为了避免在运行时出现 NullPointerException 异常,使用了如下代码:

```
String textContent = request.getParameter("girl");
       if(textContent == null) {
          textContent = "";
       }
```

3.1.2 处理汉字信息

当用 request 对象获取用户提交的汉字字符时,会出现乱码问题,所以对含有汉字字符的信息必须采取特殊的处理方式——页面中使用 page 指令指定编码必须是 gb2312:

```
<%@ page contentType = "text/html; charset = gb2312" %>
```

“charset”中的首字母小写(c 为小写字母)。可以使用两种方式避免 request 对象获取的信息出现乱码,以下分别给予介绍。

1. 对信息重新编码

request 将获取的信息重新编码，即用 ISO-8859-1 进行编码，并将编码存放到一个字节数组中，然后再将这个数组转化为字符串。如下：

```
String str = request.getParameter("message");
byte  b[] = str.getBytes("ISO - 8859 - 1");
str = new String(b);
```

2. request 设置编码

request 在获取信息之前使用 setCharacterEncoding 方法设置自己的编码为 gb2312：

```
request.setCharacterEncoding("gb2312");
```

例子 3_3 使用上述的第 2 种方式避免乱码问题。在例子 3_3 中，example3_3.jsp 通过表单向自己提交一份通信费账单，然后计算出消费总额。例子 3_3 使用了 String 类的 String [] split(String regex)方法，该方法可以用参数 regex 指定的正则表达式作为分隔标记分解出当前字符串中的语言符号，例如可以用正则表达式"[0123456789.]+ "匹配所有的数字序列，用"[^0123456789.]+ "匹配所有的非数字序列。example3_3.jsp 效果如图 3.3 所示。

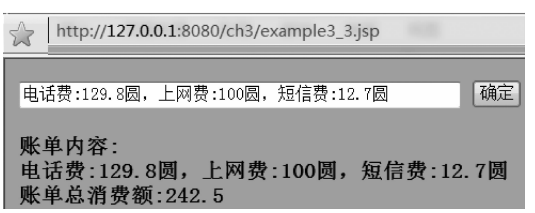


图 3.3　处理汉字信息

例子 3_3

example3_3.jsp（如图 3.3 所示）

```
<%@ page contentType = "text/html;charset = gb2312" %>
<HTML>
<body bgcolor = cyan><font size = 3>
    <form  action = "" method = post name = form>
       <input type = "text" name = "information_fees" size = 50>
       <input type = "submit" value = "确定" name = "submit">
    </form>
    <%   request.setCharacterEncoding("gb2312");
         String information_fees = request.getParameter("information_fees");
         double number = 0;
         if(information_fees == null) {
            information_fees = "";
         }
     %>  <b> 账单内容:<br><% = information_fees %><br>
     <%  String []a = information_fees.split("[^0123456789.]");
```

```
            double sum = 0;
            for(String s:a) {
               try {
                 sum += Double.parseDouble(s);
               }
               catch(NumberFormatException exp){}
            }
        %> <b> 账单总消费额:<% = sum %>
    </font></body></HTML>
```

3.1.3 常用方法举例

当用户访问一个页面时,会提交一个 HTTP 请求给 JSP 引擎,这个请求包括一个请求行、http 头和信息体,例如:

```
post/example3_1.jsp/HTTP.1.1
host: localhost: 8080
accept - encoding: gzip, deflate
```

其中,首行叫请求行,规定了向访问的页面请求提交信息的方式,如 post、get 等方式,以及请求的页面的名字和使用的通信协议。

第 2、3 行分别是两个头(Header):host 和 accept-encoding,称 host、accept-encoding 是头名字,而 localhost:8080 以及 gzip,deflate 分别是两个头的值。一个典型的请求通常包含很多的头,有些头是标准的,有些和特定的浏览器有关。

一个请求还包含信息体,即 HTML 标记组成的部分,可能包括各式各样用于提交信息的表单等,例如:

```
<form action = "" method = post name = form >
    < input type = "text" name = "boy">
    < input type = "submit" value = "" name = "submit">
</form >
```

尽管服务器非常关心用户提交的 HTTP 请求中表单的信息,如前面的例子 3_1 和例子 3_2 中使用 request 的 getParameter 方法获取表单提交的有关信息。但实际上,request 对象调用相关方法可以获取请求的许多细节信息。内置对象 request 常用方法如下:

(1) getProtocol() 获取用户向服务器提交信息所使用的通信协议,比如 http/1.1 等。

(2) getServletPath() 获取用户请求的 JSP 页面文件的目录。

(3) getContextPath()获取用户请求的当前 Web 服务目录。

(4) getContentLength() 获取用户提交的整个信息的长度。

(5) getMethod() 获取用户提交信息的方式,比如 post 或 get。

(6) getHeader(String s) 获取 HTTP 头文件中由参数 s 指定的头名字的值。一般来说参数 s 可取的头名有:accept、accept-language、content-type、accept-encoding、user-agent、host、content-length、connection、cookie 等。比如,s 取值 user-agent 将获取用户的浏览器的版本号等信息。

(7) getHeaderNames() 获取头名字的一个枚举。

(8) getHeaders(String s) 获取头文件中指定头名字的全部值的一个枚举。

(9) getRemoteAddr() 获取用户的 IP 地址。

(10) getRemoteHost() 获取用户机的名称(如果获取不到,就获取 IP 地址)。

(11) getServerName() 获取服务器的名称。

(12) getServerPort() 获取服务器的端口号。

(13) getParameterNames()获取用户提交的信息体部分中 name 参数值的一个枚举。

下面的例子 3_4 使用了 request 的一些常用方法,效果如图 3.4 所示。

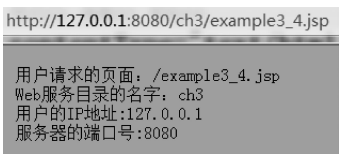


图 3.4　request 获取的信息

例子 3_4

example3_4.jsp(效果如图 3.4 所示)

```
<%@ page contentType = "text/html;charset = gb2312" %>
<%@ page import = "java.util.*" %>
<MHML><body bgcolor = cyan><font size = 2 >
    <% String path = request.getServletPath();          //请求的页面
       String webDir = request.getContextPath();        //获取当前 Web 服务目录
       webDir = webDir.substring(1);                    //去掉 Web 服务目录前面的目录符号"/"
       String clientIP = request.getRemoteAddr();       //用户的 IP 地址
       int serverPort = request.getServerPort();        // 服务器的端口号
     %>
  用户请求的页面: <% = path %>
<br>Web 服务目录的名字: <% = webDir %>
<br>用户的 IP 地址:<% = clientIP %>
<br>服务器的端口号:<% = serverPort %>
</font></body></HTML>
```

3.1.4　处理 HTML 标记

本节对经常用于提交信息的 HTML 标记做一简单介绍。有关细节建议读者查阅 HTML 资料。

JSP 页面可以含有 HTML 标记,当用户通过浏览器请求一个 JSP 页面时,Tomcat 服务器将该 JSP 页面中的 HTML 标记直接发送到用户的浏览器,由用户的浏览器负责执行这些 HTML 标记。而 JSP 页面中的变量声明、程序片以及表达式由 Tomcat 服务器处理后,再将有关的结果用文本方式发送到用户端的浏览器。

HTML 是 Hypertext Marked Language 的缩写,即超文本置标语言。目前的 HTML 大约有一百多个标记,这些标记可以描述数据的显示格式。如果读者对 HTML 语言比较陌生,建议补充这方面的知识。

1. <form>标记

由于用户经常需要使用表单提交数据,所以有必要对表单做一个简明的介绍。

表单的一般格式是:

```
<form action = "提交信息的目的地页面"  method = get| post  name = "表单的名字">
```

```
    数据提交手段部分
</form>
```

其中<form…>…</form>是表单标记,其中的 method 属性取值 get 或 post。get 方法和 post 方法的主要区别是:使用 get 方法提交的信息会在提交的过程中显示在浏览器的地址栏中,而用 post 方法提交的信息不会显示在地址栏中。提交手段包括:文本框、列表、文本区等,例如:

```
<form action = "tom.jsp" method = "post" >
        <input type = "text" name = "boy" value = "ok" >
        <input type = "submit" value = "送出" name = "submit">
</form>
```

表单标记经常将下列标记作为表单的子标记,以便提供提交数据的手段,这些标记都以 GUI 形式出现,方便用户输入或选择数据。比如,文本框、下拉列表、滚动列表等。

```
<input …>
<select …    ></select>
<option …>  </option>
<textArea …> </textArea>
```

2. <input>标记

在表单标记<form>中<input>标记作为子标记用来指定表单中数据的输入方式以及表单的 submit 按钮。<input>标记中的 type 属性可以指定输入方式的 GUI 对象,name 属性用来指定这个 GUI 对象的名称。<input>标记的基本格式是:

```
<input  type = "输入对象的 GUI 类型"  name = "名字">
```

