一.指令

1.指令写法:<%@ 指令名字 %>, 如下的page指令写法:

|  |
| --- |
| <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%> |

二.page指令

1.page指令的写法: <%@ page ...>

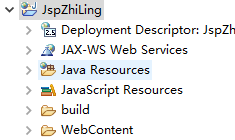
2.page指令的language属性, language=”java”, 表明jsp页面中可以写java代码。

3.page指令的contentType属性, contentType="text/html; charset=UTF-8", 指明文件的内容类型是一个简单文本、html网页, 内容编码是UTF-8。

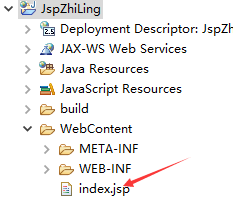
4.page指令的pageEncoding属性, pageEncoding="UTF-8", jsp内容编码是UTF-8。

5.新建一个名称叫做JspZhiLing的Web工程

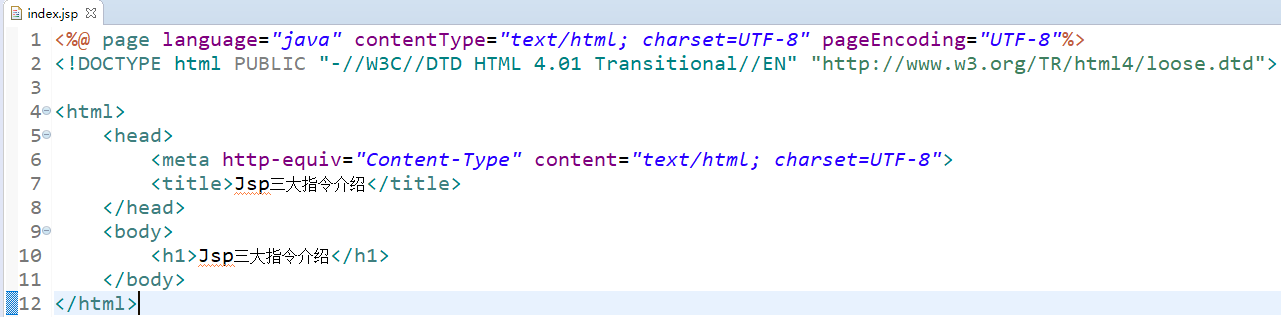
(1).新建一个名称叫做JspZhiLing的Web工程



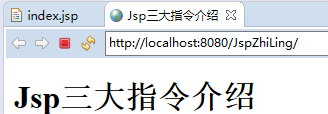
(2).在WebContent下新建一个index.jsp



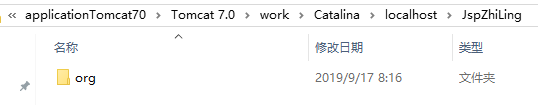
(3).编写index.jsp



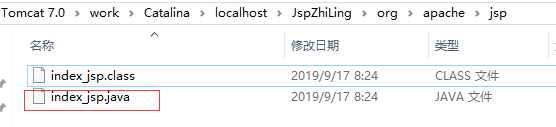
(4).部署运行



(5).在Tomcat的work-->localhost目录下找到项目



(6).进入目录找到index\_jsp.java, 我们的index.jsp被翻译成了index\_jsp.java



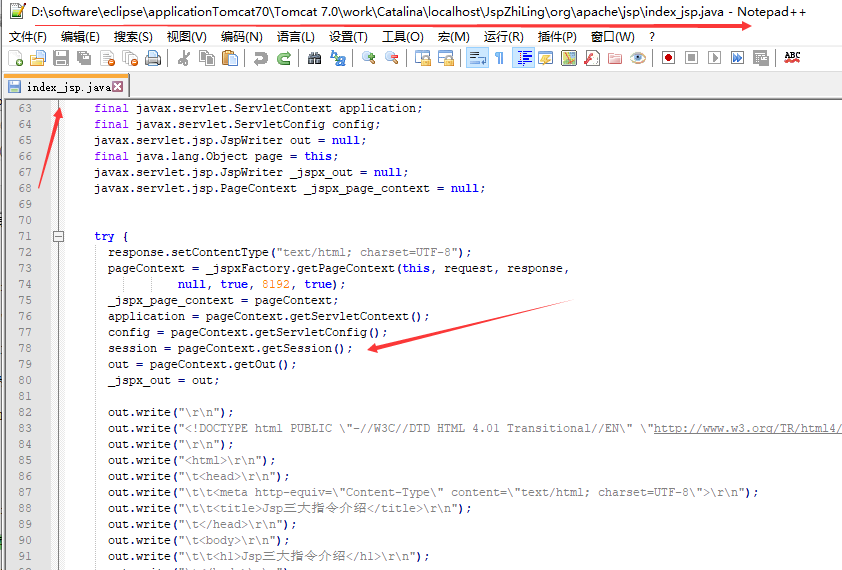
6.page指令的session属性

(1).session的值可以为true或者false， 默认为true。

(2).用于控制在这个jsp页面里面, 能够直接使用session对象。

(3).翻译后的java文件, 如果该值是true, 那么在代码里面会有getSession()的调用; 如果是false, 那么就不会有该方法调用, 也就是没有session对象了。在页面上自然也就不能使用session了。

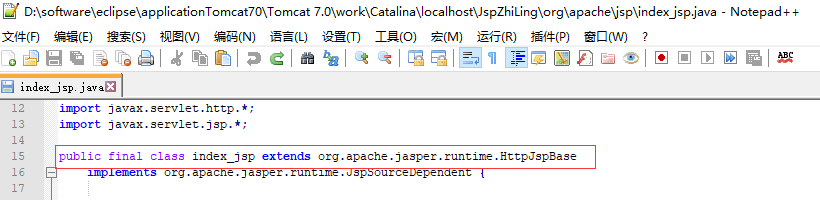
(4).不添加session属性, 部署运行, 打开index\_jsp.java, 发现有getSession(), 说明session属性的默认值是true



