



Java Server Page

# JSP程序设计(中)

isszym sysu.edu.cn

2016.11.19

# 如何上传文件

fileUploadShow.jsp



The screenshot shows a web browser window with the title '实现文件上传'. The address bar displays '202.116.76.22:8080/jsp/fileUploadShow.jsp'. The form contains the following elements:

- 名称:
- 性别:
- 年龄:
- 文件1:
- 文件2:
- 

↑  
点击

## *fileupload.jsp*



如果上传的文件使用汉字作为文件名，需要conf/server.xml的Connector中加入  
URIEncoding="UTF-8"

```
<Connector port="8080" protocol="HTTP/1.1"  
    connectionTimeout="20000"  
    redirectPort="8443"  
    URIEncoding="UTF-8" />
```

## fileUploadShow.jsp

<http://172.18.187.230:8080/jsp/fileUploadShow.jsp>

```
<%@ page language="java" import="java.util.*" contentType="text/html;
charset=utf-8"%>
<!DOCTYPE HTML>
<html>
<head>
    <title>实现文件上传</title>
</head>
<body>
    <%request.setCharacterEncoding("utf-8");%>
    <p>实现文件上传</p>
    <form name="fileupload" action="fileupload.jsp" method="POST"
        enctype="multipart/form-data">
        <p>名称: <input type="text" name="name" size=24 value="David"></p>
        <p>性别: <input type="text" name="sex" SIZE=24 value="male"></p>
        <p>年龄: <input type="text" name="age" SIZE=24 value="28"></p>
        <p>文件1: <input type="file" name="file1" size=24></p>
        <p>文件2: <input type="file" name="file2" size=24></p>
        <p><input type="submit" name="submit" value="OK"></p>
    </form>
</body>
</html>
```

## *fileupload.jsp*

```
<%@ page pageEncoding="utf-8" contentType="text/html; charset=utf-8"%>
<%@ page import="java.io.*, java.util.*,org.apache.commons.io.*"%>
<%@ page import="org.apache.commons.fileupload.*"%>
<%@ page import="org.apache.commons.fileupload.disk.*"%>
<%@ page import="org.apache.commons.fileupload.servlet.*"%>
<html><head><title>文件传输例子</title></head>
<body><%request.setCharacterEncoding("utf-8");%>
    <% boolean isMultipart
        = ServletFileUpload.isMultipartContent(request); //检查表单中是否包含文件
    if (isMultipart) {
        FileItemFactory factory = new DiskFileItemFactory();
        //factory.setSizeThreshold(yourMaxMemorySize); //设置使用的内存最大值
        //factory.setRepository(yourTempDirectory); //设置文件临时目录
        ServletFileUpload upload = new ServletFileUpload(factory);
        //upload.setSizeMax(yourMaxRequestSize); //允许的最大文件尺寸
        List items = upload.parseRequest(request);
        for (int i = 0; i < items.size(); i++) {
            FileItem fi = (FileItem) items.get(i);
            if (fi.isFormField()) { //如果是表单字段
                out.print(fi.getFieldName()+":"+fi.getString("utf-8"));
            }
            else { //如果是文件
```

```
DiskFileItem dfi = (DiskFileItem) fi;
if (!dfi.getName().trim().equals("")) { //getName()返回文件名称或空串
    out.print("文件被上传到服务上的实际位置: ");
    String fileName=application.getRealPath("/file")
        + System.getProperty("file.separator")
        + FilenameUtils.getName(dfi.getName());
    out.print(new File(fileName).getAbsolutePath());
    dfi.write(new File(fileName));
}
} //if
} //for
} //if
%>
</body>
</html>
```

<http://www.runoob.com/jsp/jsp-file-uploading.html>

# 如何在网页之间传递参数

1、URL参数: 利用HTTP响应形成网页时放在链接里。

`<a href="www.abc.com?x=2&y=3">做加法</a>`

用HTTP请求传递到下个网页，下一个网页通过`request.getParameter("x")`取到值。

2、利用session

利用服务器端保存<key value>对，服务器利用HTTP响应向客户端发回SessionID(cookie)，根据SessionID该服务器可以唯一确定客户端。以后客户端每次向该服务器发出HTTP请求时会发出该SessionID，服务器端利用该SessionID去访问临时存放<key，value>对。当浏览器窗口完全关闭或很长时间没有访问该服务器后该sessionID失效。

