# jsp 语法基础：

jsp的三种注释

三种scriptlet及其区别

page指令

jsp中连接数据库

包含指令

跳转指令

## 具体内容：

任何语言都有自己的语法，java有，jsp虽然是在java上的一种应用，但是依然有其自己扩充的语法，而且在jsp中，所有的java语言都可以使用。

#### 注释

在jsp中注释分为两类：

·显式注释：直接使用html风格的注释<!-- -->

·隐式注释：直接使用java的注释

|·jsp自己的注释：<%-- --%>

观察三种注释的区别

显式注释在查看网页源文件的时候可以看到注释内容而隐式注释不行

#### scriptlet

先从单词意思来看：script：脚本，let：在java里面表示小程序的意思，所以综合来看叫做脚本小程序

从使用上来讲，在整个jsp中script有三种形式：

·<%%>

·<%!%>

·<%=%>

第一种scriptlet：

<%%> 一般使用定义变量，编写语句

范例：在scriptlet中定义变量，编写语句，编写一个for循环

|  |
| --- |
| <%  **int** sum = 0;  **for**(**int** i =0;i<100;i++){  sum += i;  }  out.print("<h1>sum="+sum+"<h21/>");  %> |

第二种scriptlet：<%!%>

可以在此scriptlet中声明全局常量，编写方法，编写类，但是一般情况下，jsp中不会直接定义一个类，而且在此scriptlet中不会编写语句

·定义全局常量

|  |
| --- |
| <%!**public** **static** **final** String DBDRIVER = "oracle.jdbc.driver.OracleDriver";  **public** **static** **final** String DBURL = "jdbc:oracle:thin:@192.168.10.250:1521:wance";%>  <%  out.print("<h1>"+DBDRIVER+"</h1>");  %>  <%  out.print("<h1>"+DBURL+"</h1>");  %> |

也可以在<%!%> 编写若干方法，有时也会采用这种方式

·编写方法：

|  |
| --- |
| <%!**public** **int** add(**int** i, **int** j) {  **return** i + j;  }%>  <%  out.print("<h1>" + add(200, 300) + "</h1>");  %> |

第三种scriptlet：<%=%>

此种scriptlet有一个别名，称为表达式输出，可以直接使用此种形式的scriptlet输出一个变量或者一个具体的内容

范例：

|  |
| --- |
| <%!**public** **int** add(**int** i, **int** j) {  **return** i + j;  }%>  <%  **int** i = 10;  String name="Lisay";  %>  <h1><%=add(300,200) %></h1><br>  <h1><%=i %></h1><br>  <h1><%=name %></h1><br> |

我们已经知道的有两种向页面输出的方式，一种是out.print(),另一种就是使用表达式输出，对于这两种方式，哪一种会更好？

#### 使用哪种输出方式更好

out.print(),和表达式输出（<%=%>）功能一样,如果想要知道哪种方式更加好，则可以通过一个范例来说明

·打印一个100\*100的表格

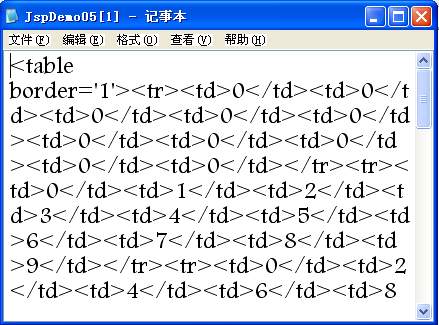
如果想要完成此功能，则肯定要使用for循环，两层循环，第一层控制行，第二层控制列

第一种方式：

使用out.print实现：

|  |
| --- |
| <%  out.print("<table border='1'>");  **for**(**int** i = 0;i<100;i++){  out.print("<tr>");  **for**(**int** j = 0;j<100;j++){  out.print("<td>");  out.print(i\*j);  out.print("</td>");  }  out.print("</tr>");  }  out.print("</table>");  %> |

