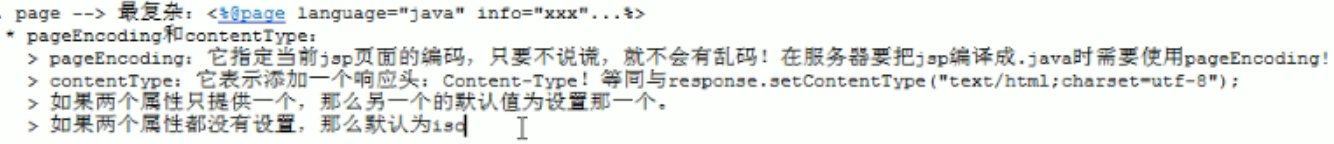
# Java Web（3）

## JSP三大指令

### Page（一大）



### Import

导包，可以出现多次。

### ErrorPage

当前页面如果抛出异常，那么要转发到哪一个页面，由errorPage来指定。

返回状态码：200

### isErrorPage

指定当前页面是否为处理错误的页面，当该属性为true时，这个页面会设置状态码为500，而且这个页面可以使用9大内置对象中的exception！

\*错误页面可以在web.xml中配置：



### Include（二大）

静态包含

* 与RequestDispatcher的include()方法的功能相似！
* <%@include %>它是在jsp编译成java文件时完成的！他们共同生成一个java（就是一个servlet） 文件，然后再生成一个class！
* RequestDispatcher的include()是一个方法，包含和被包含的是两个servlet，即两个.class！他们只是把响应的内容在运行时合并了。
* 作用：把页面分解了，使用包含的方式组合在一起，这样一个页面中不变的部分（或可重用的部分），就是一个独立的JSP，而我们只需要处理变化的页面。

### Taglib（导入标签库）（三大）

两个属性：

* Prefix:指定标签库在本页面中的前缀！由我们自己来起名称，以防止有多个标签库时名称冲突。
* Uri:指定标签库的位置！
* <%@taglib prefix=”s” uri=”/position...”%>
* 前缀的用法：<s:text>

## JSP九大内置对象

### Out(jspWriter)

等同于response.Writer()，用来向客户端发送文本数据。

### Config(ServletConfig)

### Page

当前JSP页面的“this” ,即当前对象。

它的引用类型是Object，其源码中有：Object page = this。

### PageContext

页面上下文对象。

1. 一个顶九个；
2. Servlet中有三大域对象，而JSP中有四大域，PageContext就是最后一个域对象！

》ServletContext：整个应用程序；

》session：整个会话（一个会话中只有一个用户）；

》request：一个请求链；

》pageContext：一个JSP页面！这个域是在当前jsp 页面和当前jsp页面中使用的标签之间共享数据！即标签与页面之间传输数据。

>域对象；

>代理其他域（往其他域中存东西）PageContext.setAttribute(“aaa”,”AAA”,PageContext.SESSION\_SCOPE);