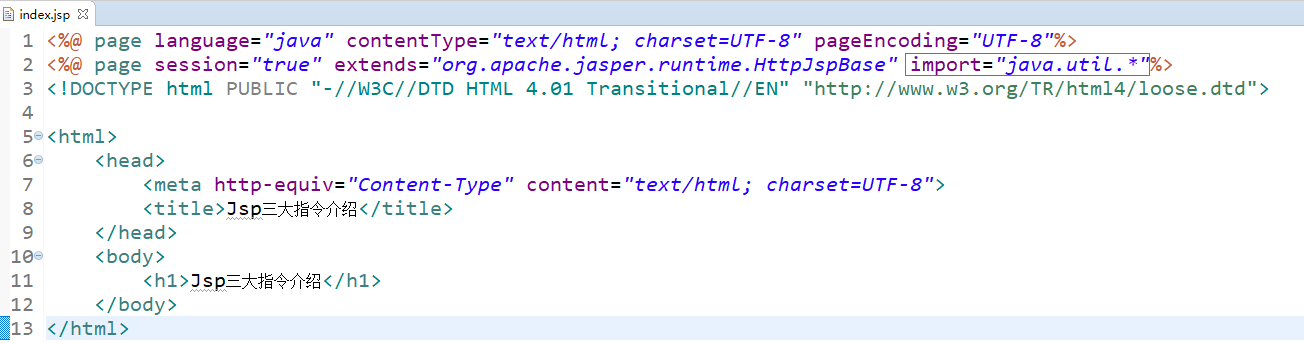
7.page指令的extends属性

(1).extends用于指定jsp翻译成java文件后, 继承的父类是谁, 一般不用改。

(2).打开index.jsp.java, 父类是org.apache.jasper.runtime.HttpJspBase



8.page指令的import属性, 导入java的包

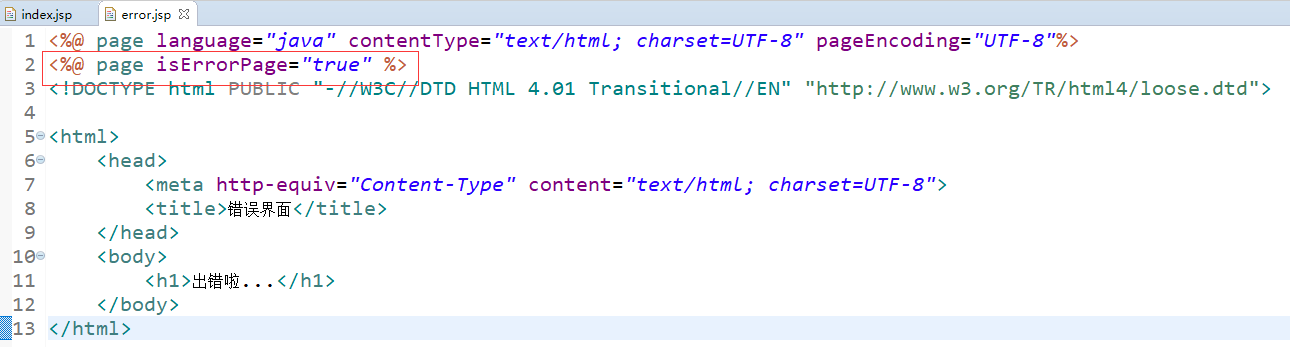


9.page指令的errorPage和isErrorPage属性

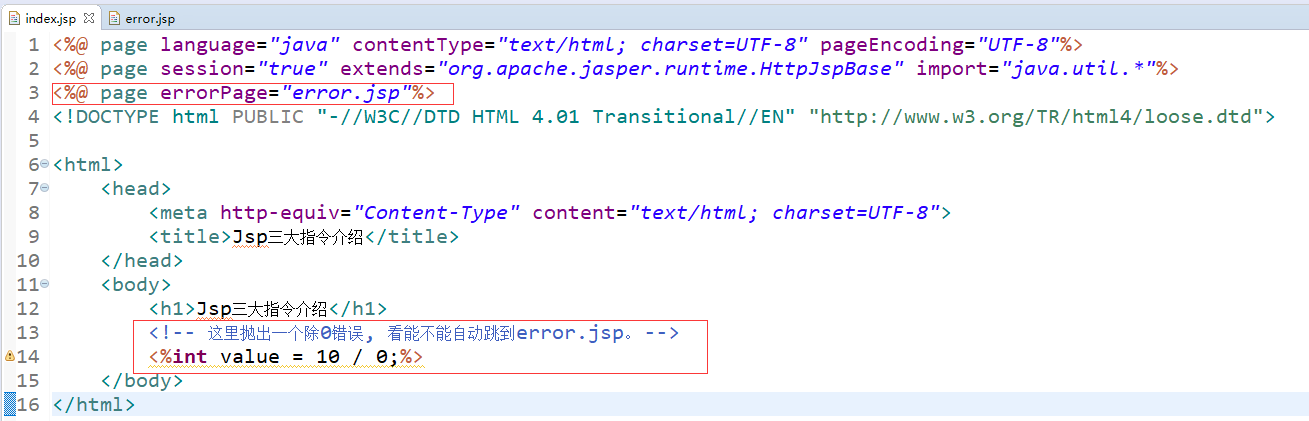