此时已经完成了表格的打印。但是此时我们观察生成后的代码：

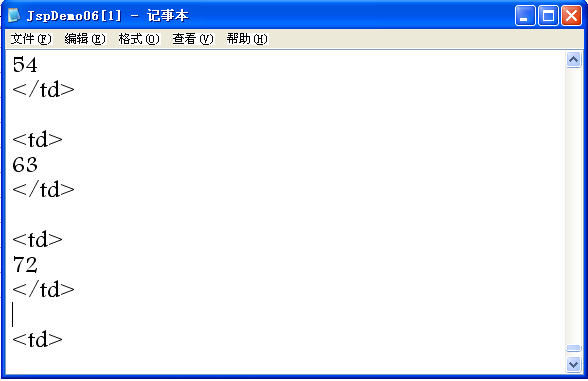


此代码没有任何的缩进，非常混乱，而且此时想要为表格增加边框的话，需要使用border属性

现在使用第二种方式：使用表达式输出

|  |
| --- |
| <table border=*'1'*>  <%  **for** (**int** i = 0; i < 10; i++) {  %>  <tr>  <%  **for** (**int** j = 0; j < 10; j++) {  %>  <td>  <%  out.print(i \* j);  %>  </td>  <%  }  %>  </tr>  <%  }  %>  </table> |

此时功能已经完成，而且源文件代码缩进非常有美感，而且代码本身已经把html和java代码区分开来。



表达式输出还有一个好处：可以方便的使用工具修改页面属性，而使用out.print 输出的话，工具无法识别页面。

经过以上比较之后，发现使用表达式输出明显更好，所以现在给你们制定一个严格的要求，在以后的JSP页面中绝对不能出现out.print() 语句

#### 动态打印表格

用户通过表单，输入行和列，之后想jsp页面传递这两个参数，打印表格

printable.html

|  |
| --- |
| <form action=*"printTable.jsp"* method=*"post"*>  <table>  <caption>打印表格</caption>  <tr>  <td>输入行：<input type=*"text"* name=*"rows"*/></td>  </tr>  <tr>  <td>输入列：<input type=*"text"* name=*"cols"*/></td>  </tr>  <tr>  <td><input type=*"submit"* value=*"提交"*/><input type=*"reset"* value=*"重置"*/></td>  </tr>  </table>  </form> |

printable.jsp

|  |
| --- |
| <%  **int** rows = Integer.parseInt(request.getParameter("rows"));  **int** cols = Integer.parseInt(request.getParameter("cols"));  %>  <table border=*"1"* bordercolor=*"green"*>  <%  **for** (**int** i = 0; i < rows; i++) {  %>  <tr>  <%  **for** (**int** j = 0; j < cols; j++) {  %>  <td><%=i \* j%></td>  <%  }  %>  </tr>  <%  }  %>  </table> |

以上代码实现了动态打印表格，但是你觉得这个程序好吗？

现在在html页面中加入了javascript验证，如果现在屏蔽javascript的功能，那么加入javascript验证就没有效果，所以我们应该在服务端加入验证

|  |
| --- |
| <%  **int** rows = 0;  **int** cols =0;  **try**{  rows = Integer.parseInt(request.getParameter("rows"));  cols = Integer.parseInt(request.getParameter("cols"));  }**catch**(Exception e){  }  %>  <table border=*"1"* bordercolor=*"green"*>  <%  **for** (**int** i = 0; i < rows; i++) {  %>  <tr>  <%  **for** (**int** j = 0; j < cols; j++) {  %>  <td><%=i \* j%></td>  <%  }  %>  </tr>  <%  }  %>  </table> |