服务器通过属性 name 指定的名字来获取“输入对象的 GUI 类型”中提交的数据。“输入对象的 GUI 类型”可以是 text(文本框)、checkbox(复选框)、submit(“提交”按钮)等。

(1) 文本框 text。当输入对象的 GUI 类型是 text 时,除了用 name 为 text 指定名字外,还可以为 text 指定其他的一些值。例如:

```
<input type = "text" name = "me"  value = "hi"  size = "12"  align = "left"  maxlength = "30">
```

其中,value 的值是 text 的初始值;size 是 text 对象的长度(单位是字符);align 是 text 在浏览器窗体中的对齐方式;maxlength 指定 text 可输入的最多字符数目。request 对象通过 name 指定的名字来获取用户在文本框输入的字符串。如果用户没有在文本框输入任何信息,就单击表单中的 submit 按钮,request 对象将调用 getPeremeter 方法获取由 value 指定的值;如果 value 未指定任何值,getPeremeter 方法获取的字符串的长度为 0,即该字符串为:""。

(2) 单选按钮 radio。当输入对象的 GUI 类型是 radio 时,除了用 name 为 radio 指定名字外,还可以为 radio 指定其他的一些值。例如:

```
<input type = "radio"  name = "rad"  value = "red"  align = "top"  checked = "java" >
```

其中,value 指定 radio 的值;align 是 radio 在浏览器窗体中的对齐方式;如果几个单选按

钮的 name 取值相同，那么同一时刻只能有一个被选中。request 对象调用 getPeremeter 方法获取被选中的 radio 中 value 属性指定的值。checked 如果取值是一个非空的字符串，那么该单选按钮的初始状态就是选中状态。

(3) 复选框 checkbox。当输入对象的 GUI 类型是 checkbox 时，除了用 name 为 checkbox 指定名字外，还可以为 checkbox 指定其他的一些值。例如：

```
<input type="checkbox"  name= "ch"  value="pink"  align= "top"  checked="java">
```

其中，value 指定 checkbox 的值；复选框与单选按钮的区别就是可以多选，即如果几个 checkbox 的 names 取值相同，那么同一时刻可有多个 checkbox 被选中。这时，request 对象需调用 getParameterValues 方法，获取被选中的多个 checkbox 中 value 属性指定的值。为了使服务器通过 getParameterValues 方法能获取提交的值，复选框 name 的值应互不相同。checked 如果取值是一个非空的字符串，那么该复选框的初始状态就是选中状态。

(4) 口令框 password。它是输入口令用的特殊文本框，输入的信息用“*”回显，防止他人偷看口令。

```
<input type= "password"  name= "me" size= "12"  maxlength="30">
```

服务器通过 name 指定的字符串获取 password 提交的值。比如，用户在口令框中输入“bird88_1”，那么 bird88_1 将被提交给服务器，口令框仅仅起不让别人偷看的作用，不提供加密措施。

(5) 隐藏 hidden。当＜input＞中的属性 type 的值是 hidden 时，＜input＞没有可见的输入界面，表单直接将＜input＞中 value 属性的值提交给服务器。例如：

```
<input type= "hidden"  name="h"  value= "123"  >
```

request 对象调用 getParemeter 方法，通过 name 的名字来获取由 value 指定的值。

(6) submit 按钮。为了能把表单的数据提交给服务器，一个表单至少要包含一个 submit 按钮。

```
<input type= "submit"  name="me"  value="确定"  size="12"  >
```

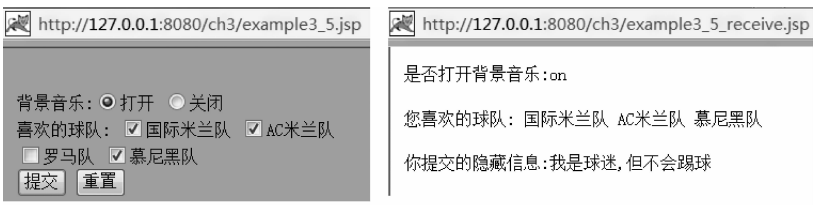
单击 submit 按钮后，服务器就可以获取表单提交的各个数据。当然 request 对象调用 getParemeter 方法也可以获取 submit 按钮的值，getParemeter 方法通过 name 指定的名字来获取 submit 按钮提交的由 value 指定的值。

(7) 重置按钮 reset。重置按钮将表单中输入的数据清空，以便重新输入数据。

```
<input type="reset" >
```

在下面的例子 3_5 中，JSP 页面 example3_5.jsp 用表单向 example3_5_receive.jsp 页面提交数据，example3_5_receive.jsp 页面使用 request 对象获得 example3_5.jsp 提交的数据。用户在 example3_5.jsp 页面单击表单的 submit 提交按钮提交信息，所提交的信息包括通过 radio 选择的是否打开背景音乐的信息、通过 checkbox 选择的球队信息、通过 hidden 隐藏的信息。调试例子 3_5 时，需要将名字是 back.mp3 的 mp3 文件存放到 Web 服务目录 ch3 的子目录 sound 中。example3_5.jsp 和 example3_5_receive.jsp 的效果如图 3.5(a)和图 3.5(b)所示。

注意：HTML 标记＜bgsound src＝"音乐文件" loop＝"整数值" /＞用于设置网页的背景音乐，音乐文件可以是.wav 和.mp3 格式。loop 的值取正整数表示背景音乐播放的次数，取－1 为无限循环播放。



(a) 使用表单提交数据　　(b) 用request获得数据

图 3.5　数据的提交与获取

例子 3_5

example3_5.jsp(效果如图 3.5(a)所示)

```
<%@ page contentType = "text/html;charset = gb2312" %>
<HTML><body bgcolor = cyan><font size = 2>
     <form action = "example3_5_receive.jsp" method = post name = form>
       <br>背景音乐:<input type = "radio" name = "R" value = "on" >打开
           <input type = "radio" name = "R" value = "off" checked = "default">关闭
       <br>喜欢的球队:
           <input type = "checkbox" name = "item" value = "国际米兰队" >国际米兰队
           <input type = "checkbox" name = "item" value = "AC 米兰队" > AC 米兰队
       <br><input type = "checkbox" name = "item" value = "罗马队" >罗马队
           <input type = "checkbox" name = "item" value = "慕尼黑队" >慕尼黑队
           <input type = "hidden" value = "我是球迷,但不会踢球" name = "secret">
       <br><input type = "submit" value = "提交" name = "submit">
           <input type = "reset" value = "重置" >
    </form>
</font></body></HTML>
```

example3_5_receive.jsp(效果如图 3.5(b)所示)

```
<%@ page contentType = "text/html;charset = gb2312" %>
<%! public String handleStr(String s) {
        try {   byte [] bb = s.getBytes("iso－8859－1");
                s = new String(bb);
        }
        catch(Exception exp){}
        return s;
     }
 %>
<HTML><body><font size = 2>
    <%
        String onOrOff = request.getParameter("R");                    //获取 radio 提交的值
        String secretMess = request.getParameter("secret");            //获取 hidden 提交的值
                                                                       //获取 checkbox 提交的值
      String itemName[] = request.getParameterValues("item");
        out.println("<p> 是否打开背景音乐:" + onOrOff);
```

```
        out.println("<p> 您喜欢的球队:");
        if(itemName == null) {
           out.print("一个都不喜欢");
        }
        else {
          for(int k = 0;k < itemName.length;k++) {
            out.println(" " + handleStr(itemName[k]));
          }
        }
        out.println("<P> 你提交的隐藏信息:" + handleStr(secretMess));
        if(onOrOff.equals("on")) {
    %>    <bgsound src = 'sound/back.mp3' loop = " - 1"/>
    <%  }
    %>
</font></body></HTML>
```

3. <select>、<option>标记

下拉式列表和滚动列表通过<select>和<option>标记来定义,经常作为<form>的子标记为表单提供选择数据的 GUI。<select>标记将<option>作为子标记,形成下拉列表或滚动列表。

下拉列表的基本格式是:

```
<select  name = "myName">
   <option  value = "item1">
   <option  value = "item2">
  …
</select>
```

在 select 中增加 size 属性的值就变成滚动列表,size 的值是滚动列表的可见行的数目。滚动列表的基本格式是:

```
< select name = "myName"  size = "正整数">
   < option  value = "item1">
   < option  value = "item2">
…
</select >
```

request 对象通过 name 获取滚动列表中被选中的 option 的值(参数 value 指定的值)。

在下面的例子 3_6 中,用户通过下拉列表为当前页面选择一首背景音乐,通过滚动列表为当前页面选择一幅图像。调试例子 3_6 时,需要将名字是 back1.mp3、back2.mp3 和 back3.mp3 的 mp3 文件存放到 Web 服务目录 ch3 的子目录 sound 中,需要将名字是 back1.jpg、back2.jpg 和 back3.jpg 的 jpg 文件存放到 Web 服务目录 ch3 的子目录 image 中。example3_6 的效果如图 3.6 所示。

