# Jsp与javabean学习笔记

# JSP

## C:\Users\Administrator\Desktop\jsp学习\1. jsp1.JPG1. jsp示例­­­­­­­

## 2. jsp源码提供的对象名称



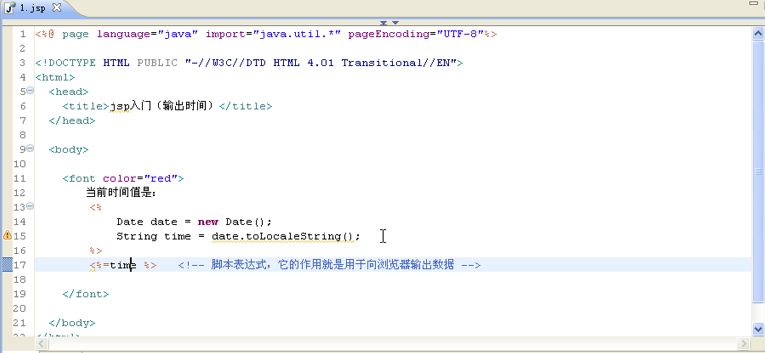
## C:\Users\Administrator\Desktop\jsp学习\3. jsp与servlet的关系（jsp显示，servlet控制）.JPG3. jsp与servlet的关系

Jsp用作输出，servlet用作 控制（MVC模型）

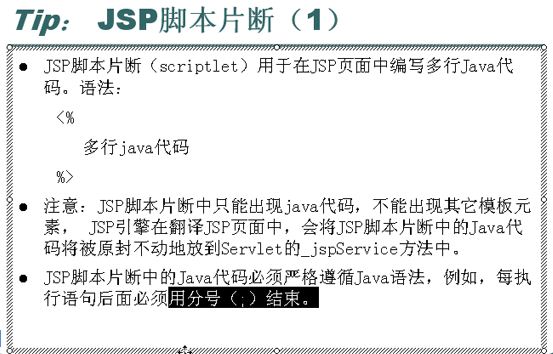
## 4. jsp语法

#### C:\Users\Administrator\Desktop\jsp学习\4. jsp语法（jsp模板元素）.JPG1.jsp模板元素

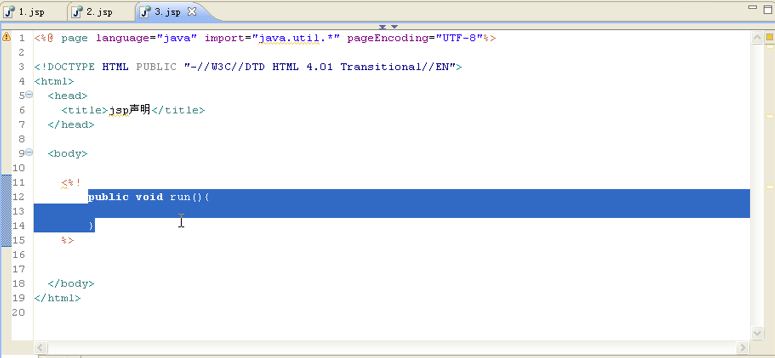
#### 2. jsp脚本表达式

脚本表达式：jsp中的 一种输出方式 ，例如：<%= time %> 后面不能加分号。

#### 3. jsp脚本片段

jsp中存放java代码的模块，不能出现其他元素，用 ; 结尾,可以几个连起来使用。

#### 4. jsp申明

为了避免service方法里面套方法。给servlet自己的方法。（或者静态模块，属性字段）

#### 5. jsp注释

jsp注释： <%-- 注释 --%>

注意：这个注释可以被jsp编译的时候识别

HTML注释： <!—注释 -->

注意：这个注释内容会发给web浏览器，再被浏览器识别，不好，占用了流量资源。

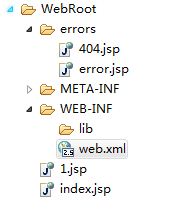
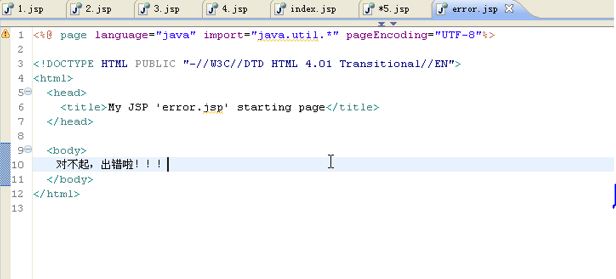
## 5. jsp指令