3、利用cookie

利用客户端保存<key，value>对，每次向同一个网站发出HTTP请求时会一起发出，可以设置保存时间。key=value

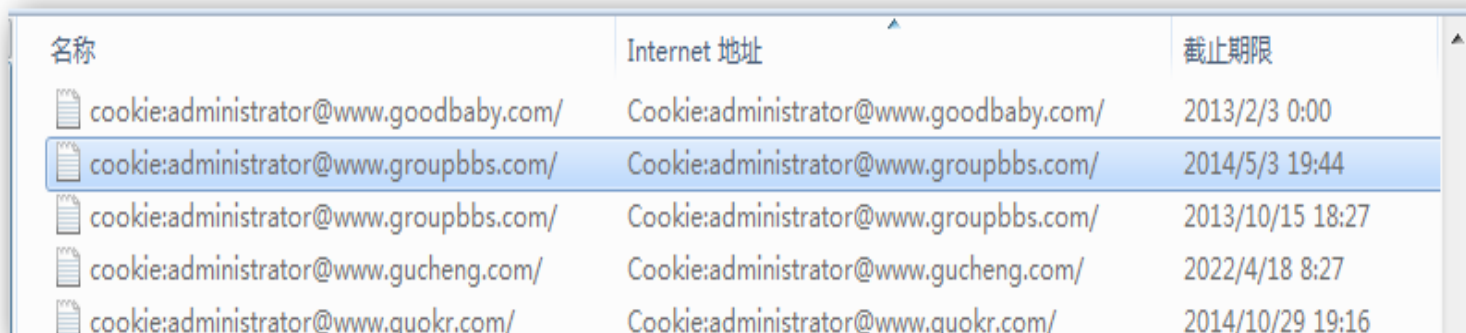
4、利用post

\* 3和4都很少用于在页面之间传递参数。

# 用Cookie把数据保存在客户端

- 概述

- Web服务器不保留状态。
- Cookie采用key-value方式在客户端保留一些短信息，例如：用户名和密码。当再次访问相同的网站，浏览器会自动把这些数据提交给服务器。
- 每个cookie有很多项：名(name)、值(value)、域(domain)、路径(path)、过期时间(expires)。
- 查看IE 上的cookie： 工具/选项/常规/浏览历史记录/设置/查看文件。

A screenshot of the Internet Explorer 'View Cookies' window. It displays a table of cookies with three columns: '名称' (Name), 'Internet 地址' (Internet Address), and '截止期限' (Expiration Date). The table lists five cookies, all with the name 'cookie:administrator@' followed by a domain. The second row, for 'www.groupbbs.com/', is highlighted in blue.

名称	Internet 地址	截止期限
cookie:administrator@www.goodbaby.com/	Cookie:administrator@www.goodbaby.com/	2013/2/3 0:00
cookie:administrator@www.groupbbs.com/	Cookie:administrator@www.groupbbs.com/	2014/5/3 19:44
cookie:administrator@www.groupbbs.com/	Cookie:administrator@www.groupbbs.com/	2013/10/15 18:27
cookie:administrator@www.gucheng.com/	Cookie:administrator@www.gucheng.com/	2022/4/18 8:27
cookie:administrator@www.quokr.com/	Cookie:administrator@www.quokr.com/	2014/10/29 19:16



## • cookie的内容

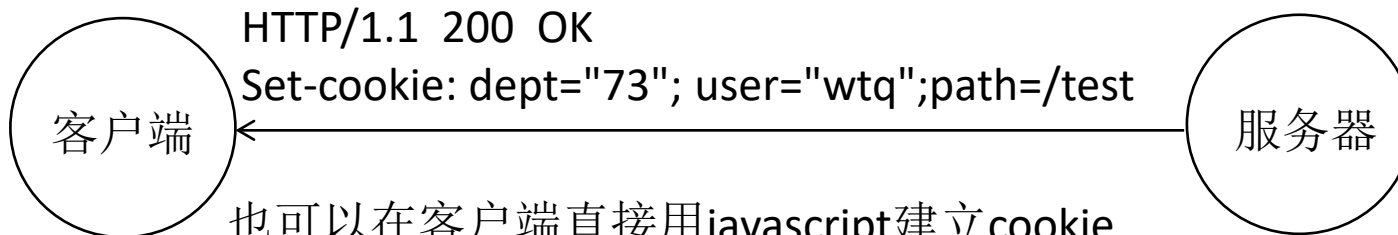
```
1  UserRnd      -----
2  Rnd=843
3  www.groupbbs.com/
4  1536
5  138375808
6  30369477
7  2438931904
8  30296051
9  *

Login
Password=123456&UserName=
admin
www.groupbbs.com/
1536
1342440960
30310740
883030176
30298469
*
```

第1行: Cookie 名称  
第2行: Cookie 的值 (域名+路径)  
第3行: Cookie 所属站点的地址  
第4行: 一个标记值(是否被加密)  
第5行: 超时时间的低位(Cardinal/DWORD)  
第6行: 超时时间的高位  
第7行: 创建时间的低位  
第8行: 创建时间的高位  
第9行: 固定为 \* , 表示一节的结束

## • http中的cookie

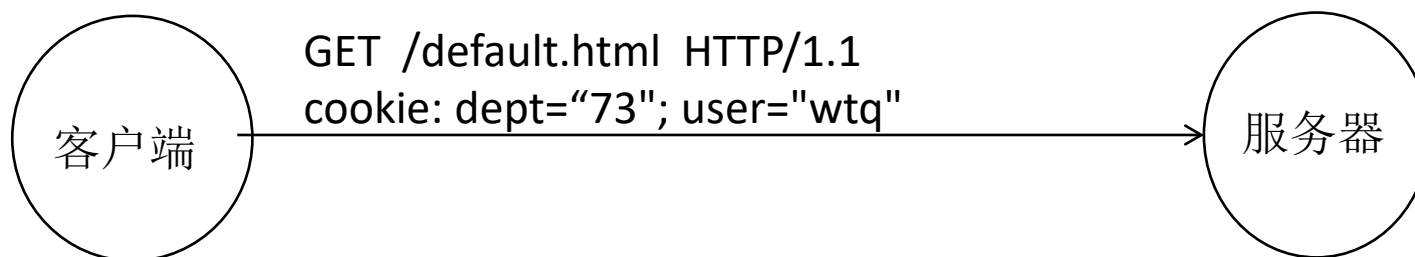
服务器端可以利用HTTP响应建立新的cookie



也可以在客户端直接用javascript建立cookie

\*未设路径使用当前路径，未设域名采用当前域名

以后每次提交的HTTP请求会带上为该页面设置的有效cookie



\* 客户端每次提交网页都会附上在当前页面及其祖先路径上设置的Cookie。

```
Set-Cookie: <name>=<value>[; <name>=<value>]...  
            [; expires=<date>][; domain=<domain_name>]  
            [; path=<some_path>][; secure][; httponly]
```