例子 3_6

example3_6.jsp(效果如图 3.6 所示)

```
<%@ page contentType = "text/html;charset = gb2312" %>
<%! public String handleStr(String s) {
```

```
            try {   byte [] bb = s.getBytes("iso-8859-1");
                    s = new String(bb);
            }
            catch(Exception exp){}
            return s;
        }
%>
   <%    String music = request.getParameter("music");
         String pic = request.getParameter("pic");
         if(music == null) music = "";
         if(pic == null)   pic = "";
         music = handleStr(music);
         pic   = handleStr(pic);
   %>
<HTML><center>
    <body background = "image/<% = pic %>"><font size = 2 >  <!-- 背景图像 -->
    <bgsound src = "sound/<% = music %>" loop = -1/>       <!-- 背景音乐 -->
    <form action = "" method = post name = form>
    <b>选择背景音乐:<br>
        <select name = "music" >
           <Option selected value = "back1.mp3">绿岛小夜曲
           <Option value = "back2.mp3">我是一片云
           <Option value = "back3.mp3">红河谷
        </select>
    <br><b>选择背景图像:<br>
        <select name = "pic" size = 2>
           <option value = "back1.jpg">荷花图
           <option value = "back2.jpg">玫瑰图
           <option value = "back3.jpg">校园图
        </select> <br>
      <input type = "submit" value = "提交" name = "submit">
    </form>
</font></body></Center></HTML>
```

图 3.6 下拉列表和滚动列表

4. <textArea>标记

<textArea>是一个能输入或显示多行文本的文本区,在表单中使用<textArea>作为子标记可以提交多行文本给服务器。<textArea>的基本格式为:

```
<textArea  name = "名字"  rows = "文本可见行数"  cols = "文本可见列数"  >
</textArea>
```

5. ＜table＞标记

表格以行列形式显示数据，不提供数据输入功能。经常将某些数据或 GUI 放置在表格的单元格中，让界面更加简练、美观。

表格由＜table＞标记定义，一般格式是：

```
< table >
    < tr  width = "该行的宽度">
        < th  width = "单元格的宽度" >单元格中的数据</th >
          …
        < td  width = "单元格的宽度" >单元格中的数据</td > …
    </tr >
…
</table >
```

其中

```
< tr > …</tr >
```

定义表格的一个行，＜th＞或＜td＞标记定义这一行中的表格单元。二者的区别是＜th＞定义的单元着重显示，＜td＞称为普通单元，不着重显示。一行中的着重单元和普通的单元可以交替出现，也可以全是着重单元或普通单元。＜table ＞中增加选项 border 可指明该表格是否带有边框。

在例子 3_7 中，用户通过 example3_7. jsp 提供的表单输入表格的行数和列数，然后 example3_7. jsp 按照用户输入的行数和列数创建相应的表格。example3_7. jsp 效果如图 3. 7 所示。

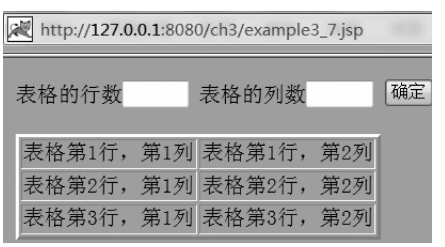


图 3. 7　表格

例子 3_7

example3_7. jsp(效果如图 3. 7 所示)

```
< % @ page contentType = "text/html;charset = gb2312" % >
< HTML >
< body bgcolor = cyan >< font size = 3 >
    < form  action = "" method = post name = form >
       表格的行数< input type = "text" name = "table_rows" size = 6 >
       表格的列数< input type = "text" name = "table_cols" size = 6 >
       < input type = "submit" value = "确定" name = "submit">
    </form >
  < %   String rows = request. getParameter("table_rows");
       String cols = request. getParameter("table_cols");
       if(cols == null || rows == null) {
          rows = cols = "0";
       }
       int m  = Integer. parseInt(rows);
       int n  = Integer. parseInt(cols);
  % >   < table border = 2 >
  < %   for(int i = 1;i < = m;i++) {
  % >      < tr >
```

```
    <%       for(int j=1;j<=n;j++) {
     %>        <td>表格第<% =i%>行,第<% =j%>列</td>
    <%       }
     %>       </tr>
    <%   }
     %>   </table>
</font></body></HTML>
```

6. <image>标记

使用<image>标记可以显示一幅图像,<image>标记的基本格式为:

```
<image  src="图像文件的URL">描述文字</image>
```

如果图像文件和当前页面在同一 Web 服务目录中,图像的文件的地址就是该图像文件的名字;如果图像文件在当前 Web 服务目录的一个子目录中,比如 image 子目录中,那么"图像文件的 URL"就是"image/图像文件的名字"。

<image>标记中可以使用 width 和 height 属性指定被显示的图像的宽为和高,如果省略 width 和 height 属性,<image>标记将按图像的原始宽度和高度来显示图像。

7. <embed>标记

使用<embed>标记可以播放音乐和视频,当浏览器执行该标记时,会把浏览器所在机器上的默认播放器嵌入到浏览器中,以便播放音乐或视频文件。<embed>标记的基本格式为:

```
<embed  src="音乐或视频文件的URL">描述文字</embed>
```

如果音乐或视频文件和当前页面在同一 Web 服务目录中,<embed>标记中 src 属性的值就是该文件的名字;如果视频文件在当前 Web 服务目录的一个子目录中,比如 avi 子目录中,那么<embed>标记中 src 属性的值就是"avi/视频文件的名字"。

< embed >标记中经常使用的属性及取值如下:

autostart 属性,取值"true"或"false",autostart 属性的值用来指定音乐或视频文件传送完毕后是否立刻播放。该属性的默认值是 false。

loop 属性,取值为正整数指定音乐或视频文件重复播放的次数,取值为-1 则无限循环播放。

width 和 height 属性,取值均为正整数,用 width 和 height 属性的值指定播放器的宽和高。如果省略 width 和 height 属性,将使用默认值。

下面的例子 3_8 中页面使用了<image>和<embed>标记。用户通过 example3_8.jsp 页面使用<image>标记显示一幅图像,使用下拉列表选择要播放视频;example3_8.jsp 页面使用<embed>标记播放用户选择的视频。其中图像文件和视频文件分别存放在当前 Web 服务目录 ch3 的子目录 image 和 avi 中。

例子 3_8

example3_8.jsp(效果如图 3.8 所示)

```
<%@ page contentType="text/html;charset=gb2312" %>
```

```
<%! public String handleStr(String s) {
        try {   byte [] bb = s.getBytes("iso-8859-1");
                s = new String(bb);
        }
        catch(Exception exp){}
        return s;
    }
%>
  <%   String video = request.getParameter("video");
       if(video == null)   video = "";
       video = handleStr(video);
  %>
<HTML><center>
    <form action = "" method = post name = form>
    <b>选择视频:<br>
        <select name = video >
           <option value = "我的祖国.avi">我的祖国
           <option value = "梦幻.avi">梦幻
           <option value = "夕阳山顶.avi" selected>夕阳山顶
        </select>
       <input type = "submit" value = "提交" name = "submit">
    </form>
    <image src = "image/flower.jpg" width = 120 height = 90></image><!-- 图像 -->
    <embed src = "avi/<% = video %>" width = 300 height = 180 >视频</embed><!-- 视频 -->
</font></body></Center></HTML>
```

图 3.8　显示图像与播放视频

3.1.5　处理超链接

HTML 的超链接标记

文字说明

是一个常用标记。用户单击超链接标记的“文字说明”,可以访问超链接给出的链接页面。

使用超链接标记还可以增加参数以及参数的值,以便向所链接的页面传递值,格式如下:

文字说明

超链接所链接的页面,使用 request(参数 n)获得超链接传递过来的值。需要注意的是,<a>标记向所链接的页面传递串值时,串值中不能含有汉字字符(否则会出现乱码问题)。

在下面的例子 3_9 中,example3_9.jsp 页面使用超链接向 example3_9_receive.jsp 页面传递商品的编号和价格。example3_9.jsp 和 example3_9_receive.jsp 的效果如图 3.9(a)和图 3.9(b)所示。

(a) 使用超链接　　(b) 获取超链接传递的值

图 3.9　使用超链接的效果

例子 3_9

example3_9.jsp(效果如图 3.9(a)所示)

```
<%@ page contentType = "text/html;charset = gb2312" %>
<HTML><body bgcolor = pink>
    商品编号 A1001,价格 8765
    <a href = "example3_9_receive.jsp?id = A1001&price = 8765" >
      购买
    </a>
</body></HTML>
```

example3_9_receive.jsp(效果如图 3.9(b)所示)

```
<%@ page contentType = "text/html;charset = gb2312" %>
<HTML><body bgcolor = #EEEEFF>
   <%   String id = request.getParameter("id");
        String price = request.getParameter("price");
   %>
   <b>商品编号:<% = id %><br>
      商品价格:<% = price %>
</body></HTML>
```