#### page指令

page 表示当前jsp页面，可以理解为一个对象，就是说可以把jsp当做一个对象来看待。

page指令的主要功能有以下两个：

·设置MIME类型

·导包指令

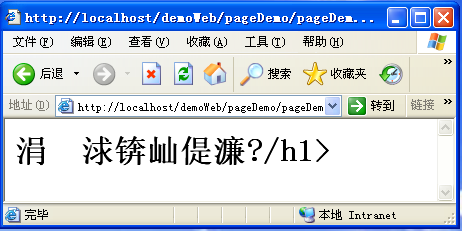
MIME：多功能网际邮件扩充协议

##### 设置MIME类型

假设现在建立一个jsp页面，要求里面有中文

|  |
| --- |
| <h1>中国，您好</h1> |

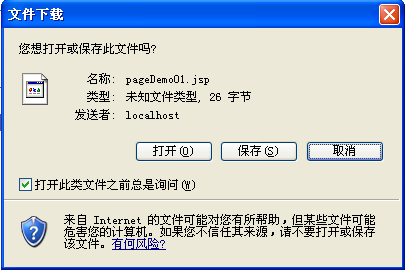
显示效果：



之所以出现乱码的原因，是由于页面的编码和浏览器的显示编码不一致造成的，所以，此时为了正确的在浏览器中显示效果，最好是在页面加载的时候就已经设置好自己所需要的编码了，这个时候，就必须使用page指令，指定MIME类型。

|  |
| --- |
| <%@page contentType=*"****text/html; charset=UTF-8****"* %>  <h1>中国，您好</h1> |

以上红色就是指定MIME类型，是整个页面按照html的风格显示，字符集就是utf-8，但是很多同学在设置的时候由于设置不当，会出现以下的情况：



凡是出现以上情况的，肯定是MIME类型设置错误，就是text写成test

那么MIME类型还有那些呢？

我们 之前说过 \*.html 和\*.htm的页面处理方式完全一样，这是因为其内部的MIME处理方式就是一样的。

conf/web.xml:

|  |
| --- |
| <mime-mapping>  <extension>htm</extension>  <mime-type>text/html</mime-type>  </mime-mapping>  <mime-mapping>  <extension>html</extension>  <mime-type>text/html</mime-type>  </mime-mapping>  <mime-mapping> |

只要合理的应用了MIME类型，则一个页面可以显示成任意的风格，现在我们将jsp页面显示成一个文档的效果

|  |
| --- |
| <%@page contentType=*"application/msword; charset=UTF-8"* %>  <h1>中国，您好</h1> |

##### 使用page指令导包

在page指令中，使用import进行导包，import指令时page指令中唯一可以设置多次的指令，目的是导包

|  |
| --- |
| <%@page contentType=*"text/html; charset=UTF-8"* %>  <%@page import=*"java.util.\*,java.io.\*"* %>  <%@page import=*"java.sql.\*"* %> |

既然jsp可以导入sql包，那么是不是意味着可以连接数据库？

jsp是可以连接到数据库，但是如果想要连接的话，必须在oracle的驱动包，在tomcat中的lib目录下面加入oracle的驱动程序包即可，服务器必须重启

|  |
| --- |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <%@page import=*"java.util.\*,java.io.\*"*%>  <%@page import=*"java.sql.\*"*%>  <%!**public** **static** **final** String DBDRIVER = "oracle.jdbc.driver.OracleDriver";  **public** **static** **final** String DBURL = "jdbc:oracle:thin:@192.168.10.250:1521:wance";  **public** **static** **final** String DBNAME = "system";  **public** **static** **final** String PASSWORD = "wance";%>  <%  Connection con = **null**;  PreparedStatement pstamt = **null**;  ResultSet res = **null**;  **try** {  Class.forName(DBDRIVER);  con = DriverManager.getConnection(DBURL, DBNAME, PASSWORD);  String sql = "select ID\_KEY,PRODUCT\_NUM,PRODUCT\_NM,PRODUCT\_OUT,CUSTOMER from jt39\_jjm\_customer";  pstamt = con.prepareStatement(sql);  res = pstamt.executeQuery();  %>  <center>  <h1>  商品消费查询  </h1>  <table border=*"1"* bordercolor=*"green"* width=*"70%"*>  <tr>  <th>  idk  </th>  <th>  商品编号  </th>  <th>  商品名称  </th>  <th>  消费数量  </th>  <th>  消费者  </th>  </tr>  <%  **while** (res.next()) {  **int** id\_key = res.getInt(1);  String product\_num = res.getString(2);  String product\_nm = res.getString(3);  **int** product\_out = res.getInt(4);  String customer = res.getString(5);  %>  <tr>  <td><%=id\_key%></td>  <td><%=product\_num%></td>  <td><%=product\_nm%></td>  <td><%=product\_out%></td>  <td><%=customer%></td>  </tr>  <%  }  } **catch** (Exception e) {  } **finally** {  **try** {  res.close();  pstamt.close();  con.close();  } **catch** (Exception e) {  }  ;  }  %>  </table>  </center> |