[参考](#)

## • JSP Cookie 编程

```
<%  
Cookie cookie = new Cookie("cookieName","cookievalue");  
cookie.setMaxAge(3600); // 设置保留时间 3600秒。  
                        // 设置为负值表示只保存在内存，关闭浏览器则消失。  
                        // 设置为0表示要删除该cookie。  
cookie.setPath("/");    //设置路径为根目录  
response.addCookie(cookie);  
%>
```

[setCookie.jsp](#)

```
<%  
Cookie[] cookies = request.getCookies();  
for(Cookie cookie : cookies){  
    String name= cookie.getName();    // get the cookie name  
    String value=cookie.getValue();    // get the cookie value  
}  
%>
```

[getCookie.jsp](#)

## • Cookie对象的方法

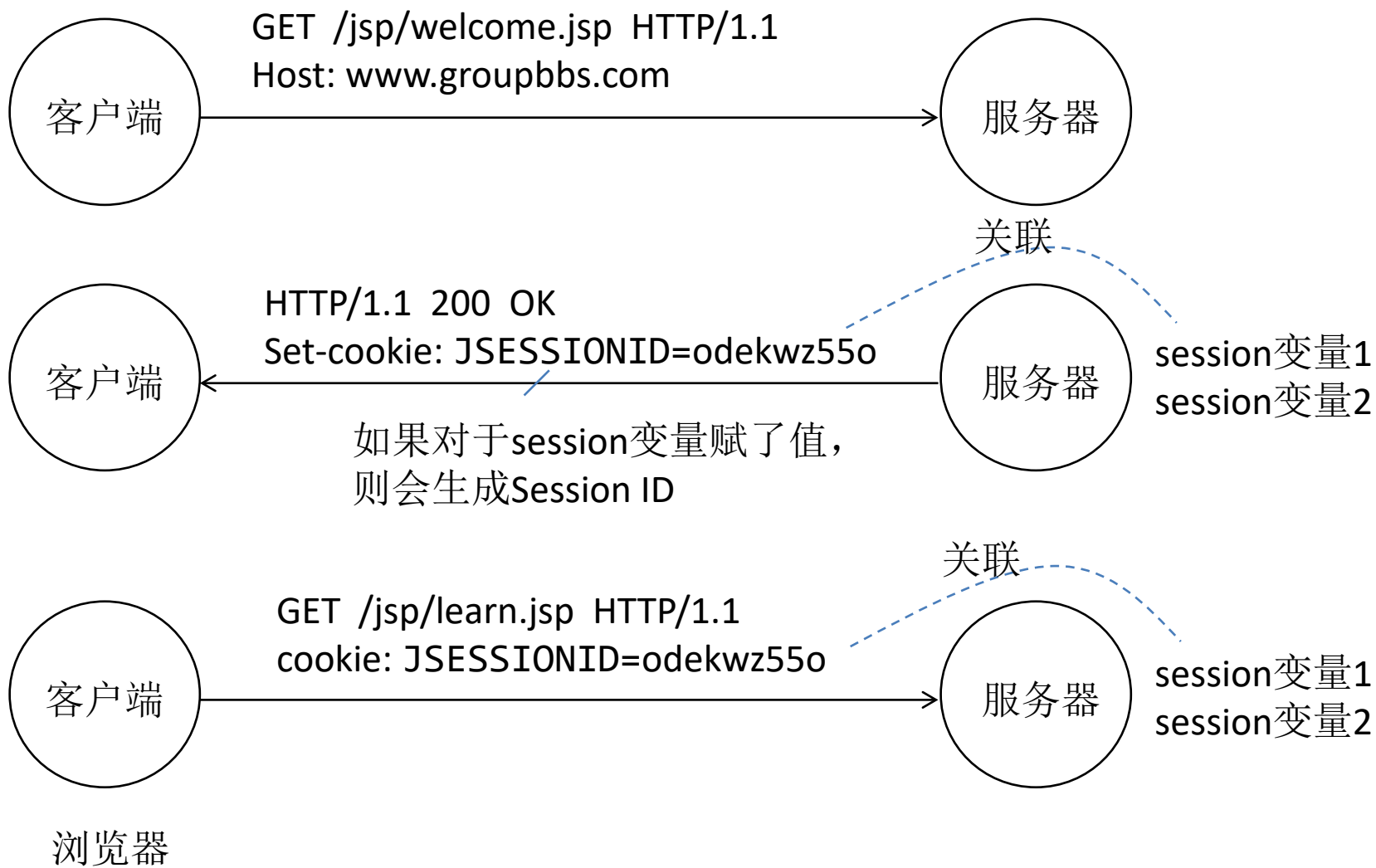
类型	方法名	方法解释
String	getComment()	返回cookie中注释,如果没有注释的话将返回空值.
String	getDomain()	返回cookie中Cookie适用的域名. 使用getDomain() 方法可以指示浏览器把Cookie返回给同一域内的其他服务器,而通常Cookie只返回给与发送它的服务器名字完全相同的服务器。注意域名必须以点开始（例如.yesky.com）
int	getMaxAge()	返回Cookie过期之前的最大时间，以秒计算。
String	getName()	返回Cookie的名字。名字和值是我们始终关心的两个部分。
String	getPath()	返回Cookie适用的路径。如果不指定路径，Cookie将返回给当前页面所在目录及其子目录下的所有页面。
boolean	getSecure()	如果浏览器通过安全协议发送cookies将返回true值，如果浏览器使用标准协议则返回false值。
String	getValue()	返回Cookie的值。笔者也将在后面详细介绍getValue/setValue。
int	getVersion()	返回Cookie所遵从的协议版本。
void	setComment(String purpose)	设置cookie中注释。
void	setDomain(String pattern)	设置cookie中Cookie适用的域名
void	setMaxAge(int expiry)	以秒计算，设置Cookie过期时间。
void	setPath(String uri)	指定Cookie适用的路径。
void	setSecure(boolean flag)	指出浏览器使用的安全协议，例如HTTPS或SSL。
void	setValue(String newValue)	cookie创建后设置一个新的值。
void	setVersion(int v)	设置Cookie所遵从的协议版本。

