




软件开发环境

主讲教师 刘凡

fanliu@hhu.edu.cn



第五章

JSP内置对象



本章主要内容

◆ 5.1 request对象

◆ 5.2 response对象

◆ 5.3 session对象

◆ 5.4 application对象

◆ 5.5 out对象

概述

- JSP为简化页面的开发提供了一些内部对象。
- JSP的编写者可以**直接引用**这些内置对象，不需要由JSP的编写者显式声明，也不需要实例化。它们由**JSP容器**实现和管理，在所有JSP页面中都能使用内部对象。
- 内部对象只对**Java程序片**和**Java表达式**有用，在声明中不能使用。

9大内部对象

- request对象
- response对象
- session对象
- application对象
- out对象
- pageContext对象
- config对象
- page对象
- exception对象

5.1 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。

5.1 request对象

1、获取用户提交信息

- 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= “JSP页面” method= get | post >`

提交手段

`</form>`

- JSP页面可以让request对象用`getParameter(String s)`方法获取表单提交的信息。

5.1 request对象

example3_1.jsp

```
<%@ 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>
```


5.1 request对象

example3_1_computer.jsp

```
<%@ 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>
```

5.1 request对象

example3_2.jsp

```
<%@ 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>
```

5.1 request对象

2、处理汉字信息

使用两种方式避免request对象获取的信息出现乱码：

- **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。如下列所示：

```
String str=request.getParameter("message");
```

```
byte b[]=str.getBytes("ISO-8859-1");
```

```
str=new String(b);
```

- **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");
```

5.1 request对象

example3_3.jsp

```
<%@ 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>
```

5.1 request对象

2、处理汉字信息

- 乱码问题：表单乱码、页面乱码、参数乱码、源文件乱码
- 页面乱码：在JSP页面中，中文显示乱码有两种情况：一种是HTML中的中文乱码，另一种是在JSP中动态输出的中文乱码。
- 解决：页面中使用page指令指定编码必须是GB2312.
- 注意：charset的首字母小写

5.1 request对象

2、处理汉字信息

- 参数乱码：使用get方法提交表单的时候传递的参数如果是中文的话很可能会出现乱码。

- **解决：**仅仅转换这个中文字符串或者设置JSP页面显示编码都是不能解决问题的，需要修改Tomcat服务器的server.xml配置文件才能解决问题。

```
<Connector port="8080" protocol="HTTP/1.1" URIEncoding="gb2312"  
    connectionTimeout="20000"  
    redirectPort="8443" />
```

5.1 request对象

2、处理汉字信息

- 源文件乱码：在Eclipse或者MyEclipse中由于默认的JSP编码格式为ISO-8859-1，所以当打开由其他编辑器编辑的JSP文件时会出现乱码，对于这个问题我们只需要更改一下Eclipse或者是MyEclipse中对JSP的默认编码就可以了。

5.1 request对象

3、request对象常用方法

- `getProtocol()`：获得客户向服务器传送数据所使用的通信协议和它版本号，例如：HTTP/1.1。
- `getServerName()`：获得接受请求的服务器主机名
- `getServerPort()`：获得服务器主机的端口号
- `getRemoteHost()`：获得客户机的全名，如果名字获取不得，则获得客户机的IP地址。
- `getRemoteAddr()`：获得发送请求的客户机的IP地址。
- `getMethod()`：获得客户提交信息方式，如get、post或put等。

5.1 request对象

3、request对象常用方法

- `getServletPath()`: 获得客户请求的JSP页面的文件目录。
- `getContentLength()`: 获得客户提交信息长度, 以字节为单位。
- `getHeader(String name)`: 获得HTTP头文件中由参数name指定的头名字的值。
- `getHeaderNames()`: 获得客户请求中所有头部域的名字。
- `getPathInfo()`: 获得客户请求时关联到URL的附加路径信息, 没有此信息则返回空值。

5.1 request对象

3、request对象常用方法

- `getCookies()`：返回客户端的Cookies对象，结果是一个Cookies数组。如果浏览器没有发送Cookies，则返回空值。
- `getRequestURL()`：获得发出请求字符串的客户端地址。
- `getParameter(String name)`：获得客户提交给服务器的name参数值。
- `getParameterNames()`：获得客户提交给服务器的所有参数名。
- `getParameterValues(String name)`：获得指定参数所有值。

5.1 request对象

example3_4.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import="java.util.*" %>
<MHTML><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
        的名称
        webDir = webDir.substring(1); //去掉名称前面的目录符号: /
        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>
```