3.2　response 对象

当用户访问一个服务器的页面时,会提交一个 HTTP 请求,服务器收到请求时,返回 HTTP 响应。响应和请求类似,也有某种结构,每个响应都由状态行开始,可以包含几个头及可能的信息体(网页的结果输出部分)。

3.1 节学习了用 request 对象获取用户请求提交的信息,与 request 对象相对应的对象是 response 对象。可以用 response 对象对用户的请求作出动态响应,向用户端发送数据。比如,当一个用户请求访问一个 JSP 页面时,该页面用 page 指令设置页面的 contentType 属性的值是 text/html,那么 JSP 引擎将按照这种属性值响应用户对页面的请求,将页面的

静态部分返回给用户，用户浏览器接收到该响应就会使用 HTML 解释器解释执行所收到的信息。

3.2.1　动态响应 contentType 属性

页面用 page 指令设置页面的 contentType 属性的值，那么 JSP 引擎将按照这种属性值作出响应，将页面的静态部分返回给用户，用户浏览器接收到该响应就会使用相应的手段处理所收到的信息。由于 page 指令只能为 contentType 指定一个值来决定响应的 MIME 类型，如果想动态地改变这个属性的值来响应用户，就需要使用 response 对象的 setContentType(String s)方法来改变 contentType 的属性值。该方法中的参数 s 可取值有：text/html、text/plain、image/gif、image/x-xbitmap、image/jpeg、image/pjpeg、application/x-shockwave-flash、application/vnd.ms-powerpoint、application/vnd.ms-excel、application/msword 等。

当用 setContentType 方法动态改变了 contentType 的属性值，即响应的 MIME 类型，JSP 引擎就会按照新的 MIME 类型将 JSP 页面的输出结果返回给用户。

在下面的例子 3_10 中，当用户单击按钮，选择将当前页面保存为一个 Word 文挡时，JSP 页面动态地改变 contentType 的属性值为 application/msword。这时，用户的浏览器会提示用户启用 Microsoft Word 来显示或保存当前页面。

例子 3_10

example3_10.jsp

```
<%@ page contentType = "text/html;charset = gb2312" %>
<HTML><body bgcolor = cyan><font size = 3>
  <p>我正在学习 response 对象的
  <br> setContentType 方法
  <p>将当前页面保存为 word 文档吗?
   <form action = "" method = "get" name = form>
      <input type = "submit" value = "yes" name = "submit">
    </form>
  <% String str = request.getParameter("submit");
     if(str == null) {
        str = "";
     }
     if(str.equals("yes")) {
        response.setContentType("application/msword;charset = gb2312");
     }
  %>
</font></body></HTML>
```

3.2.2　response 的 HTTP 文件头

当用户访问一个页面时，会提交一个 HTTP 请求给 JSP 引擎，这个请求包括一个请求行、http 头和信息体，例如：

```
post/example3_1.jsp/HTTP.1.1
host: localhost: 8080
accept - encoding: gzip, deflate
```

其中首行叫请求行,规定了向访问的页面提交请求信息的方式,如 post、get 等,以及请求的页面的名字和使用的通信协议。

第 2、3 行分别是两个头(Header):host 和 accept-encoding,称 host、accept-encoding 是头名字,而 localhost:8080 以及 gzip,deflate 分别是两个头的值。一个典型的请求通常包含很多的头,有些头是标准的,有些和特定的浏览器有关。

同样,响应也包括一些头。response 对象可以使用方法

```
addHeader(String head,String value);
```

或

```
setHeader(String head ,String value)
```

动态添加新的响应头和头的值,将这些头发送给用户的浏览器。如果添加的头已经存在,则先前的头被覆盖。

在下面的例子 3_11 中,response 对象添加一个响应头“Refresh”,其头值是“5”。那么用户收到这个头之后,5 秒钟后将再次刷新该页面,导致该网页每 5 秒刷新一次。

例子 3_11

example3_11.jsp

```
<%@ page contentType = "text/html;charset = gb2312" %>
<%@ page import = "java.util. * " %>
<HTML><body bgcolor = cyan>
<p>现在的时间是: <br>
<%    out.println("" + new Date());
       response.setHeader("Refresh","5");
  %>
</body></HTML>
```

3.2.3 response 重定向

在某些情况下响应用户时,需要将用户重新引导至另一个页面。例如,如果用户输入的表单信息不完整,就会再被引导到该表单的输入页面。

可以使用 response 的 sendRedirect(URL url)方法实现用户的重定向。

在下面的例子 3_12 中,用户在 example3_12.jsp 页面的表单中输入姓名提交给 example3_12_receive.jsp 页面,如果未输入姓名就提交表单就会重新定向到 example3_12.jsp 页面。

例子 3_12

example3_12.jsp

```
<%@ page contentType = "text/html;charset = gb2312" %>
<HTML><body bgcolor = yellow>
```

```
<p>填写姓名:<br>
    <form action="example3_12_receive.jsp" method="post" name=form>
      <input type="text"   name="name">
      <input type="submit" value="确定">
    </form>
</body></HTML>
```

example3_12_receive.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<HTML><body  bgcolor=#DDEEFF>
  <% String name=request.getParameter("name");
       if(name==null||name.length()==0) {
          response.sendRedirect("example3_12.jsp");
       }
       byte[] b = name.getBytes("iso-8859-1");
       name = new String(b);
   %>
     <b>欢迎<% = name %>来到本网页。
</body></HTML>
```

3.2.4　response 的状态行

当服务器对用户请求进行响应时,它发送的首行称为状态行。

状态行包括 3 位数字的状态代码和对状态代码的描述(称作原因短语)。下面列出 5 类状态码及其简要描述:

- 1yy(1 开头的 3 位数):主要是实验性质的。
- 2yy:用来表明请求成功,例如,状态代码 200 可以表明已成功取得了请求的页面。
- 3yy:用来表明在请求满足之前应采取进一步的行动。
- 4yy:当浏览器给出无法满足的请求时,返回该状态代码,例如 404 表示请求的页面不存在。
- 5yy:用来表示服务器出现问题。例如,500 说明服务器内部发生错误。

一般不需要修改状态行,在出现问题时,服务器会自动响应,发送相应的状态码。我们也可以使用 response 对象的 setStatus(int n)方法来改变响应的状态行的内容。在下面的例子 3_13 中,使用 setStatus(int n)方法设置响应的状态行。

例子 3_13

example3_13.jsp(效果如图 3.10(a)所示)

```
<%@ page contentType="text/html;charset=gb2312" %>
<HTML><body bgcolor=cyan><font size=2>
<b>点击超链接看页面是否能响应用户:
<br> <A HREF="example3_13_bird.jsp">去看看是否欢迎您
</font></body></HTML>
```

example3_13_bird.jsp(效果如图 3.10(b)所示)