# 用Session对象把数据保存在服务器端

- 概述

- ✓ http协议是无状态的，但是通过session可以在Web服务器上临时保存少量变量的状态。session保存的状态只与当前浏览器相关。在用户第一次访问页面时Web服务器就会送回sessionId(作为cookie)。服务器根据sessionId访问session变量。
- ✓ 通过服务器端语言(JSP, ASP.net, PHP)设置session对象。一个网站的后台程序可以利用session变量在后台（服务器）保留状态。session与Web服务器有关，一个Web服务器中设置的session，另一个Web服务器访问不了。
- ✓ 如果一个网站的后台程序设置了session变量，当通过浏览器第一次访问该网站时，服务器脚本语言就会生成一个唯一的session ID，并作为cookie传送到客户端保存。以后每次访问该网站，该session ID都会和其它cookie一起送到服务器。通过session ID服务器端可以找到为该session保存的变量，使得这些变量可以被不同网页所共享。
- ✓ session的生命周期和浏览器有关，浏览器关闭session将被删除。但是如果你长时间不关闭浏览器，又不访问该网站，session的生命周期也会结束，一般是三十分钟。

- http协议中的session



## • session对象编程

// 取出当前浏览器的session Id

```
<% String id= session.getId(); %>
```

//在一个网页中给session变量user赋值

```
<% session.setAttribute("user", "LiJN"); %>
```

//在另一个网页中取出session变量user的值

```
<% String user= (String)session.getAttribute("user"); %>
```

//取到所有session变量的名和值

```
<% Enumeration enums=session.getAttributeNames();  
    while(enums.hasMoreElements()){  
        String name=(String)enums.nextElement();  
        out.println(name+": "+session.getAttribute(name)+"<br />");  
    }  
%>
```

```
//删除session变量
session.removeAttribute("user");
// 令所有session无效
session.invalidate();
long tm=session.getCreateTime("user"); //获得变量的创建时间(毫秒)
out.print(new Date(tm))

long ac=session.getLastAccessedTime();//获得最后的访问时间(毫秒)
session.setMaxInactiveInterval(10); //设置session最大不活动时间(秒)
long mi=session.getMaxInactiveInterval();//秒
boolean n=session.isNew(); //是否是新建的session
```

默认的interval time为30分钟（过期后session边的无效），存放在tomcat/conf/web.xml中：

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



# include指令


## 功能

用于在JSP转换为Servlet程序前将源文件包含进来。include为指令标签。

## 语法

```
<%@ include file="文件的绝对路径或相对路径" %>
```

## 说明



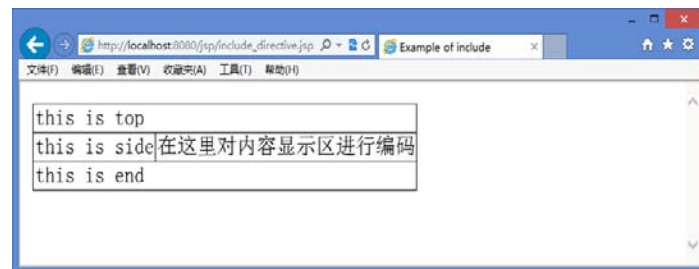
注意：不是url

该属性指定被包含的文件，不支持表达式，也不允许传递参数。被包含的文件的内容原封不动地插入到主文件中，只要被包含的文件发生改变，整个主页面文件就会重新被编译，编译后只有一个.class文件。主文件和子文件的page指令不应重复。

[http://172.18.187.230:8080/jsp/include\\_directive.jsp](http://172.18.187.230:8080/jsp/include_directive.jsp)

```
<%@ page contentType="text/html; charset=gb2312"%>
<html> <head> <title>Example of include</title>
<style type="text/css">
    table {
        border-collapse: collapse;
    }
    table, td, th {
        border: 1px solid black;
    }
</style> </head>
<body>
<table style="border: 1px solid black">
    <tr><td colspan="2">
        <%@ include file="top.jsp"%>
    </td>
</tr>
<tr> <td>
        <%@ include file="side.jsp"%></td>
        <td>在这里对内容显示区进行编码</td>
</tr>
<tr> <td colspan="2">
        <%@ include file="end.jsp"%>
    </td>
</tr>
</table>
</body> </html>
```

top.jsp、side.jsp和end.jsp都只有一句话，分别是：this is top、this is side和this is end。执行include\_directive.jsp的结果：



# 动作标签<jsp:include>

用于向当前的页面中嵌入其它的动态文件并执行，类似于调用一个外部函数。

```
<jsp:include page="relativeURL" flush="true|false"/>
```

或者

```
<jsp:include page=" relativeURL " flush="true|false">  
  <jsp:param name="参数名称" value="参数值"/>  
</jsp:include>
```

属性flush="true|false"表示当输出缓冲区满时，是否清空缓冲区，默认为false。通过page属性指定被包含的页面，该属性支持JSP表达式。只在页面被请求的时候才包含指定文件。被包含的文件中不能含有某些JSP代码，例如，不能设置HTTP头。对于静态文件，会直接输出，对于动态文件，则需要编译并执行这个文件。