总结：

使用page指令可以设置MIME类型，所有的MIME类型都可以在tomcat下的conf文件夹中的web.xml文件中找到。

使用import导包指令

在jsp操作数据库

##### 最后两个知识点 包含指令和跳转指令

包含：include

跳转：forward

###### 包含指令：include

include 表示包含，一般的网页格式：

|  |  |
| --- | --- |
| 头信息 | |
| 操作菜单 | 中间内容可以随便更换 |
| 尾部信息 | |

现在如果想要实现以上的功能有两种方式：

·分别将头信息、操作菜单、尾部信息写在不同的页面中，用包含方式包含进来

·将头信息、操作菜单、尾部信息写同一个页面中

以上两种形式肯定第一种比较好使，因为维护的时候只需要维护包含文件

在jsp中，include包含指令有两种语句格式：

·@include指令

·<jsp:include>指令

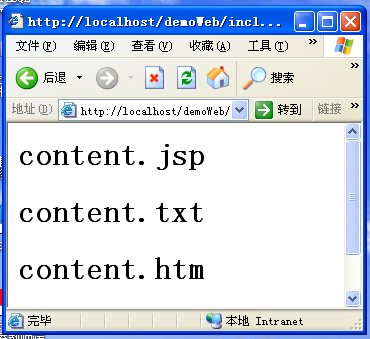
1 @include 指令

这个包含是属于静态包含，主页面不能向被包含的页面传递参数。可以包含任意格式的文件，只是把文件中的内容拿过来

现在新建了三个不同类型的文件，现在使用包含语句，将三个文件包含的主页面

|  |
| --- |
| <h1>content.jsp</h1>  <%@include file="content.txt"%>  <%@include file="content.htm"%>  <%@include file="content.inc"%> |

已经正常显示了被包含文件中内容，但是这种包含属于静态包含



2<jsp:include>指令

这种包含称为动态包含，如果被包含的界面是jsp页面，则先处理之后再将结果包含，

而如果是非jsp页面的话，则只把文件静态包含进来，功能与@include是一致的

此种包含也存在两种语法：

第一种：

|  |
| --- |
| <jsp:include page=”被包含的页面”/> |

第二种：

|  |
| --- |
| <jsp:include page=”被包含的页面”>  <!—向被包含的页面传递参数-->  <jsp:param name=”参数名称” value=”值”>  <jsp:param name=”参数名称” value=”值”>  <jsp:param name=”参数名称” value=”值”>  。。。。  </jsp:include> |

使用第一种包含：

|  |
| --- |
| <h1>content.jsp</h1>  <jsp:include page="content.txt"/>  <jsp:include page="content.htm"/>  <jsp:include page="content.inc"/> |



以上包含和静态包含语法很相识，效果也很相识，因为被包含的页面中并没有任何的动态页面，所以功能和静态包含页面很类似

现在，如果此时被包含的页面时动态的，就是除了可以包含页面之外，还可以像被包含的页面传递参数

第二种包含：

建立一个includeDemo03.jsp，里面可以向服务器发送接收参数的请求

|  |
| --- |
| <h1>第一个参数=<%=request.getParameter("ref1")%></h1>  <h1>第二个参数=<%=request.getParameter("ref2")%></h1> |

现在 此页面完成，建立includeDemo04.jsp,里面使用第二种语法包含includeDemo03.jsp

|  |
| --- |
| <h1>includeDemo04.jsp</h1>  <jsp:include page="includeDemo03.jsp">  <jsp:param name="ref1" value="JJm"/>  <jsp:param name="ref2" value="www.wanczy.com"/>  </jsp:include> |