(1).errorPage指的是错误的页面, 我们可以指定一个错误页面的路径。

(2).isErrorPage声明某一个页面到底是不是错误的页面。

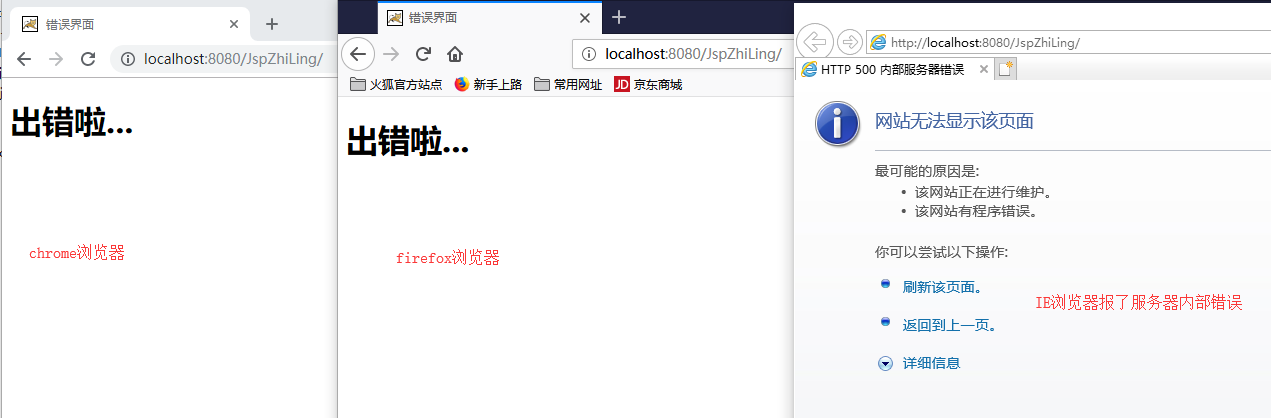
(3).新建一个error.jsp, 声明它是一个错误页面。



(4).在index.jsp里面指明errorPage的路径是error.jsp, 同时页面上抛一个除0的错误



(5).部署运行, 使用chrome和firefox浏览器可以自动跳到error.jsp, 使用IE浏览器却不能跳到error.jsp, 说明此次写法并不能兼容所有浏览器。