主页面将请求转发到被包含的页面，并将执行结果输出到浏览器中，然后返回主页面继续执行后面的代码。JSP编译器会分别对这两个文件进行编译。被包含文件的改动后不会引起主页面文件重新编译，而只是在执行时重新编译被包含的文件。

例子:

http://202.116.76.22/jsp/include\_action.jsp

```
<%@ page language="java" import="java.util.*"
      pageEncoding="ISO-8859-1"%>

<html>
<body>
    Here is three new stories:
    <ol>
        <li><jsp:include page="included1.jsp" />
        <li><jsp:include page="included2.jsp" />
        <li><jsp:include page="included3.jsp" />
    </ol>
</body>
</html>
```

http://202.116.76.22/jsp/included1.jsp

```
<html>
<head>
<title>new story 1</title>
</head>
<body>this is story 1!
</body>
</html>
```

Here is three new stories:

1. this is story 1!
2. this is story 2!
3. this is story 3!

\* included2.jsp、included3.jsp与included1.jsp类似

# 动作标签<jsp:forward>

本动作标签用来将请求在后台转移到另外一个JSP文件(相当于goto)，会继承原页面的所有请求参数。

```
<jsp:forward page="relativeURL" />
```

<jsp:forward>标签实现的是请求的转发操作，而不是请求重定向，即它只是简单把请求转发给另一个文件而不是重新产生一个新的请求。

forward.jsp

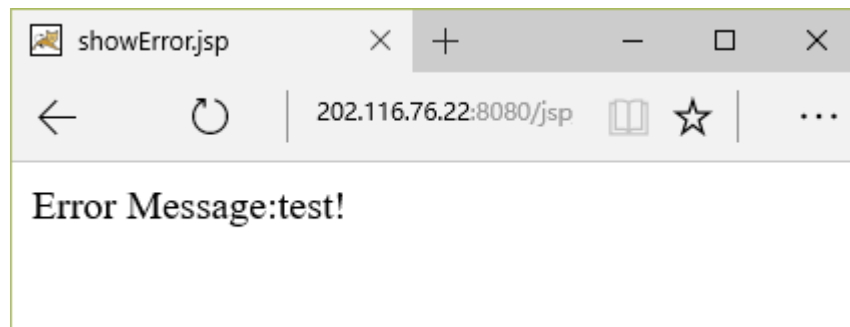
```
<%@ page language="java" import="java.util.*"
      pageEncoding="ISO-8859-1"%>
<!DOCTYPE HTML>
<html>
  <head>
    <title>My JSP 'forward.jsp' starting page</title>
  </head>
  <body>
    <%String url = "showError.jsp?error=test!"; %>
    <jsp:forward page="<%=url%>" />
  </body>
</html>
```

前台转移方式：  
response.sendRedirect(url)

## showError.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<!DOCTYPE HTML>
<html>
  <head>
    <title>showError.jsp</title>
  </head>

  <body>
    <%= "Error Message: "+request.getParameter("error")%><br>
  </body>
</html>
```



# 注释语句

**功能：**对JSP程序进行说明，不会被执行。

**语法：**

- (1) **HTML注释：** `<!-- 注释内容 -->` 会出现在客户端
- (2) **脚本外注释：** `<%-- 注释内容 --%>` 不会出现在客户端
- (3) **脚本内注释：**
  - `// 行注释内容`
  - `/* 多行注释 */`
  - `/**可以被javadoc读取 */`

## comments.jsp

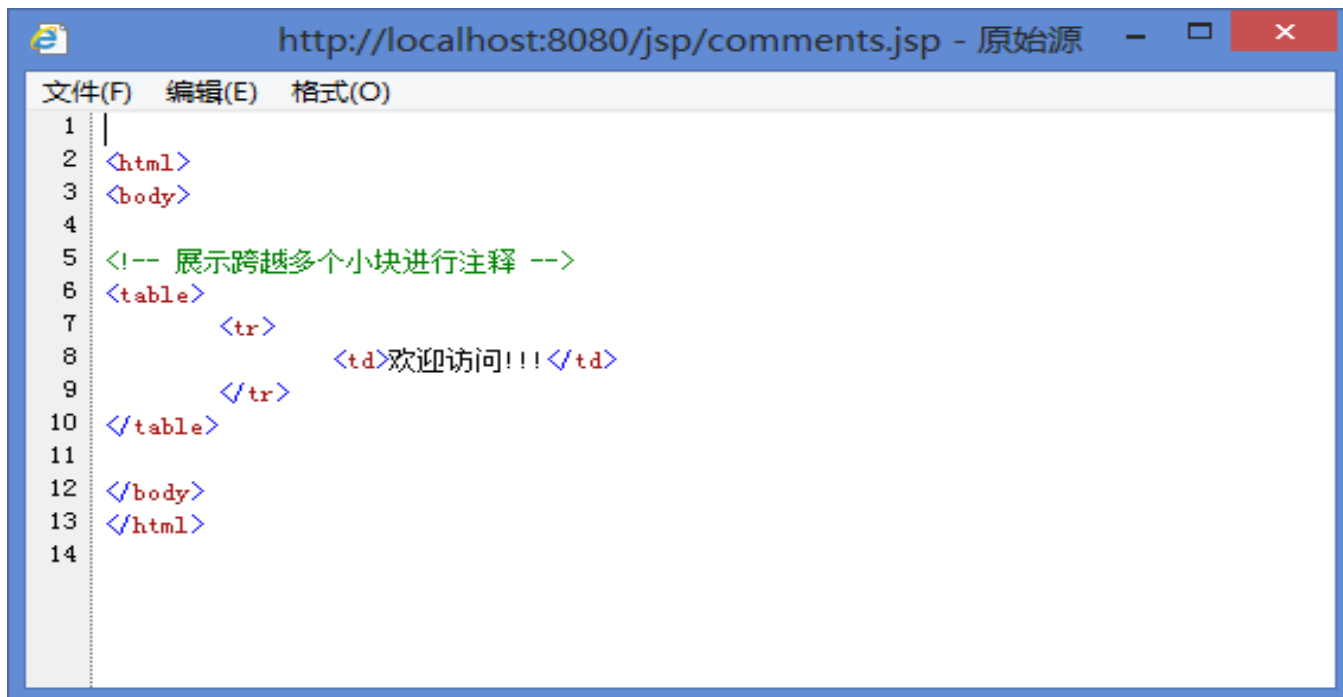
```
<%@ page contentType="text/html; charset=gb2312"%>
<html>
<body>
<!-- 欢迎提示信息! --%>
<!-- 展示跨越多个小块进行注释 -->
<table>
    <tr>
        <td>欢迎访问!!!</td>
    </tr>
</table>
<%
    String state = "用户";           // 定义当前状态
    /* if(state.equals("用户")){     // 判断两个字符串是否相等
        state="版主";
    }
    将变量state赋值为"版主"。<br>
*/
%>
</body>
</html>
```



执行结果:



客户端源码:



# JSP 的基本内置对象

- 概述

JSP包含可直接使用的9种基本内置对象：

request	用来获取客户端请求的信息
response	网页传回用户端的回应
pageContext	网页的属性是在这里管理
session	与请求有关的会话期
application	<b>Servlet</b> 正在执行的内容
out	用来传送响应的输出
config	<b>Servlet</b> 的构架部件
page	JSP网页本身
exception	针对错误网页，未捕捉的例外

- request对象

request对象用来获取用户提交的信息。

- (1) getParameter()

取提交属性name为empid控件的值:

```
String empId=request.getParameter("empid");
```

取得所有提交的控件的name:

```
Enumeration enums=request.getParameterNames();
```

取提交属性name为cities一组控件的值:

```
String cities[]=request.getParameterValues("cities");
```

URL的参数也是用getParameter取出:

http请求:

```
http://localhost:8080/queryStu.jsp?stuId=10032004&cs=true
```

queryStu.jsp:

```
String empId=request.getParameter("stuId");
```

```
String cs=request.getParameter("cs");
```

```
<% Enumeration enums=request.getParameterNames();
    while(enums.hasMoreElements()){
        String name=(String)enums.nextElement();
        out.println(name+": "+request.getParameter(name)+"<br>");
    }
%>
```

```
<% String[] courses=request.getParameterValues("courses");
    try{
        for(int i=0; i<courses.length; i++){
            out.print(courses[i]+" ");
        }
    }
    catch(Exception e){
    }
%>
```

所有提交的内容都会自动执行`urlencode`，`getParameter`会自动做一次解码。  
对于URL中的汉字参数需要进行两次编码，读取时只需要一次解码。

## URL用汉字做参数的方法:

```
<%@ page contentType="text/html; charset=gb2312" %>
<a href="ds.jsp?url=<%=java.net.URLEncoder.encode("编码的是这里", "GB2312")%>">点击这里</a>
```

```
<%
if(request.getParameter("url")!=null){
    str=request.getParameter("url");
    str=java.net.URLDecoder.decode(str, "GB2312");
    str=new String(str.getBytes("ISO-8859-1"));
    out.print(str);
}
%>
```

另一种方法是修改server.xml :

```
<Connector port="8080" maxThreads="150" minSpareThreads="25"
    maxSpareThreads="75" enableLookups="false" redirectPort="8443"
    acceptCount="100" debug="0" connectionTimeout="20000"
    disableUploadTimeout="true"
    <!--在里边加上这个参数-->
    URIEncoding="gb2312"
/>
    http://www.blogjava.net/super-nini/archive/2010/03/23/316253.html
```

## (2) 其它常用方法: request2.jsp

<code>setCharacterEncoding()</code>	设置本http请求的内容所用的编码,以便 <code>getParameter()</code> 等可以正确译码。
<code>getProtocol()</code>	获取http响应所使用的通信协议, 如http/1.1等。
<code>getScheme()</code>	获取URL协议, 如http等。
<code>getContextPath()</code>	获取URL相对路径(虚拟目录), 如/image等。
<code>getServletPath()</code>	获取请求的JSP页面所在的目录。
<code>getContentLength()</code>	获取HTTP请求的长度。
<code>getMethod()</code>	获取表单提交信息的方式, 如POST或GET。
<code>getHeader(String s)</code>	获取请求中头的值。
<code>getHeaderNames()</code>	获取头名字的一个枚举。
<code>getHeaders(String s)</code>	获取头的全部值的一个枚举。
<code>getRemoteAddr()</code>	获取客户的IP地址。
<code>getRemoteHost()</code>	获取客户机的名称(如果获取不到, 就获取IP地址)。
<code>getServerName()</code>	获取服务器的名称。
<code>getServerPort()</code>	获取服务器的端口号。
<code>getParameterNames()</code>	获取表单提交的name参数值的一个枚举。
<code>getRequestURI()</code>	获得URI
<code>getQueryString()</code>	获得URL的查询字符串

<http://blog.csdn.net/lutinghuan/article/details/8275958>

**URL: *http://www.groupbbs.com:8080/jsp/file/request.jsp?x=1&y=2***

request.getScheme():	http
request.getContextPath():	/jsp
request.getServletPath():	/file/request.jsp
request.getContentLength():	-1(没有正文)
request.getMethod():	GET
request.getRemoteAddr():	132.236.174.191
request.getRemoteHost():	132.236.174.191
request.getServerName():	www.groupbbs.com
request.getServerPort():	8080
getRequestURI():	/jsp/file/request.jsp
getQueryString():	x=1&y=2

getRequestURL():  
http://www.groupbbs.com:8080/jsp/file/request.jsp

getRealPath("/"): C:\Program Files\Apache Software Foundation\Tomcat 8.0\webapps\jsp\  
getServletContext().getRealPath("/"): C:\Program Files\Apache Software Foundation\Tomcat 8.0\webapps\jsp\

**可以用读文件的方法读取http请求的全部内容:**