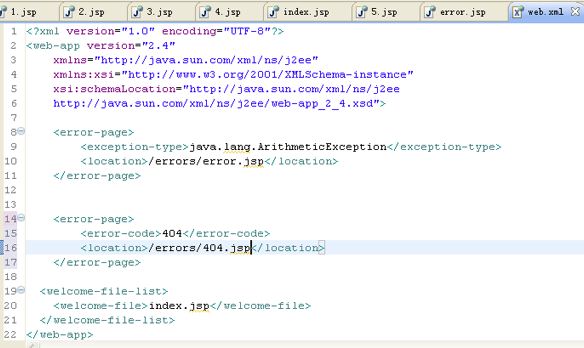
#### 1. jsp page 指令



##### 注意. 错误处理

首先在WebRoot目录下创建errors 的 folder，创建handle error的jsp文件；

其次在web.xml中配置出错的情况（可以用异常类型，或 错误代码）



#### 2. jsp include 指令

例如，统一的页头和页脚可以写成public folder下的jsp文件，然后其他jsp资源文件直接调用。

Page指令：

1 <%@ page

2 [ language="java" ]

3 [ extends="package.class" ]

4 [ import="{package.class | package.\*}, ..." ]

5 [ session="true | false" ]

6 [ buffer="none | 8kb | sizekb" ]

7 [ autoFlush="true | false" ]

8 [ isThreadSafe="true | false" ]

9 [ info="text" ]

10 [ errorPage="relative\_url" ]

11 [ isErrorPage="true | false" ]

12 [ contentType="mimeType [ ;charset=characterSet ]" | "text/html ; charset=ISO-8859-1" ]

13 [ pageEncoding="characterSet | ISO-8859-1" ]

14 [ isELIgnored="true | false" ]

15 %>

* 注：errorPage

<%@ page language="java" import="java.util.\*" **errorPage="/ErrorPage/error.jsp"** pageEncoding="UTF-8"%>

page指令的**errorPage属性**指明了出错后跳转到**"/ErrorPage/error.jsp"，**error.jsp页面代码如下：

<%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>

<html>

<head>

<title>错误信息友好提示页面</title>

</head>

<body>

对不起，出错了，请联系管理员解决！

</body>

</html>

Include指令：

* taglib指令

#### 3. jsp 九大隐式对象

* **Jsp out 对象：**

**Jsp out对象的作用：用于输出（在缓存中，输出比 response.getWriter().write() 输出 慢）**

* **Jsp pageContext 对象：**

**pageContext对象的作用：**

1. 封装了对其他8大隐式对象的使用；(在调用某个域的变量的时候，统一通过pagecontext对象的调用来完成。)

2. 自身是一个域对象，可以保存数据；（也叫：page域，本页面有效。）

3.封装了web开发的常见操作（forward, include等）

例如：

* **存变量在session域中。**

<%

pageContext.setAttribute(“data”, “abc”, PageContext.SESSION\_SCOPE);

%>

**存数据到session域内**

session.setAttribute("name3", "玄天邪帝");

* **从session域取出变量**

<%= pageContext.getAttribute(“data”, PageContext.SESSIONz\_SCOPE) %>

**第二种做法：使用**session.getAttribute("attributeName");**去取出session对象中值**

* **从四个域（page域，request域， session域， application域）中寻找变量。**

<%

pageContext.findAttribute(“data”);

%>

注1：**jsp的九大隐式对象**

Request, response, session, page, application, exception, out, pageContext, config

<%

session.setAttribute("name", "session对象");//使用session对象,设置session对象的属性

out.print(session.getAttribute("name")+"<br/>");//获取session对象的属性

pageContext.setAttribute("name", "pageContext对象");//使用pageContext对象,设置pageContext对象的属性

out.print(pageContext.getAttribute("name")+"<br/>");//获取pageContext对象的属性

application.setAttribute("name", "application对象");//使用application对象,设置application对象的属性

out.print(application.getAttribute("name")+"<br/>");//获取application对象的属性

out.print("Hello Jsp"+"<br/>");//使用out对象

out.print("服务器调用index.jsp页面时翻译成的类的名字是："+page.getClass()+"<br/>");//使用page对象

out.print("处理请求的Servlet的名字是："+config.getServletName()+"<br/>");//使用config对象

out.print(response.getContentType()+"<br/>");//使用response对象

out.print(request.getContextPath()+"<br/>");//使用request对象

%>

注2：**jsp的域对象：**

pageContext 域: 本页面可以使用； （也叫：page 域） （页面范围内共享）

request 域：本页面可以使用，forward到别的页面也可以使用；（请求范围内共享）

session 域：会话范围内都可以取出来（一个人共享），只要不关闭浏览器，任何servlet,jsp 都可以使用；  
application 域：所有人可以共享。