```
<HTML><body>
```

```
<% response.setStatus(408);
 %>
<b> "设置响应是 408,所以不显示这句话";<! -- 不能再发送到用户端了 -->
</body></HTML
```

(a) 单击超链接　　(b) 状态码是408的情况

图 3.10　设置响应状态行的效果

表 3.1 是状态代码表。

表 3.1　状态代码表

状 态 代 码	代 码 说 明
101	服务器正在升级协议
100	用户可以继续
201	请求成功且在服务器上创建了新的资源
202	请求已被接受但还没有处理完毕
200	请求成功
203	用户端给出的元信息不是发自服务器的
204	请求成功,但没有新信息
205	用户必须重置文档视图
206	服务器执行了部分 get 请求
300	请求的资源有多种表示法
301	资源已经被永久移动到新位置
302	资源已经被临时移动到新位置
303	应答可以在另外一个 URL 中找到
304	get 方式请求不可用
305	请求必须通过代理来访问
400	请求有语法错误
401	请求需要 HTTP 认证
403	取得了请求但拒绝服务
404	请求的资源不可用
405	请求所用的方法是不允许的
406	请求的资源只能用请求不能接受的内容特性来响应
407	用户必须得到认证
408	请求超时
409	发生冲突,请求不能完成
410	请求的资源已经不可用
411	请求需要一个定义的内容长度才能处理
413	请求太大,被拒绝
414	请求的 URL 太大
415	请求的格式被拒绝

续表

状 态 代 码	代 码 说 明
500	服务器发生内部错误,不能服务
501	不支持请求的部分功能
502	从代理和网关接受了不合法的字符
503	HTTP 服务暂时不可用
504	服务器在等待代理服务器应答时发生超时
505	不支持请求的 HTTP 版本

3.3　session 对象

HTTP 协议是一种无状态协议。一个用户向服务器发出请求(request),然后服务器返回响应(response),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用户。当一个用户访问一个 Web 服务目录时,可能会在这个服务目录的几个页面反复连接、反复刷新一个页面或不断地向一个页面提交信息等,服务器应当通过某种办法知道这是同一个用户。Tomcat 服务器可以使用内置 session 对象(会话)记录有关连接的信息。内置对象 session 由 Tomcat 服务器负责创建,session 是实现了 HttpSession 接口类的一个实例,可以在 Toamcat 服务器的 webapps\tomcat-docs\servletapi 中查找 HttpSession 接口的方法。

3.3.1　session 对象的 id

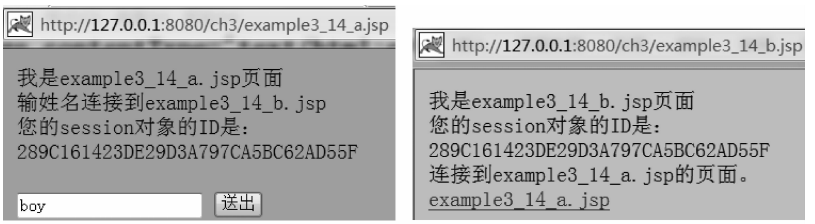
当一个用户首次访问 Web 服务目录中的一个 JSP 页面时,Tomcat 服务器产生一个 session 对象,这个 session 对象调用相应的方法可以存储用户在访问该 Web 服务目录中各个页面期间提交的各种信息,比如姓名、性别等信息。这个 session 对象被分配了一个 String 类型的 id 号,Tomcat 服务器同时将这个 id 号发送到用户端,存放在用户的 Cookie 中。这样,session 对象和用户之间就建立起一一对应的关系,即每个用户都对应着一个 session 对象(称作用户的会话),不同用户的 session 对象互不相同,具有不同的 id 号码。当用户再访问连接该 Web 服务目录的其他页面时,或从该 Web 服务目录连接到其他 Web 服务器再回到该 Web 服务目录时,Tomcat 服务器不再分配给用户的新 session 对象,而是使用完全相同的一个,直到 session 对象达到了最大生存时间或服务器关闭,服务器取消用户的 session 对象,即和用户的会话对应关系消失。当用户重新连接到该 Web 服务目录时,服务器为该用户再创建一个新的 session 对象。

简单地说,用户在访问一个 Web 服务目录期间,服务器为该用户分配一个 session 对象(称作用户的会话),服务器可以在各个页面使用这个 session 记录当前用户的有关信息。服务器保证不同用户的 session 对象互不相同。

注意:同一个用户在不同的 Web 服务目录中的 session 是互不相同的。

在下面的例子 3_14 中,用户在服务器的某个 Web 服务目录中的两个页面 example3_

14_a.jsp 和 example3_14_b.jsp 进行连接,两个页面的 session 对象是完全相同的。example3_14_a.jsp 和 example3_14_b.jsp 效果如图 3.11(a)和 3.11(b)所示。



(a) example3_14_a中session对象的ID　　(b) example3_14_b中session对象的ID

图 3.11　session 对象的 ID

例子 3_14

example3_14_a.jsp(效果如图 3.11(a)所示)

```
<%@ page contentType = "text/html;charset = gb2312" %>
<HTML><body  bgcolor = cyan>
    我是 example3_14_a.jsp 页面<br>输姓名连接到 example3_14_b.jsp
    <% String id = session.getId();
       out.println("<br>您的 session 对象的 ID 是:<br>" + id);
    %>
   <form action = "example3_14_b.jsp" method = post name = form>
        <input type = "text" name = "boy">
        <input type = "submit" value = "送出" name = submit>
   </form>
</body></HTML>
```

example3_14_b.jsp(效果如图 3.11(b)所示)

```
<%@ page contentType = "text/html;charset = gb2312" %>
<HTML><body  bgcolor = #EECCFF>
    我是 example3_14_b.jsp 页面
    <%  String id = session.getId();
        out.println("<br>您的 session 对象的 ID 是:<br>" + id);
    %>
<BR> 连接到 example3_14_a.jsp 的页面。<br>
<a href = "example3_14_a.jsp"> example3_14_a.jsp </A>
</body></HTML>
```

3.3.2　session 对象与 URL 重写

session 对象是否能和用户建立起一一对应关系依赖于用户端是否支持 Cookie。如果用户端不支持 Cookie,那么用户在不同网页之间的 session 对象可能是互不相同的,因为服务器无法将 id 存放到用户端,就不能建立 session 对象和用户的一一对应关系。用户将 Cookie 设置为禁止后(选择浏览器菜单→工具→Internet 选项→隐私,将第 3 方 Cookie 设置成禁止),运行上述的例子 3_13 会得到不同的结果。也就是说,同一用户对应了多个 session 对象,这样服务器就无法知道在这些页面上访问的实际上是同一个用户。

如果用户端不支持 Cookie,JSP 页面可以通过 URL 重写来实现 session 对象的唯一性。所谓 URL 重写,就是当用户从一个页面重新连接到一个页面时,通过向这个新的 URL 添加参数,把 session 对象的 id 传带过去,这样就可以保障用户在该网站各个页面中的 session 对象是完全相同的。可以使用 response 对象调用 encodeURL()或 encodeRedirectURL()方法实现 URL 重写。比如,如果从 first.jsp 页面连接到 second.jsp 页面,首先在程序片中实现 URL 重写:

```
String str = response.encodeRedirectURL("second.jsp");
```

然后将连接目标写成<%= str %>即可。

3.3.3　session 对象存储数据

session 对象驻留在服务器端,该对象调用某些方法保存用户在访问某个 Web 服务目录期间的有关数据。session 对象使用下列方法处理数据。

(1) public void setAttribute (String key,Object obj)。session 对象可以调用该方法将参数 Object 指定的对象 obj 添加到 session 对象中,并为添加的对象指定一个索引关键字。如果添加的两个对象的关键字相同,则先前添加的对象被清除。

(2) public Object getAttribute(String key)。获取 session 对象索引关键字是 key 的对象。由于任何对象都可以添加到 session 对象中,因此用该方法取回对象时,应强制转化为原来的类型。

(3) public Enumeration getAttributeNames()。session 对象调用该方法产生一个枚举对象,该枚举对象使用 nextElemets()遍历 session 中的各个对象所对应的关键字。

(4) public void removeAttribute(String name) session 对象调用该方法移除关键字 key 对应的对象。

当 session 对象处理数据时,非常类似一个购物车,一个用户访问一个商场(类似一个 Web 服务目录)期间,可以把商品放入自己购物车,也可以从自己的购物车移出商品。

在下面的例子 3_15 中,我们用 session 对象模拟购物车、存储用户的姓名和购买的书籍,三个 JSP 页面都保存在 Web 服务目录 ch3 中。

例子 3_15

example3_15_a.jsp(如图 3.12(a)所示)

```
<%@ page contentType = "text/html;charset = gb2312" %>
<head>
   <br>输入姓名:<a href = "example3_15_a.jsp">确定姓名页面</a>
   <br>选择图书:<a href = "example3_15_b.jsp">选择图书页面</a>
   <br>结账:      <a href = "example3_15_c.jsp">结账页面</a>
</head>
<HTML><body bgcolor = cyan><font size = 3>
   <p>输入姓名
    <form action = "" method = post name = form>
        <input type = "text" name = "name">
        <input type = "submit" value = "确定" name = submit>
```

```
    </form>
    <%   String name = request.getParameter("name");
         if(name == null)
            name = "";
         else
            session.setAttribute("name",name);                    //将名字存入用户的 session 中
    %>
</font></body></HTML>
```

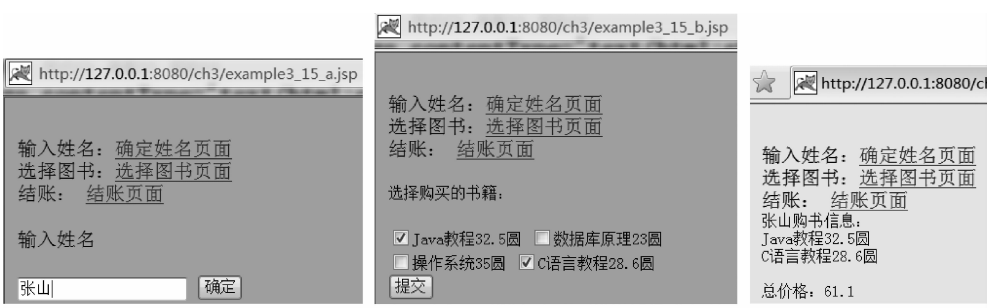
example3_15_b.jsp(效果如图 3.12(b)所示)