```
int contentLength = request.getContentLength();  
DataInputStream in = new DataInputStream(request.getInputStream());
```

## • response对象

response对象用来设置返回给用户的信息。

```
<% response.setContentType("application/msword;charset=GB2312");  
    // 设置返回的媒体类型和编码.  
%>  
<% response.setContentType("image/jpeg");  
    // 设置返回的媒体类型和编码.  
%>  
<% response.addHeader("refresh",5); //增加头部。不存在就增加，否则更新。  
    // 设置每5秒刷新一次当前网页  
%>  
<% response.sendRedirect("example.jsp"); // 重定向到example.jsp  
  
%>  
<% response.setStatus(404); //设置状态码  
    // 404为错误码.  
%>
```



- out对象

把一串字符输出到HTTP响应页面。

```
out.print(String/boolean/double/float/long var1);  
out.println(String/boolean/double/float/long var2);  
out.newLine();  
out.flush();  
out.close();  
out.write();    //写字节流
```

在全局函数或类中不能访问out对象，只能通过参数带入。

- application对象

整个应用程序（包含很多网页）共享的对象。

```
application.getAttribute(String key);
```

```
application.getAttributeNames();
```

```
application.setAttribute(String key, Object obj);
```

```
application.removeAttribute(String key);
```

```
application.getServletInfo();
```

有些JSP Web服务器不支持application对象，需要先用ServletContext()声明这个对象，然后用getServletContext()进行初始化。

# JavaBean

还可以利用JavaBean来使用类。JavaBean是一个公共类，实现了java.io.Serializable接口。JavaBean的构造器不能带参数。JavaBean不能直接访问类的数据域，只能通过方法isXXX()、getXXX()或setXXX()访问数据域。 isXXX()用于返回布尔类型。

```
package com.group;
public class Add{
    private int numA;
    private int numB;
    public int getNumA(){
        return numA;
    }
    public int getNumB(){
        return numB;
    }
    public void setNumA(int val){
        numA = val;
    }
    public void setNumB(int val){
        numB = val;
    }
    public int getSum(){
        return numA+numB;
    }
}
```

Add.java

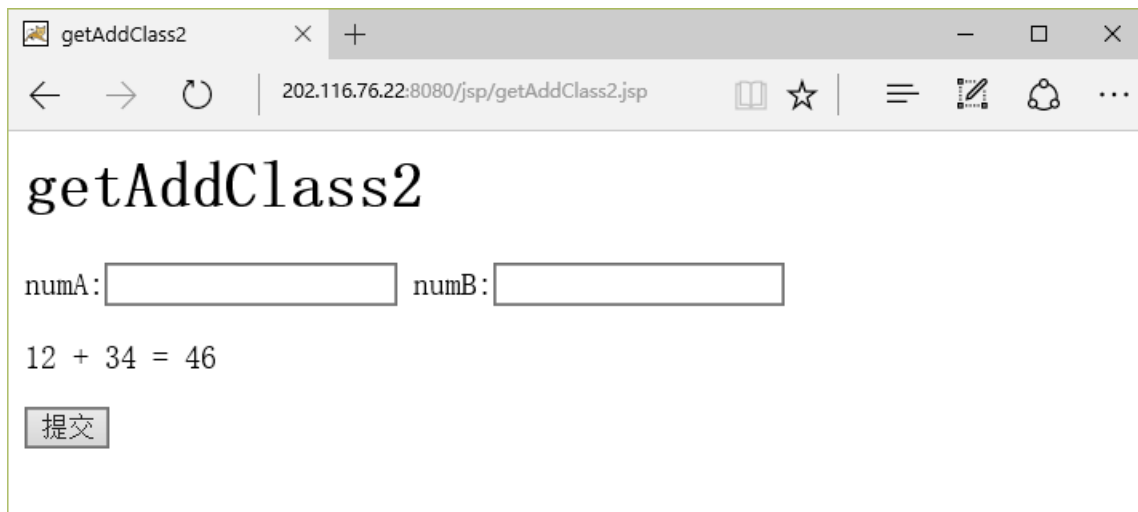
## getSumClass1.jsp

```
<%@ page contentType="text/html; charset=gb2312"%>
<html>
<head>
<title>getAddClass1</title>
</head>
<body>
  <h1>getAddClass1</h1>
  <jsp:useBean id="add" scope="page" class="com.group.Add" />
  <jsp:setProperty name="add" property="numA" value="33"/>
  <jsp:setProperty name="add" property="numB" value="45"/>
  <jsp:getProperty name="add" property="sum"/>
</body>
</html>
```

page:	当前页面有效（刷新页面后失效）
request:	当前请求有效，效果与page相同
session:	当前会话有效（sessionID不同时失效）
application:	当前应用有效（重启Web服务器时失效）
/	* 只有失效后才会产生新对象并重新执行。