注：

* 使用request的forward方法跳转到了另一个page界面，不属于一个pageContext域。
* 使用request的forward方法跳转，不会改变url的值，属于一个request域。（服务器跳转）
* 使用超链接跳转，会改变url的值，属于一个session域。（客户端跳转。）

<a href="${pageContext.request.contextPath}/requestScopeDemo04.jsp">

跳转到requestScopeDemo04.jsp</a>

* 当开启多个浏览器窗口时，仍然可以保留数据，数据application域。

forward跳转是服务器跳转,地址栏不变。超链接是客户端跳转,地址栏改变

jsp四种属性范围的使用场合

　　1、request：如果客户向服务器发请求，产生的数据，用户看完就没用了，像这样的数据就存在request域,像***新闻数据***，属于用户看完就没用的。  
　　2、session：如果客户向服务器发请求，产生的数据，用户用完了等一会儿还有用，像这样的数据就存在session域中，像***购物数据***，用户需要看到自己购物信息，并且等一会儿，还要用这个购物数据结帐。  
　　3、application(servletContext)：如果客户向服务器发请求，产生的数据，用户用完了，还要给其它用户用，像这样的数据就存在application(servletContext)域中，像***聊天数据***。

## 6. jsp标签

#### Include标签：用于引入其他元素：例如 页头、页脚（动态引用）

**<jsp:include page="relativeURL | <%=expression%>" flush="true|false" />**

**e.g.**

<body>

<%--使用jsp:include标签引入其它JSP页面--%>

<jsp:include page="/jspfragments/head.jsp"/>

<h1>网页主体内容</h1>

<jsp:include page="/jspfragments/foot.jsp"/>

</body>

#### Forward标签：用于转发页面。

**<jsp:forward page="relativeURL | <%=expression%>" />**

e.g. <jsp:forward page="/forwarddemo02.jsp"/>

此跳转属于服务器端跳转。只要是服务器端跳转，则地址栏永远没有变化。

#### Param标签：用于传递数据。

当使用<jsp:include>和<jsp:forward>标签引入或将请求转发给其它资源时，可以使用<jsp:param>标签向这个资源传递参数。  
　　语法1：  
   **<jsp:include page="relativeURL | <%=expression%>">**  
**<jsp:param name="parameterName" value="parameterValue|<%= expression %>" />**  
**</jsp:include>**

**e.g：**

**<%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>**

**<h1>JspIncludeTagDemo03.jsp</h1>**

**<hr/>**

**<jsp:include page="/jspfragments/Inc.jsp">**

**<jsp:param name="parm1" value="hello" />**

**<jsp:param name="parm2" value="gacl" />**

**</jsp:include>**

1 <%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>

2 <h1>接收从JspIncludeTagDemo03.jsp页面中传递过来的参数：</h1>

3 <h2><%=request.getParameter("parm1")%></h2>

4 <h2><%=request.getParameter("parm2")%></h2>

　　语法2：  
    **<jsp:forward page="relativeURL | <%=expression%>">**  
**<jsp:param name="parameterName" value="parameterValue|<%= expression %>" />**  
**</jsp:include>**

## ****7. el表达式****

## 8.自定义标签

自定义标签主要用于移除Jsp页面中的java代码。

**开发步骤：**

1. **编写一个实现Tag接口的Java类(标签处理器类)**

2. 在WEB-INF/目录下新建tld文件，在tld文件中对标签处理器类进行描述

3. 使用指令引入要使用的标签库。

**<%@taglib uri="标签库的uri"  prefix="标签的使用前缀"%>**

**e.g:**

* gacl.tld文件:

<!-- 为自定义标签库设置一个uri，uri以/开头，/后面的内容随便写，如这里的/gacl ，

在Jsp页面中引用标签库时，需要通过uri找到标签库

在Jsp页面中就要这样引入标签库：<%@taglib uri="/gacl" prefix="gacl"%> -->

<uri>/gacl</uri>

为标签处理器类配一个标签名，在Jsp页面中使用标签时是通过标签名来找到要调用的标签处理器类的

通过viewIP就能找到对应的me.gacl.web.tag.ViewIPTag类

<name>viewIP</name>

* 在JSP文件中使用自定义标签：

<!-- 使用taglib指令引用gacl标签库，标签库的前缀(prefix)可以随便设置，如这里设置成 prefix="xdp" -->

<%@taglib uri="/gacl" prefix="xdp"%>

<!--使用自定义标签viewIP --!>

<xdp:viewIP/>

# Javabean

## 1. Javabean简介：

public class Person {

//------------------Person类封装的私有属性---------------------------------------

private String name;

private String sex;

//------------------Person类的无参数构造方法---------------------------------------

public Person() {

}

//------------------Person类对外提供的用于访问私有属性的public方法---------------------------

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getSex() {

return sex;

}

public void setSex(String sex) {

this.sex = sex;

}

}

## 2. Javabean的属性：

**JavaBean的属性可以是任意类型，并且一个JavaBean可以有多个属性**。

* 每个属性通常都需要具有相应的setter、 getter方法，setter方法称为属性修改器，getter方法称为属性访问器。书写规范：camelCase
* 一个JavaBean的某个属性也可以只有set方法或get方法，这样的属性通常也称之为只写、只读属性。

## 3. 在JSP中使用JavaBean

　　JSP技术提供了三个关于JavaBean组件的动作元素，即JSP标签，它们分别为：

* <jsp:useBean>标签：用于在JSP页面中查找或实例化一个JavaBean组件。
* <jsp:setProperty>标签：用于在JSP页面中设置一个JavaBean组件的属性。
* <jsp:getProperty>标签：用于在JSP页面中获取一个JavaBean组件的属性。

#### 3.1 <jsp:useBean>标签

*<jsp:useBean id="beanName" class="package.class" scope="page|request|session|application"/>*

"**id**"属性用于指定JavaBean实例对象的引用名称和其存储在域范围中的名称。  
"**class**"属性用于指定JavaBean的完整类名（即必须带有包名）。  
"**scope**"属性用于指定JavaBean实例对象所存储的域范围，其取值只能是page、request、session和application等四个值中的一个，其默认值是page。

注：

带标签体的<jsp:useBean>标签

语法：  
*<jsp:useBean ...>   
        Body   
    </jsp:useBean>*功能：  
    Body部分的内容只在<jsp:useBean>标签创建JavaBean的实例对象时才执行。这种做法用得不多，了解一下即可

#### 3.2 <jsp:setProperty>标签

**语法格式一：**  
　　　　**<jsp:setProperty name="beanName" property="propertyName" value="string字符串"/>**

***字符串会自动转化成javabean中对应的类型。***

**语法格式二：**  
　　　　**<jsp:setProperty name="beanName" property="propertyName" value="<%= expression %>" />**

***用脚本表达式来代替 复合数据类型（有别于基本数据类型。）***

**语法格式三：**  
　　　　**<jsp:setProperty name="beanName" property="propertyName" param="parameterName"/>**

***标签可以使用请求参数为bean的属性赋值***

**语法格式四：**  
　　　　**<jsp:setProperty name="beanName" property= "\*" />**

***property="\*"代表bean的所有属性，一次性赋值所有变量，按顺序排列。***

<jsp:useBean id="person" class="gacl.javabean.study.Person" scope="page"/>

<%--

使用jsp:setProperty标签设置person对象的属性值

jsp:setProperty在设置对象的属性值时会自动把字符串转换成8种基本数据类型

但是jsp:setProperty对于复合数据类型无法自动转换

--%>

<jsp:setProperty property="name" name="person" value="白虎神皇"/>

<jsp:setProperty property="sex" name="person" value="男"/>

<%-- 对于复合数据类型Date，没有自动转换数据类型的功能，需要使用脚本表达式--%>

<jsp:setProperty property="birthday" name="person" value="<%=new Date()%>"/>

#### 3.3 <jsp:getProperty>标签

* <jsp:getProperty name="beanInstanceName" property="PropertyName" />
* 这个标签调用javabean的 getter()方法，将参数的值转化为String.
* e.g:

<body>

<%--使用jsp:getProperty标签获取对象的属性值 --%>

<h2>姓名：<jsp:getProperty property="name" name="person"/></h2>

<h2>性别：<jsp:getProperty property="sex" name="person"/></h2>

<h2>出生日期：<jsp:getProperty property="birthday" name="person"/></h2>

</body>