5.1 request对象

4、处理HTML标记

1. <form> 标记

<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>

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>

5.1 request对象

4、处理HTML标记

2. <input>标记

<input type="输入对象的类型" name="名字">

(1) 文本框text

<input type="text" name="me" value="hi" maxlength="30">

(2) 单选框radio

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

(3) 复选框checkbox

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

5.1 request对象

4、处理HTML标记

2. <input>标记

<input type="输入对象的类型" name="名字">

(4) 口令框password

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

(5) 隐藏hidden

<input type="hidden" name="h" value="123" > request对象调用getParameter方法，通过name的名字来获取由value指定的值。

(6) 提交键submit

(7) 重置键: reset

<input type="reset" >

5.1 request对象

example3_5.jsp

```
<%@ 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>
```

5.1 request对象

example3_5_receive.jsp

```
<%@ 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
        String itemName[]=request.getParameterValues("item"); //获取checkbox值
        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>
```


5.1 request对象

4、处理HTML标记

3. <select>和<option>标记

下拉列表的基本格式是：

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

滚动列表的基本格式是：

```
<select name="myName" size="正整数">  
  <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。

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）。

5.1 request对象

example3_6.jsp

```
<%@ 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);
%>
```

5.1 request对象

example3_6.jsp

```
<HTML><center>
  <body background="image/<%= pic %>"><font size=2 >
    <bgsound src = "sound/<%=music %>" loop = -1/>
    <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>
```

5.1 request对象

4、处理HTML标记

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>
```

5.1 request对象

4、处理HTML标记

5. <table>标记

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。<table >中增加选项border可指明该表格是否带有边框。

5.1 request对象

example3_7.jsp

```
<%@ 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>
```

5.1 request对象

4、处理HTML标记

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称；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/图像文件的名称”。
- <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。

5.1 request对象

4、处理HTML标记

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称；如果视频文件在当前Web服务目录一个子目录中，比如avi子目录中，那么<embed>标记中src属性的值就是“avi/视频文件的名称”。

5.1 request对象

4、处理HTML标记

7. <embed>标记

<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。

5.1 request对象

example3_8.jsp

```
<%@ 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 vedio = request.getParameter("vedio");
    if(vedio==null) vedio = "";
    vedio = handleStr(vedio);
%>
```

5.1 request对象

example3_8.jsp

```
<HTML><center>
<form action="" method=post name=form>
<b>选择视频:<br>
  <select name=vedio >
    <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/<%=vedio %>" width=300 height=180 >视频
</embed><!-- 视频 -->
</font></body></Center></HTML>
```

5.1 request对象

5、处理超链接

- 在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）。

5.1 request对象

example3_9.jsp

```
<%@ 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>
```

5.1 request对象

example3_9_receive.jsp

```
<%@ 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>
```

5.2 response对象

- 与request对象相对应的对象是response对象，response对象对客户的请求做出响应，向客户端发送数据。
- 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引擎将按着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。

5.2 response对象

1、动态响应contentType属性

- 由于page指令只能为contentType指定一个值来决定响应的MIME类型，如果想动态的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：text/html、text/plain、image/gif等
- 当用setContentType方法动态改变了contentType的属性值，JSP引擎就会按着新的MIME类型将JSP页面的输出结果返回给用户

5.2 response对象

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>
```

5.2 response对象

2、response的HTTP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/example/example3_1.jsp/HTTP.1.1 //请求行

host:localhost:8080 //HTTP头:host

accept-encoding:gzip,deflate //HTTP头:accept-encoding

- 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）。

5.2 response对象

2、response的HTTP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。

5.2 response对象

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>
```

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。

5.2 response对象

3、response重定向

- 某些情况下，当响应用户时，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。

5.2 response对象

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>
```

5.2 response对象

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>
```