```
<%@ page contentType = "text/html;charset = gb2312" %>
<head>
   <br>输入姓名:<a href = "example3_15_a.jsp">确定姓名页面</a>
   <br>选择图书:<a href = "example3_15_b.jsp">选择图书页面</a>
   <br>结账:      <a href = "example3_15_c.jsp">结账页面</a>
</head>
<HTML><body bgcolor = cyan><font size = 2>
  <p>选择购买的书籍:
    <form action = "" method = post name = form>
      <input type = "checkbox" name = "choice" value = "Java 教程 32.5 圆" >Java 教程 32.5 圆
      <input type = "checkbox" name = "choice" value = "数据库原理 23 圆" >数据库原理 23 圆
      <br><input type = "checkbox" name = "choice" value = "操作系统 35 圆" >操作系统 35 圆
      <input type = "checkbox" name = "choice" value = "C 语言教程 28.6 圆" >C 语言教程 28.6 圆
      <br><input type = "submit" value = "提交" name = "submit">
   </form>
   <%   String book[] = request.getParameterValues("choice");
        if(book!= null) {
           StringBuffer str = new StringBuffer();
           for(int k = 0;k < book.length;k++) {
              str.append(book[k] + "<br>");
           }
           session.setAttribute("book",str);                      //将书籍放入用户的 session 中
        }
   %>
  </font></body></HTML>
```

example3_15_c.jsp(效果如图 3.12(c)所示)

```
<%@ page contentType = "text/html;charset = gb2312" %>
<%@ page import = "java.util.*" %>
<head>
   <br>输入姓名:<a href = "example3_15_a.jsp">确定姓名页面</a>
   <br>选择图书:<a href = "example3_15_b.jsp">选择图书页面</a>
   <br>结账:      <a href = "example3_15_c.jsp">结账页面</a>
</head>
<%! public String handleStr(String s) {
        try {   byte [] bb = s.getBytes("iso-8859-1");
                s = new String(bb);
        }
        catch(Exception exp){}
```

```
            return s;
        }
    %>
<HTML><body bgcolor = #EEEEFF><font size = 2>
<%   String personName = (String)session.getAttribute("name");
     StringBuffer bookMess = null;
     if(personName == null || personName.length() == 0) {
       out.println("到输入名字页面输入姓名");
     }
     else {
        bookMess = (StringBuffer)session.getAttribute("book");
     }
  %>
  <%   String buyBook = new String(bookMess);
       double sum = 0;
       String[] price = buyBook.split("[^0123456789.]");    //分解出价格
       if(price!= null) {
         for(String item:price)
           try {   sum += Double.parseDouble(item);
           }
           catch(NumberFormatException exp){}
       }
  %><br><% = handleStr(personName) %>购书信息：<br>
    <% = handleStr(buyBook) %> <br>
    总价格：<% = sum %>
</FONT></body></HTML>
```



(a) 输入姓名　　(b) 选择书籍　　(c) 结账

图 3.12　例子 3_15 效果

下面的例子 3_16 是一个猜数字的小游戏。当用户访问服务器上的 example3_16_number.jsp 时，随机分配给用户一个 1～100 之间的整数，然后将这个整数存在用户的 session 对象中。用户在表单里输入自己的猜测。用户输入的猜测提交给 example3_16_result.jsp，该页面负责判断用户给出的猜测是否和用户的 session 对象中存放的那个整数相同。如果相同就连接到 example3_16_success.jsp；如果不相同就连接到 example3_16_large.jsp 或 example3_16_small.jsp，然后，用户在这些页面再重新提交猜测到 example3_16_result 页面。

例子 3_16

example3_16_number. jsp(效果如图 3.13(a)所示)

```
<%@ page contentType = "text/html;charset = gb2312" %>
<HTML><body bgcolor = #EEDDEE><font Size = 2>
<p>随机分给了一个 1 到 100 之间的数,请猜!
   <%   int number = (int)(Math.random() * 100) + 1;
        session.setAttribute("count",new Integer(0));
        session.setAttribute("save",new Integer(number));
   %>
<br>输入猜测:
    <form action = "example3_16_result.jsp" method = "post" name = form>
        <input type = "text" name = "guess" >
        <input type = "submit" value = "送出" name = "submit">
    </form>
</font></body></HTML>
```

example3_16_large. jsp(效果如图 3.13(b)所示)

```
<%@ page contentType = "text/html;charset = gb2312" %>
<HTML><body bgcolor = yellow><font size = 2>
<%   Integer integer = (Integer)session.getAttribute("guess");
 %>
<p><% = integer %>数大了,请再猜:
    <form action = "example3_16_result.jsp" method = "post" name = form >
        <input type = "text" name = "guess" >
        <input type = "submit" value = "送出" name = "submit">
    </form>
</font></body></HTML>
```

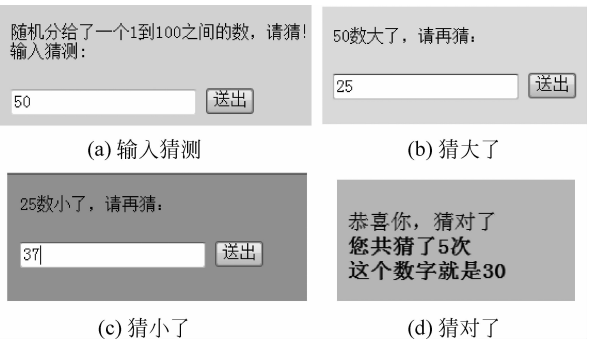
example3_16_small. jsp(效果如图 3.13(c)所示)

```
<%@ page contentType = "text/html;charset = gb2312" %>
<HTML><body bgcolor = #FF99EE><font size = 2>
<%   Integer integer = (Integer)session.getAttribute("guess");
 %>
<p><% = integer %>数小了,请再猜:
    <form action = "example3_16_result.jsp" method = "post" name = form >
        <input type = "text" name = "guess" >
        <input type = "submit" value = "送出" name = "submit">
    </form>
</font></body></HTML>
```

example3_16_success. jsp(效果如图 3.13(d)所示)

```
<%@ page contentType = "text/html;charset = gb2312" %>
<HTML><body bgcolor = pink><font size = 3>
<%   int count = ((Integer)session.getAttribute("count")).intValue();
     int num = ((Integer)session.getAttribute("save")).intValue();
 %>
<br>恭喜你,猜对了
<br><b>您共猜了<% = count %>次
```

```
<br>这个数字就是<% = num %>
</font></body></HTML>
```



(a) 输入猜测　(b) 猜大了　(c) 猜小了　(d) 猜对了

图 3.13　猜数游戏效果

example3_16_result.jsp

```
<%    String str = request.getParameter("guess");
      if(str == null || str.length() == 0) {
          response.sendRedirect("example3_16_number.jsp");
      }
      else {
         int guessNumber = Integer.parseInt(str);
         session.setAttribute("guess",new Integer(guessNumber));
         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("example3_16_success.jsp");
         }
         else if(guessNumber > realnumber){
             int n = ((Integer)session.getAttribute("count")).intValue();
             n = n + 1;
             session.setAttribute("count",new Integer(n));
             response.sendRedirect("example3_16_large.jsp");
         }
         else if(guessNumber < realnumber) {
             int n = ((Integer)session.getAttribute("count")).intValue();
             n = n + 1;
             session.setAttribute("count",new Integer(n));
             response.sendRedirect("example3_16_small.jsp");
         }
      }
%>
```

3.3.4　session 对象的生存期限

一个用户在某个 Web 服务目录的 session 对象的生存期限依赖于 session 对象是否调

用 invalidate()方法使得 session 无效或 session 对象达到了设置的最长的“发呆”状态时间以及是否关闭服务器。如果关闭服务器,那么用户的 session 消失;所谓“发呆”状态时间是指用户对某个 Web 服务目录发出的两次请求之间的间隔时间(默认的“发呆”时间是 30 分钟)。比如,用户对某个 Web 服务目录下的 JSP 页面发出请求并得到响应,如果用户不再对该 Web 服务目录发出请求(可能去请求其他的 Web 服务目录),那么用户对该 Web 服务目录进入“发呆”状态,直到用户再次请求该 Web 服务目录时,“发呆”状态结束。

可以修改 Tomcat 服务器下的 web.xml,重新设置各个 Web 服务目录下的 session 对象的最长“发呆”时间。打开 Tomcat 安装目录中 conf 文件下的配置文件 web.xml,找到