```
<%@ page contentType="text/html; charset=gb2312"%>
<html>
<head>
<title>getAddClass2</title>
</head>
<body>
  <h1>getAddClass2</h1>
  <jsp:useBean id="add" scope="page" class="com.group.Add" />
  <form method="post" action="getAddClass2.jsp">
    <jsp:setProperty name="add" property="numA"/>
    <jsp:setProperty name="add" property="numB"/>
    <p>numA:<input type="text" name="numA"/>
      numB:<input type="text" name="numB"/></p>
    <p><jsp:getProperty name="add" property="numA"/>
      + <jsp:getProperty name="add" property="numB"/>
      = <jsp:getProperty name="add" property="sum"/></p>
    <input type="submit" name="submit" value="提交"/>
  </form>
</body>
</html>
```



getAddClass2

numA:  numB:

12 + 34 = 46

```
<%@ page contentType="text/html; charset=gb2312"%>
<html>
<head>
<title>getAddClass3</title>
</head>
<body>
  <h1>getAddClass3</h1>
  <jsp:useBean id="add" scope="page" class="com.group.Add" />
  <% add.setNumA(33);add.setNumB(45);%>
  <p><% out.print(add.getNumA()+" + " + add.getNumB() + " = " + add.getSum()); %>
    </p>
</body>
</html>
```



# 附录1、利用底层功能上传文件

```
<%@ page language="java" import="java.util.*,java.io.*"
    contentType="text/html; charset=utf-8"%>
<%request.setCharacterEncoding("utf-8");%>
<%!// 读取http请求的正文并转化为字符串。
String getRequestBody(HttpServletRequest request)throws Exception{
    int contentLength = request.getContentLength();
    DataInputStream in
        = new DataInputStream(request.getInputStream());
    byte dataBytes[]=new byte[contentLength];
    int bytesRead = 0;
    int totalBytesRead = 0;
    while(totalBytesRead<contentLength){
        bytesRead = in.read(dataBytes,totalBytesRead,contentLength);
        totalBytesRead += bytesRead;
    }
    String requestBody = new String(dataBytes,"ISO-8859-1");
    return requestBody;
}
// 找出用来把正文划分为多个部分的边界字符串
String getBoundary(String contentType){
    int lastIndex = contentType.lastIndexOf("=");
    return contentType.substring(lastIndex+1,contentType.length());
}
```

// 找出正文的一个划分部分的头部中找出文件名

```
String getFileName(String partOfBody) throws Exception {  
    final String RET = "\r\n";  
    String fileName = "";  
    int lineStart = 0;  
    int lineEnd = partOfBody.indexOf(RET);  
    while(lineEnd>=0){  
        String line = partOfBody.substring(lineStart, lineEnd);  
        lineStart = lineEnd+RET.length();  
        lineEnd = partOfBody.indexOf(RET,lineStart);  
        if(line.isEmpty()){  
            break;  
        }  
        else{  
            if(line.toLowerCase().indexOf("filename")>=0){  
                fileName = line.substring(line.lastIndexOf("=")+1).trim();  
                fileName = fileName.substring(1, fileName.length()-1);  
                fileName = new String(fileName.getBytes("ISO-8859-1"), "UTF-8");  
                break;  
            }  
        }  
    }  
    return fileName;  
}
```



```

boolean saveFile(String contentPart,String path,String fileName){
    final String RET = "\r\n"; int bodyStart = 0;int lineStart = 0;
    int lineEnd = contentPart.indexOf(RET);
    while(lineEnd>=0){
        String line = contentPart.substring(lineStart, lineEnd);
        lineStart = lineEnd+RET.length();
        lineEnd = contentPart.indexOf(RET,lineStart);
        if(line.isEmpty()){
            bodyStart = lineStart;
            break;
        }
    }
    if(bodyStart==0) return false;
    String body=contentPart.substring(bodyStart);
    try{
        FileOutputStream fileOut
            = new FileOutputStream(path+fileName);
        fileOut.write(body.getBytes("ISO-8859-1"));
        fileOut.flush();
        fileOut.close();
        return true;
    }
    catch(Exception e){
        return false;
    }
}

```

%>

```

<% String msg = "";
    if(request.getMethod().equalsIgnoreCase("post")){
        String requestBody = getRequestBody(request);
        String contentType = request.getContentType();
        String boundary = getBoundary(contentType);
        int startPos = requestBody.indexOf(boundary);
        while(startPos>=0){
            int endPos = requestBody.indexOf(boundary,startPos+1);
            String partOfBody = "";
            if(endPos<0){
                partOfBody = requestBody.substring(startPos);
            }
            else{
                partOfBody= requestBody.substring(startPos
                                                    + boundary.length()+2,endPos);
            }
            String fileName = getFileName(partOfBody);
            if(!fileName.isEmpty()){
                if(saveFile(partOfBody,"c:\\temp\\",fileName)){
                    msg = fileName+"上传成功! ";
                }
                else{
                    msg = fileName+"上传失败! ";
                }
            }
            if(endPos<0) break;
            startPos = partOfBody.indexOf(boundary,endPos);
        }
    }
%>

```

```
<!DOCTYPE HTML>
<html>
<head>
<title>uploadFile.jsp</title>
</head>
<body>
<form action="uploadFile.jsp" method="post" enctype="multipart/form-data">
    选择上传文件: <br /> <input type="file" name="file1" /><br />
    <br /> <input type="submit" name="submit" value="submit" /><br />
    <br />
</form>
<%=msg%>
</body>
</html>
```