5.2 response对象

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。
- 3位数字的状态码：1yy，实验性质；2yy，表示请求成功，如200；3yy，表示请求满足前应采取进一步行动；4yy，表示无法满足请求，如404，请求页面不存在；5yy，表示服务器出现问题，如505，服务器内部发生问题。
- 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。也可以用Response对象的setStatus(int n)设置响应的状态行的内容。

5.2 response对象

example3_13.jsp

```
<%@ 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>
```

5.2 response对象

example3_13_bird.jsp

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

```
<% response.setStatus(408);//请求超时
```

```
%>
```

```
<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>
```

5.3 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。
- 但是在很多应用中，需要服务器在用户访问的一个会话期中记住客户，为客户提供个性化服务。
- Tomcat服务器可以使用内置session对象（会话）记录有关连接的信息。

5.3 session对象

1、session对象ID

- 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。
- 该对象对应一个String类型的ID号，服务器在响应客户请求的同时，把ID号发到客户端，并写入客户端的cookie中。
- 当客户关闭浏览器后，一个会话结束，服务器端该客户的session对象被取消。当客户重新打开浏览器建立新连接时，JSP引擎为该客户再创建一个新的session对象。

5.3 session对象

example3_14_a.jsp

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

```
<HTML><body bgcolor=cyan>
```

我是example3_14_a.jsp页面
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>
```

5.3 session对象

example3_14_b.jsp

```
<%@ 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>
```

5.3 session对象

2、session对象与URL重写

- 如果用户不支持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。

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

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

- 然后将连接目标写成<%= str %> 即可。

5.3 session对象

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：

- `public void setAttribute(String key, Object obj)`
- `public Object getAttribute(String key)`
- `public Enumeration getAttributeNames()`
- `public void removeAttribute(String name)`

5.3 session对象

3、session对象存储数据

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了一个索引关键字。

② `public Object getAttribute(String key)`

获取session对象索引关键字是key的对象。

5.3 session对象

3、session对象存储数据

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nextElements()遍历session中的各个对象所对应的关键字。

④ **public void removeAttribute(String name)**

session对象调用该方法移掉关键字key对应的对象

5.3 session对象

example3_15_a.jsp

```
<%@ 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>
```

5.3 session对象

example3_15_b.jsp

```
<%@ 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>
```

5.3 session对象

example3_15_b.jsp

```
<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>
```

5.3 session对象

example3_15_c.jsp

```
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;
}
%>
```

5.3 session对象

example3_15_c.jsp

```
<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.]");
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>
```

5.3 session对象

example3_16_number.jsp

```
<%@ 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>
```


5.3 session对象

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");}}}%>
```

5.3 session对象

example3_16_large.jsp

```
<%@ 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>
```

5.3 session对象

example3_16_small.jsp

```
<%@ 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>
```

5.3 session对象

example3_16_success.jsp

```
<%@ 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>
```

5.3 session对象

4、session对象的生命周期

- 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）。

5.3 session对象

4、session对象的生命周期

- 可以通过Tomcat目录conf文件下的配置文件web.xml修改默认发呆时间：

```
<session-config>
```

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

```
</session-config>
```

- session对象也可以使用相应的函数获取或设置与生存时间有关的信息：`setMaxInactiveInterval(int interval)`设置最长发呆时间（单位秒），`getMaxInactiveInterval()`获取最长发呆时间。

5.3 session对象

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");
```

5.3 session对象

example3_17.jsp