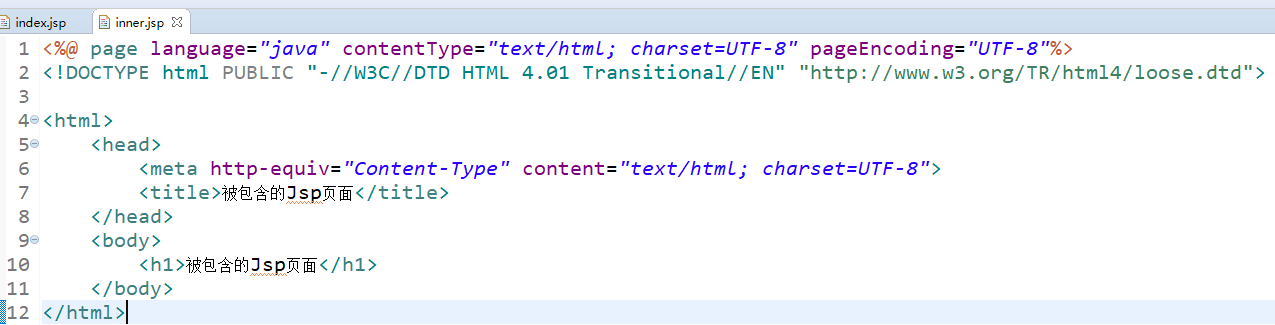
三.include指令

1.包含另外一个jsp的内容进来。

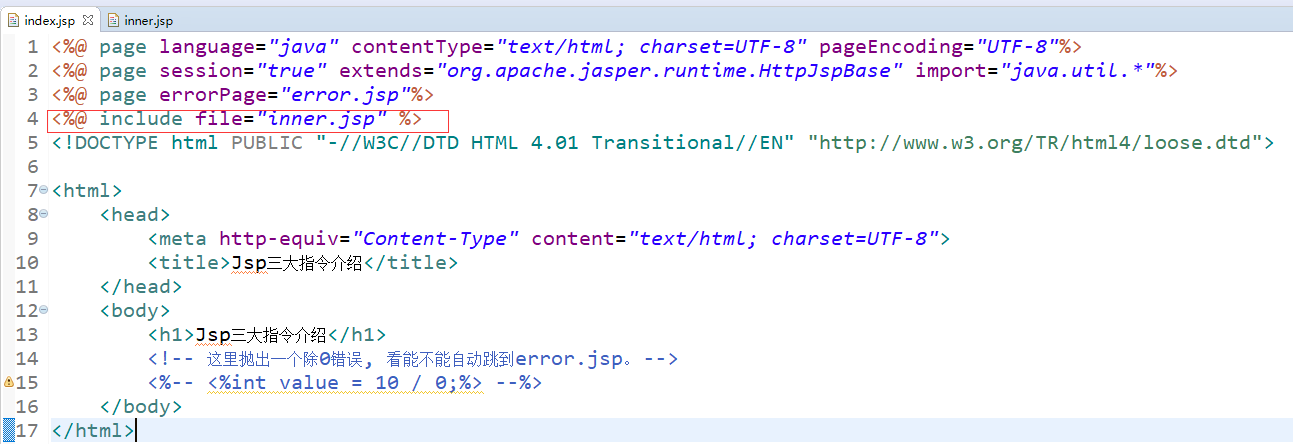
2.写法:<%@ include file="inner.jsp"%>

3.把另外一个页面的所有内容拿过来一起输出。所有的标签元素都包含进来。

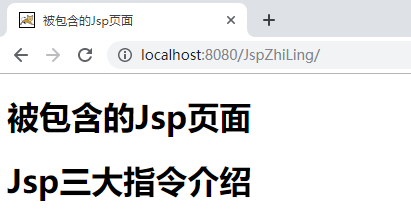
4.新建一个inner.jsp



5.修改index.jsp, 包含inner.jsp



6.部署运行



7.查看网页源码, 包含了2分html



四.taglib指令

1.写法: <%@ taglib prefix="" uri=""%>

2.uri: 标签库路径

3.prefix: 标签库的别名