>获取其他8个内置对象；

>（重要）全域查找：PageContext.findAttribute(“aaa”)，从小到大，依次查找。

### Exception(Throwable)

只有在错误页面中可以使用这个对象

### Request(HttpServletRequest)

即HttpServletRequest类的对象。

### Response(HttpServletResponse)

即HttpServletResponse的对象。

### Application(ServletContext)

即ServletContext类的对象。

### Session(HttpSession)

即HttpSession类的对象。

## JSP动作标签

1. Jsp的动作标签，与Html提供的标签有本质的区别

* 动作标签是由Tomcat（服务器）来解释执行，与Java代码一样，都是在服务器端执行的。
* Html由浏览器来执行。

1. <jsp:forward>:转发！与RequestDispatchar.foward()方法是一样的，一个是在Servlet中使用，一个是在Jsp中使用。

在页面之间传递参数：

* A页面

<jsp:forward page="b.jsp">

<jsp:param name="username" value="JOE"/>

<jsp:param name="password" value="123"/>

</jsp:forward>

* B 页面

<%

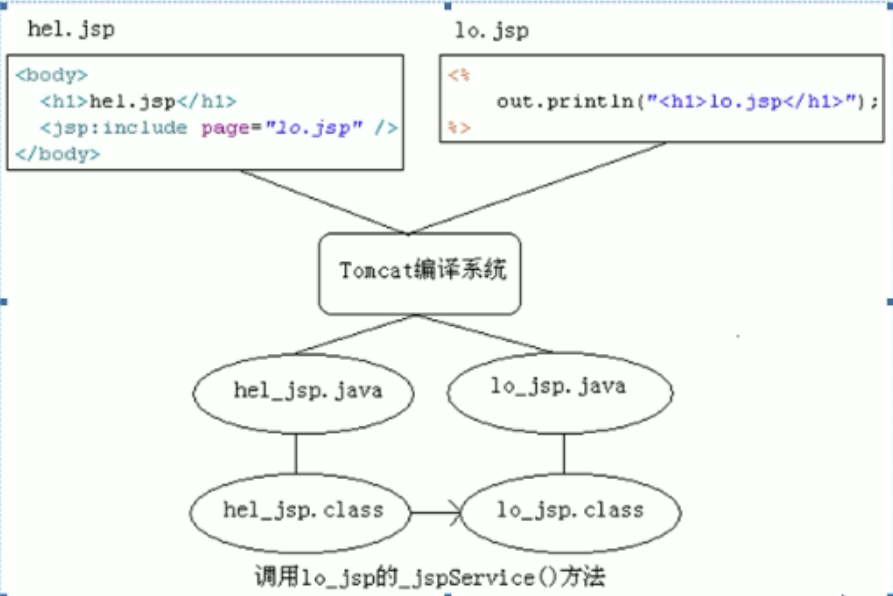
String username = request.getParameter("username");

String password = request.getParameter("password");

out.println(username+" , "+password);

%>

1. <jsp:include>:包含！同上。



1. <%@include %>与<jsp:include>的区别

前者：静态包含，生成一个java文件，生成一个class文件。

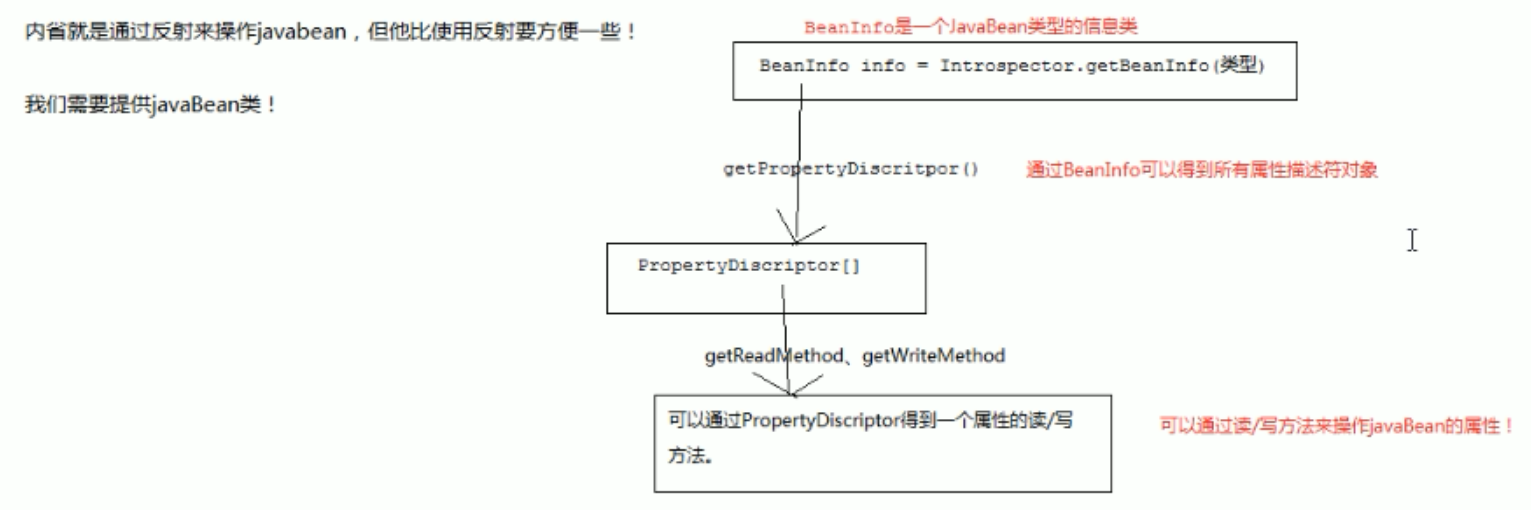
后者：动态包含，生成两个java文件，两个class文件，通过方法调用实现包含。