```
<session-config>
        <session-timeout>30</session-timeout>
</session-config>
```

将其中的 30 修改成所要求的值即可(单位为分钟)。

session 对象可以使用下列方法获取或设置与生存时间有关的信息。

(1) public long getCreationTime()　获取 session 创建的时间,单位是毫秒(GMT 时间,1970 年 7 月 1 日午夜起至 session 创建时刻所走过的毫秒数)。

(2) public long getLastAccessedTime()　获取 session 最后一次被操作的时间,单位是毫秒。

(3) public int getMaxInactiveInterval()　获取 session 最长的“发呆”时间(单位是秒)。

(4) public void setMaxInactiveInterval(int interval)　设置 session 最长的“发呆”时间(单位是秒)。

(5) public boolean isNew()　判断 session 是否是一个新建的对象。

(6) invalidate()　使 session 无效。

在下面的例子 3_17 中,session 对象使用 setMaxInactiveInterval(int interval)方法设置最长的“发呆”状态时间为 5 秒。用户可以通过刷新页面检查是否达到了最长的“发呆”时间。如果两次刷新之间的间隔超过 5 秒,用户先前的 session 将被取消,用户将获得一个新的 session 对象。

例子 3_17

example3_17.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import="java.util.*" %>
<HTML><body bgcolor=yellow>
  <%  session.setMaxInactiveInterval(5);
      boolean boo=session.isNew();
      out.println("<br>如果你第一次访问当前 Web 服务目录,您的会话是新的");
      out.println("<br>如果你不是首次访问当前 Web 服务目录,您的会话不是新的");
      out.println("<br>会话是新的吗?:"+boo);
      out.println("<br>欢迎来到本页面,您的 session 允许的最长发呆时间为"+
                  session.getMaxInactiveInterval()+"秒");
      out.println("<br>您的 session 的创建时间是"+new Date(session.getCreationTime()));
      out.println("<br>您的 session 的 Id 是"+session.getId());
      Long lastTime=(Long)session.getAttribute("lastTime");
```

```
        if(lastTime == null) {
            long n = session.getLastAccessedTime();
            session.setAttribute("lastTime",new Long(n));
        }
        else {
            long m = session.getLastAccessedTime();
            long n = ((Long)session.getAttribute("lastTime")).longValue();
            out.println("<br>您的发呆时间大约是" + (m - n) + "毫秒,大约" + (m - n)/1000 + "秒");
            session.setAttribute("lastTime",new Long(m));
        }
    }
  %>
</body></HTML>
```

3.4　application 对象

当用户第一次访问 Web 服务目录上的一个 JSP 页面时,JSP 引擎创建一个和该用户相对应的 session 对象,当用户在所访问的 Web 服务目录的各个页面之间浏览时,这个 session 对象都是同一个,而且不同用户的 session 对象是互不相同的。与 session 对象不同的是,application 对象由服务器负责创建,每个 Web 服务目录下的 application 对象被访问该服务目录的所有的用户所共享,但不同 Web 服务目录下的 application 互不相同。

3.4.1　application 对象的常用方法

(1) public void setAttribute(String key,Object obj)。application 对象可以调用该方法将参数 Object 指定的对象 obj 添加到 application 对象中,并为添加的对象指定一个索引关键字。如果添加的两个对象的关键字相同,则先前添加的对象被清除。

(2) public Object getAttribute(String key)。获取 application 对象含有的关键字是 key 的对象。由于任何对象都可以添加到 application 对象中,因此用该方法取回对象时,应强制转化为原来的类型。

(3) public Enumeration getAttributeNames()。application 对象调用该方法产生一个枚举对象,该枚举对象使用 nextElemets()遍历 application 中的各个对象所对应的关键字。

(4) public void removeAttribue(String key)。从当前 application 对象中删除关键字是 key 的对象。

(5) public String getServletInfo()。获取 Servlet 编译器的当前版本的信息。

由于 application 对象对所有的用户都是相同的,任何用户对该对象中存储的数据的改变都会影响到其他用户,因此在某些情况下,对该对象的操作需要实现同步处理。

注意:有些服务器不直接支持使用 application 对象,必须用 ServletContext 类声明这个对象,再使用 getServletContext()方法对这个 application 对象进行初始化。

3.4.2　用 application 制作留言板

在例子 3_18 中,用户通过 example3_18_input.jsp 向 example3_18_pane.jsp 页面提交

姓名、留言标题和留言内容,example3_18_pane.jsp 页面获取这些内容后,用同步方法将这些内容添加到一个向量中,然后将这个向量再添加到 application 对象中。当用户单击查看留言版时,example3_18_show.jsp 负责显示所有用户的留言内容,即从 application 对象中取出向量,然后遍历向量中存储的信息。

在这里使用了“向量”这种数据结构,Java 的 java.util 包中的 Vector<V>泛型类负责创建一个向量对象。如果你已经学会使用数组,那么使用向量就会很容易。当我们创建一个向量时不用像数组那样必须要给出数组的大小。向量创建后,例如,对于“Vector<String> a=new Vector<String>();”可以使用 add(V o)方法把 String 对象 a 添加到向量的末尾,向量的大小会自动增加。

例子 3_18

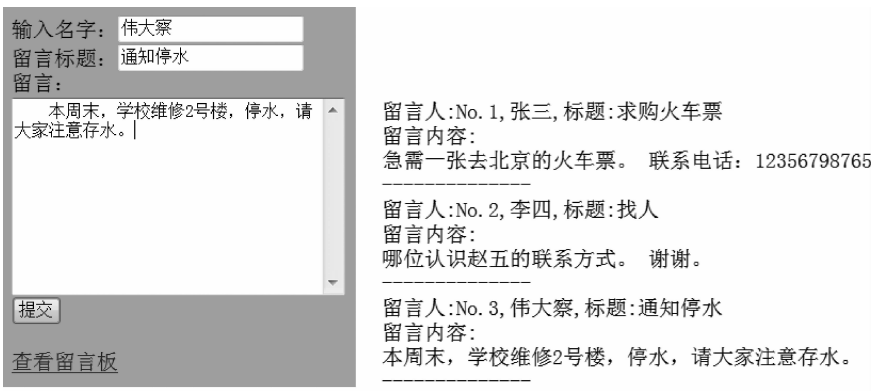
example3_18_input.jsp(如图 3.14(a)所示)

```
<%@ page contentType="text/html;charset=gb2312" %>
<HTML><body>
  <form action="messagePane.jsp" method="post" name="form">
      输入您的名字:<BR><input  type="text" name="peopleName">
      <BR>输入您的留言标题:<BR><input  type="text"  name="Title">
      <BR>输入您的留言:<BR><TEXTAREA name="messages" ROWs="10" COLS=36
WRAP="physical"></TEXTAREA>
      <BR><input type="submit" value="提交信息" name="submit">
  </form>
  <form action="showMessage.jsp" method="post" name="form1">
     <input type="submit" value="查看留言板" name="look">
  </form>
</body></HTML>
```

example3_18_show.jsp(如图 3.14(b)所示)

```
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import="java.util.*" %>
<%! public String handleStr(String s) {
        try {  byte[] bb= s.getBytes("iso-8859-1");
               s = new String(bb);
        }
        catch(Exception exp){}
        return s;
    }
%>
<HTML><body>
   <% Vector v=(Vector)application.getAttribute("Mess");
      for(int i=0;i<v.size();i++) {
         String message=(String)v.elementAt(i);
         String[]a =message.split("#");
         out.print("留言人:"+handleStr(a[0])+",");
         out.print("标题:"+handleStr(a[1])+"<br>");
         out.print("留言内容:<br>"+handleStr(a[2]));
         out.print("<br>---------------<br>");
      }
```

```
    %>
</body></HTML>
```



(a) 输入留言　　(b) 查看留言板

图 3.14　留言板效果

example3_18_pane. jsp

```
<%@ page contentType = "text/html;charset = gb2312" %>
<%@ page import = "java.util. *" %>
<HTML><body>
      <%! Vector v = new Vector();
          int i = 0;
          ServletContext  application;
          synchronized void leaveWord(String s) {            //留言方法
             application = getServletContext();;
             i++;
             v.add("No." + i + "," + s);
             application.setAttribute("Mess",v);
          }
       %>
       <% String name = request.getParameter("peopleName");
          String title = request.getParameter("title");
          String messages = request.getParameter("messages");
             if(name == null)
                name = "guest" + (int)(Math.random() * 10000);
             if(title == null)
                title = "无标题";
             if(messages == null)
                messages = "无信息";
             String s = name + "#" + title + "#" + messages;
             leaveWord(s);
             out.print("您的信息已经提交!");
       %>
   <a href = "example3_18_input.jsp" >返回留言页面
</body></HTML>
```

3.5 out 对象

out 对象是一个输出流,用来向用户端输出数据。在前面的许多例子里曾多次使用 out 对象进行数据的输出。out 对象可调用如下的方法用于各种数据的输出,例如:

(1) out.print(Boolean)或 out.println(boolean) 用于输出一个布尔值。

(2) out.print(char)或 out.println(char)输出一个字符。

(3) out.print(double)或 out.println(double)输出一个双精度的浮点数。

(4) out.print(fload)或 out.println(float) 用于输出一个单精度的浮点数。

(5) out.print(long)或 out.println(long)输出一个长整型数据。

(6) out.print(String)或 out.println(String) 输出一个字符串对象的内容。

(7) out.newLine()输出一个换行符。

(8) out.flush()输出缓冲区里的内容。

(9) out.close()关闭流。

方法 println 和 print 的区别是:println 会向缓存区写入一个换行,而 print 不写入换行。但是浏览器的显示区域目前不识别 println 写入的换行。如果希望浏览器显示换行,应当向浏览器写入
实现换行。