```
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>
```


5.3 session对象

5、session对象的特点

- ❑ 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。
- ❑ session对象被分配了一个String类型的ID，Tomcat服务器将ID发送到客户端，存放在客户的Cookie中。
- ❑ 同一用户在同一Web服务目录中的各个页面的session是相同的。
- ❑ 不同用户的session对象互不相同，具有不同的ID
- ❑ Session生存周期：关闭浏览器；调用invalidate()方法；超过最长的“发呆”时间。

5.4 application对象

- 不同的客户拥有不同的session对象。与session对象不同的是：
application对象由多个客户端用户共享。
- 服务器启动后，新建一个对应Web服务目录的application对象，
该对象一旦建立，就一直保持到服务器关闭。
- 每个Web服务目录下的application对象被访问该服务目录的所有
的用户共享，但不同Web服务目录下的application互不相同。

5.4 application对象

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了一个索引关键字，如果添加的两个对象的关键字相同，则先前添加对象被清除。

② `public Object getAttibue(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。

5.4 application对象

1、application对象的常用方法

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**nextElements()**遍历**application**中的各个对象所对应的关键字。

④ **public void removeAttribute(String key)**

从当前**application**对象中删除关键字是**key**的对象。

5.4 application对象

example3_18_input.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<HTML><body bgcolor=cyan>
  <form action="example3_18_panel.jsp" method="post"
    name="form">
    输入名字: <input type="text" name="peopleName">
    <br>留言标题: <input type="text" name="title">
    <br>留言: <br> <textArea name="messages" rows="10" cols=36
      wrap="physical"></textArea>
    <br><input type="submit" value="提交" name="submit">
  </form>
  <a href="example3_18_show.jsp">查看留言板</a>
</body></HTML>
```

5.3 session对象

example3_18_panel.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import="java.util.*" %>
<HTML><body>
    <%! Vector v=new Vector();//向量，大小可变
        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);
        }
    %>
```

5.3 session对象

example3_18_panel.jsp

```
<% 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>
```

5.3 session对象

example3_18_show.jsp

```
<%@ 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>
```


5.5 out对象

out对象是一个输出流，用来向用户端输出数据。**out**对象可调用如下的方法用于各种数据的输出：

- **out.print(boolean)、out.println(boolean)** 用于输出一个布尔值
- **out.print(char)、out.println(char)** 输出一个字符
- **out.print(double)、out.println(double)** 输出双精度的浮点数
- **out.print(long)、out.println(long)** 输出一个长整形数
- **out.print(string)、out.println(string)** 输出一个字符串
- **out.newline()** 输出一个换行符
- **out.flush()** 输出缓冲区的内容
- **out.close()** 关闭流

5.6 pageContext对象

- javax.servlet.jsp.PageContext类的实例对象
- 封装了对其它八大隐式对象的引用
- pageContext对象最常用的方法有：

void setAttribute(String name, Object value)

void setAttribute(String name, Object value, int scope)

以键/值的方式，将一个对象的值存放到pageContext中

void getAttribute(String name)

void getAttribute(String name, int scope)

根据名称去获取pageContext中存放对象的值

5.7 page对象

- page对象是当前页面转换后的Servlet类的实例
- 代表当前的JSP页面
- 有点类似于Java编程中的this指针

```
<% @ page info="我的信息" contentType="text/html;charset=GBK"%>
<html>
<body>
    <%=((javax.servlet.jsp.HttpJspPage)page).getServletInfo()%>
</body>
</html>
```

5.8 config对象

- javax.servlet.ServletConfig 接口的实例
- 代表当前JSP 配置信息
- 提供了检索 Servlet 初始化参数的方法

config对象
获取初始化参数



```
String propertyFile  
=(String)config.getInitParameter("PropertyFile");
```

返回名字为propertyFile的初始化参数的值,初始化参数在web.xml配置文件中配置,如果不存在则返回null

5.9 exception对象

- exception对象用于处理 JSP 页面中的错误、异常
- exception 对象是 java.lang.Throwable 类的实例
- 注意：exception对象仅仅在异常处理页面中有效。

```
<%@ page isErrorPage="true" %>
<html>
  <head><title> 处理错误 </title></head>
  <body>
    <% if ( exception != null )
      {out.write("\n 发生错误。 \n");}
    else
      {out.write("\n 您已访问此页面，但是没有可用的错
误信息\n");}
    %>
  </body></html>
```

5.10 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的通信协议；在JSP中，内置对象request封装了请求信息，response对象请求作出响应，向用户发送数据。
- ❑ HTTP是一种无状态协议，无法记忆连接的有关信息。通过session对象记录连接的信息，同一用户在同一Web服务目录中的各个页面的session是相同的，在不同Web服务目录中Session互不相同。
- ❑ Session生存周期依赖于关闭浏览器、调用invalidate()方法、最长的“发呆”时间。
- ❑ 内置application对象由服务负责创建，每个Web服务目录下的application对象被所有访问该服务目录的用户共享，不同Web服务目录下的application对象互不相同。