## JavaBean

### 规范



### 内省



\*步骤：

内省类 🡪 BeanInfo 🡪 属性描述符 🡪 属性的 get/set对应的Method 🡪 可以反射了

\*工具包（方便操作）：

Commons-beamutils，它是依赖内省完成！

需要的jar包：commons-beanutils.jar和commons-logging.jar

### 封装方法（重要）

见tools方法类。

## EL（表达式语言）

${变量名}

### EL的11个内置对象

11个内置对象中，其中10个是map，pageContext不是map，它就是pageContext类型，一个顶9个。

1. PageScope
2. requestScope
3. sessionScope
4. applicationScope
5. param

类型：Map<String,String>，分别为参数名和参数值；

适应于单值的参数。

1. paramValues

类型：Map<String,String[]>,分别为参数名和多个参数值；

适用于多值的参数。

1. header

同param.

1. headerValues

同paramValues.

1. initParam

获取<context-param>中的参数

1. cookie

类型：Map<String,Cookie>,其中key是cookie的name,value是cookie的对象。

取值方法：${cookie.username.value}

1. pageContext;

类型：pageContext

使用：${pageContext.request.contextPath}

## EL函数库（由JSTL提供）

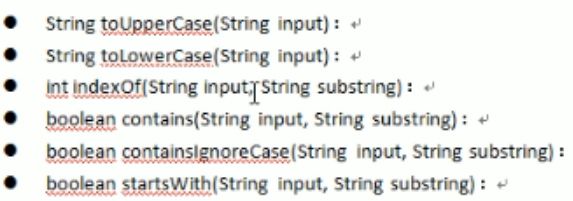
1. 导入JSTL提供的jar包
2. 

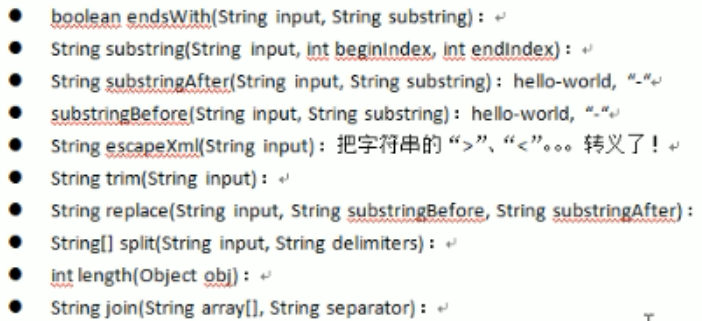
输出字符串之前，先将字符串用此函数处理，防止输出HTML标签，出现不可知 的错误。

1. 

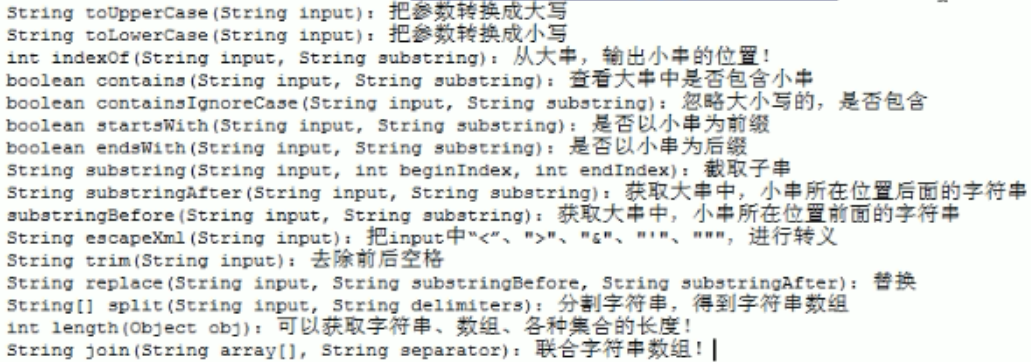
去除前后的空格。

1. 
2. 其他方法：





1. 总结：



## 自定义EL函数

* 写一个java类，类中可以定义0~N个方法，但必须是static,必须有返回值；
* 在WEB-INF目录下创建一个tld文件

<**function**>  
 <**name**>fun</**name**>  
 <**function-class**>cn.itcast.fn.MyFunction</**function-class**>  
 <**function-signature**>java.lang.String fun()</**function-signature**>  
</**function**>

* 在jsp页面中导入标签库

<%@**taglib prefix**="**it**" **uri**="**/WEB-INF/tlds/itcast.tld**" %>

* 在jsp页面中使用自定义的函数

${it:fun()}