3.6 上机实验

3.6.1 实验 3_1 request 对象

1. 相关知识点

HTTP 通信协议是用户与服务器之间一种请求与响应(request/response)的通信协议。在 JSP 中,内置对象 request 封装了用户请求信息时所提交的信息,那么该对象调用相应的方法可以获取封装的信息,即使用该对象可以获取用户提交的信息。

2. 实验目的

本实验的目的是让学生掌握怎样在 JSP 中使用内置对象 request。

3. 实验要求

编写一个 JSP 页面 input.jsp,该页面提供一个表单,用户可以通过表单输入两个数和四则运算符号提交给该页面。用户提交表单后,JSP 页面 input.jsp 将计算任务交给另一个 JSP 页面 result.jsp 去完成。

• input.jsp 的具体要求

input.jsp 页面提供一个表单,要求表单中提供两个 text 输入框,供用户输入数字;提

供一个下拉列表，该下拉列表有加、减、乘、除四个选项，供用户选择运算符号。用户在表单中输入的数字、选择运算符号提交给 result.jsp 页面。

- result.jsp 的具体要求

要求 result.jsp 页面获取 input.jsp 提交的数据，并计算出相应的结果显示给用户。

4. 参考代码

代码仅供参考，学生可按照实验要求，参考本代码编写代码。

- JSP 页面参考代码

input.jsp

```
<%@ page contentType="text/html;charset=GB2312" %>
<HTML><body bgcolor=yellow>
   <form action="result.jsp" method=post name=form>
     输入运算数、选择运算符号:<br>
     <input type=text name="numberOne" size=6>
        <select name="operator" >
           <Option value="+">加
           <Option value="-">减
           <Option value="*">乘
           <Option value="/">除
        </select>
     <input type=text name="numberTwo"  size=6>
     <br><input type="submit" value="提交" name="submit">
   </form>
</Font></BODY></HTML>
```

result.jsp

```
<%@ page contentType="text/html;charset=GB2312" %>
<HTML><body bgcolor=green>
<%
    String numberOne=request.getParameter("numberOne");
    String numberTwo=request.getParameter("numberTwo");
    String operator=request.getParameter("operator");
    if(numberOne==null){
       numberOne="0";
    }
    if(numberTwo==null){
       numberTwo="0";
    }
    try{
         double a=Double.parseDouble(numberOne);
         double b=Double.parseDouble(numberTwo);
         double r=0;
         if(operator.equals("+"))
             r=a+b;
         else if(operator.equals("-"))
             r=a-b;
         else if(operator.equals("*"))
```

```
            r = a * b;
        else if(operator.equals("/"))
            r = a/b;
        out.println(a + "" + operator + "" + b + " = " + r);
    }
    catch(Exception e){
        out.println("请输入数字字符");
    }
%>
</body></HTML>
```

3.6.2 实验 3_2 session 对象

1. 相关知识点

HTTP 协议是一种无状态协议。一个用户向服务器发出请求(request),然后服务器返回响应(response),连接就被关闭了。所以,Tomcat 服务器必须使用内置 session 对象(会话)记录有关连接的信息。同一个用户在某个 Web 服务目录中的 session 是相同的;同一个用户在不同的 Web 服务目录中的 session 是互不相同的;不同用户的 session 是互不相同的。一个用户在某个 Web 服务目录的 session 对象的生存期限依赖于 session 对象是否调用 invalidate()方法使得 session 无效或 session 对象达到了设置的最长的"发呆"时间。

2. 实验目的

本实验的目的是让学生掌握怎样使用 session 对象存储和用户有关的数据。

3. 实验要求

本实验编写 3 个 JSP 页面 login.jsp,show.jsp 和 exit.jsp。login.jsp 页面提供一个表单,用户可以通过表单输入姓名提交给 login.jsp 页面,login.jsp 页面将用户的姓名存放到用户的 session(会话)中。如果用户链接到 show.jsp 页面,该页面将检查用户的 session 中是否存放了姓名,否则就将用户重新定向到 login 页面;如果用户链接到 exit.jsp 页面,exit.jsp 将销毁用户的 session。

- login.jsp 的具体要求

login.jsp 页面提供一个表单,要求表单中提供一个 text 输入框,供用户输入名字提交给当前页面,login.jsp 页面将用户输入的名字存放到用户的 session(会话)中。

- show.jsp 的具体要求

要求 show.jsp 页面可以显示一幅图像。但前提条件是,用户事先必须在 session 中存放有名字,否则无法看到图像,而且还会被重新定向到 login.jsp 页面。

- exit.jsp 的具体要求

用户一旦访问 exit.jsp 页面,用户的 session 对象将被销毁,用户必须重新访问其他的页面获得新的 session。

4. 参考代码

代码仅供参考，学生可按照实验要求，参考本代码编写代码。

- JSP 页面参考代码

login. jsp

```
<%@ page contentType = "text/html;charset = GB2312" %>
<head>
   <a href = login.jsp> 登录</a>
   <a href = show.jsp>  看图</a>
   <a href = exit.jsp> 退出</a>
</head>
<HTML><body bgcolor = yellow>
   <form action = "" method = post name = form>
     输入名字就算登录了:<br>
     <input type = text name = "name" size = 6>
     <br> <input type = "submit" value = "提交" name = "submit">
   </form>
</Font></BODY></HTML>
<%    String name = request.getParameter("name");
      if(name == null)
        name = "";
      session.setAttribute("login_name",name);
 %>
```

show. jsp

```
<%@ page contentType = "text/html;charset = GB2312" %>
<head>
   <a href = login.jsp> 登录</a>
   <a href = show.jsp>  看图</a>
   <a href = exit.jsp> 退出</a>
</head>
<HTML><body bgcolor = cyan>
<%
    String name = (String)session.getAttribute("login_name");
    if(name == null || name.length() == 0){
        response.sendRedirect("login.jsp");                    //重定向到登录页面
     }
 %>
   <image src = "image/flower.jpg" width = 200 height = 178></image>
</body></HTML>
```

exit. jsp

```
<%@ page contentType = "text/html;charset = GB2312" %>
<head>
   <a href = login.jsp> 登录</a>
   <a href = show.jsp>  看图</a>
```

```
    <a href = exit.jsp> 退出</a>
</head>
<HTML><body bgcolor = yellow>
<%
     session.invalidate();
 %>
    <b> session 会话失效
</body></HTML>
```

3.7 小结

- HTTP 通信协议是用户与服务器之间一种提交(请求)信息与响应信息(request/response)的通信协议。在 JSP 中,内置对象 request 封装了用户提交的信息,request 对象获取用户提交信息的最常用的方法是 getParameter(String s);内置对象 response 对象对用户的请求作出动态响应,向用户端发送数据。
- HTTP 协议是一种无状态协议。一个用户向服务器发出请求(request),然后服务器返回响应(respons),但不记忆连接的有关信息。所以,Tomcat 服务器必须使用内置 session 对象(会话)记录有关连接的信息。同一个用户在某个 Web 服务目录中的 session 是相同的;同一个用户在不同的 Web 服务目录中的 session 是互不相同的;不同用户的 session 是互不相同的。
- 一个用户在某个 Web 服务目录的 session 对象的生存期限依赖于用户是否关闭浏览器、session 对象是否调用 invalidate()方法使得 session 无效或 session 对象是否达到了设置的最长的“发呆”状态时间。
- 内置对象 application 由服务器负责创建,每个 Web 服务目录下的 application 对象被访问该服务目录的所有用户共享;不同 Web 服务目录下的 application 互不相同。

习题 3

1. 假设 JSP 使用的表单中有如下的 GUI(复选框):

```
<input type = "checkbox" name = "item" value = "bird" >鸟
<input type = "checkbox" name = "item" value = "apple" >苹果
<input type = "checkbox" name = "item" value = "cat" >猫
<input type = "checkbox" name = "item" value = "moon" >月亮
```

该表单所请求的 JSP 可以使用内置对象 request 获取该表单提交的数据,那么,下列哪些是 request 获取该表单提交的值的正确语句:

A. `String a = request.getParameter("item");`

B. `String b = request.getParameter("checkbox");`

C. `String c[ ] = request.getParameterValues("item");`

D. `String d[ ] = request.getParameterValues("checkbox");`

2. 如果表单提交的信息中有汉字,接受该信息的页面应做怎样的处理?

3. 编写两个 JSP 页面 inputString.jsp 和 computer.jsp,用户可以使用 inputString.jsp 提供的表单输入一个字符串,并提交给 computer.jsp 页面,该页面通过内置对象获取 inputString.jsp 页面提交的字符串,计算并显示该字符串的长度.

4. response 调用 sendRedirect(URL url)方法的作用是什么?

5. 对例子 3_1 的代码进行改动,如果用户在 example3_1.jsp 页面提供的表单中输入了非数字字符,example3_1_ computer.jsp 就将用户重新定向到 example3_1.jsp.

6. 一个用户在不同 Web 服务目录中的 session 对象相同吗?

7. 一个用户在同一 Web 服务目录的不同子目录中的 session 对象相同吗?