现在我们向被包含的页面传递的参数在被包含的页面已经取得了，而且正常显示。

也可以传递变量：

|  |
| --- |
| <%@page contentType=*"text/html; charset=UTF-8"* %>  <h1>includeDemo04.jsp</h1>  <%  String name = "www.sohu.com";  %>  <jsp:include page=*"includeDemo03.jsp"*>  <jsp:param value=*"JJm"* name=*"ref1"*/>  <jsp:param value=*"*<%=name %>" name=*"ref2"*/>  </jsp:include> |

两种包含的区别：

我们介绍的两种包含到底是用哪一种？来看一下两端代码：

第一种方式

|  |
| --- |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <h1>includeDemo05.jsp</h1>  <%  **int** i =100;    %>  <h2>includeDemo05.jsp中i的值为：<%= i%></h2> |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <h1>includeDemo06.jsp</h1>  <%  **int** i =100;  %>  <h2>includeDemo06.jsp中i的值为：<%= i%></h2>  <%@include file = *"includeDemo05.jsp"*%> |

出现错误，提示重复定义了变量i，因为静态包含是将全部的内容包含进来，再一起处理

如果现在是动态包含，观察结果：

第二种

|  |
| --- |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <h1>includeDemo05.jsp</h1>  <%  **int** i =100;    %>  <h2>includeDemo05.jsp中i的值为：<%= i%></h2> |



发现结构可以正确显示，不会受到包含与被包含程序的影响，因为此种包含为动态包含，就是将各个页面单独处理后，在把结果包含。

相比之下，使用动态包含比使用静态包含更加方便，而且主页面可以向被包含的页面传递参数。

注意：第二种包含属于标签指令形式，此种指令在操作的时候必须完结。

##### 跳转指令：forward

跳转指令，当程序执行到这个指令的时候，程序会立刻无条件跳转，跳转指令也属于标签指令形式，所以一定要完结。语法如下：

·语法一：

<jsp:forward page=”跳转后的页面”/>

·语法二：

<jsp:forward page=”跳转后的页面” >

<jsp:param name=”参数名称” value=”参数值”/>

……

</jsp:forward>

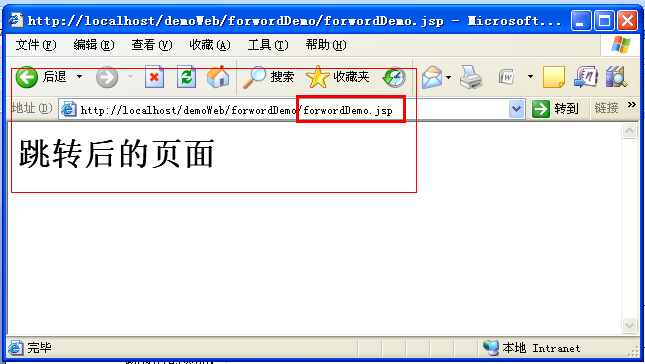
下面是有着两种语法完成跳转功能：

范例：跳转指令所在的页面

|  |
| --- |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <jsp:forward page=*"content.jsp"*/> |

跳转后的页面：

|  |
| --- |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <h1>跳转后的页面</h1> |



从页面显示效果来看，页面已经完成了跳转，但是地址栏并没有发生变化，因为此种跳转属于服务器端跳转，只要是服务器端跳转，地址栏永远不会发生变化

当然，也可以像跳转后的页面传递参数，同样在跳转后的页面使用request.getParameter()进行接收

第二种语法：

跳转后的页面：

|  |
| --- |
| <!-- 跳转后的页面 -->  <%@page contentType=*"text/html; charset=UTF-8"*%>  <h1><%= request.getParameter("ref1") %></h1>  <h1><%=request.getParameter("ref2") %></h1> |

被跳转页面：

|  |
| --- |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <jsp:forward page=*"forwardDemo02.jsp"*>  <jsp:param value=*"Hello World!!"* name=*"ref1"* />  <jsp:param value=*"www.wanczy.com"* name=*"ref2"* />  </jsp:forward> |



总结：

使用包含的时候一定要使用动态包含，因为处理更加方便

一定要清除的知道两种包含的区别

灵活的是由forward指令进行页面跳转

预习：

jsp的内置对象和属性范围，这个是jsp中的核心内容