# JSP

**【学习目标】理解、了解、应用、记忆**

通过今天的学习，参训学员能够：（解释的时候说出二级目标的掌握程度）

1. **【理解】能够阐述jsp的实现原理**
2. **【应用】掌握jsp的page指令**
3. **【应用】掌握jsp的include指令**
4. **【应用】掌握jsp的taglib指令**
5. **【理解】能够阐述 jsp内置/隐式对象（9个）**

# JSP的由来

任何一个技术，诞生用来解决问题：

Java Server Page全称。

场景：

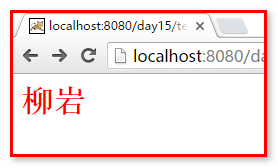
准备页面：

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>  <title>Insert title here</title>  </head>  <body>  <!-- 需求：根据不同请求参数，显示不同古代美女名称 -->  <a href=*"/day15/test?param=1"*>美女1</a>  <a href=*"/day15/test?param=2"*>美女2</a>  <a href=*"/day15/test?param=3"*>美女3</a>  </body>  </html> |

准备servlet：

|  |
| --- |
| **package** cn.igeek.web;  **import** java.io.IOException;  **import** java.io.PrintWriter;  **import** javax.servlet.ServletException;  **import** javax.servlet.http.HttpServlet;  **import** javax.servlet.http.HttpServletRequest;  **import** javax.servlet.http.HttpServletResponse;  //需求：根据不同请求参数，显示不同古代美女名称  /\*\*  \* 问题一：在修改页面的样式的时候，会有大量的字符串操作，比较复杂。  \* 问题二：前端工程师修改页面十分困难。  \* 技术点：jsp技术，也要可以完成Servlet中动态展示网页内容的功能，必然，也要可以书写java代码。  \* JSP是什么：简化书写页面标签，前端工程师也能看的懂这个页面  \* \*/  **public** **class** TestServlet **extends** HttpServlet {  **public** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  //获取请求参数  String param = request.getParameter("param");  response.setContentType("text/html;charset=utf-8");  PrintWriter writer = response.getWriter();  **if**("1".equals(param)){  writer.write("<h1><font color=\"red\">柳岩</font></h1>");  }**else** **if**("2".equals(param)){  writer.write("<h1><font color=\"red\">锁哥</font></h1>");  }**else** **if**("3".equals(param)){  writer.write("老王");  }**else**{  writer.write("参数不合法");  }    }  **public** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  doGet(request, response);  }  } |

效果：



通过上面的操作发现：根据用户请求动态生成网页源代码的时候，我们使用的是response对象，获取输出流输出网页源代码，这种方式有几个问题：

1. 有大量的字符串拼接操作，操作复杂。
2. 前端工程师修改页面代码困难

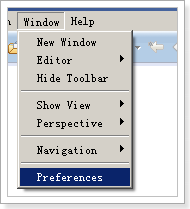
为了解决以上问题sun公司给出了：Java Server Page——简称jsp技术

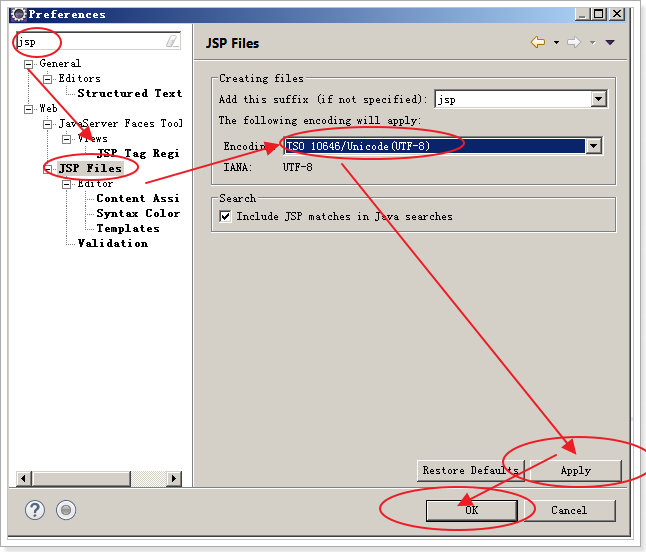
Jsp是为了同时满足动态生成网页和简化页面书写的需求诞生的。

## JSP快速入门（创建jsp文件，简化页面书写）

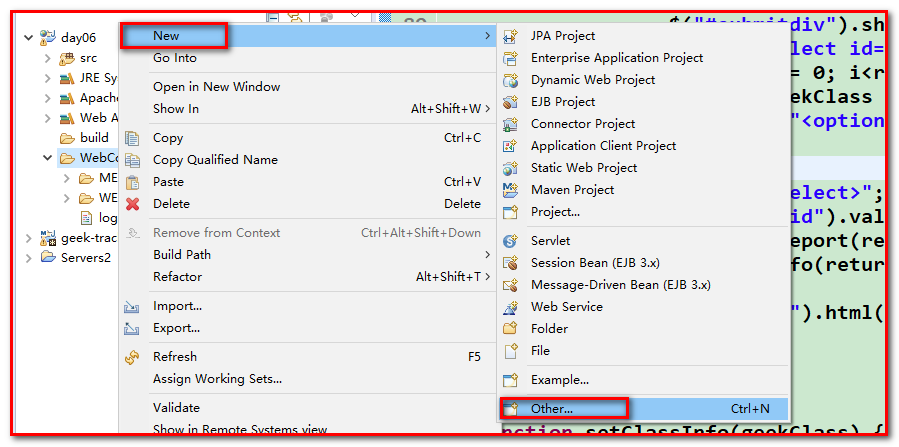
可以直接使用eclipse的模版创建JSP文件：

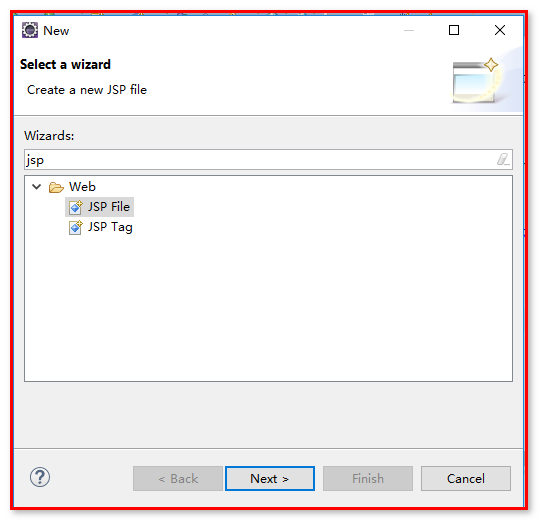
eclipse修改JSP的默认编码表：utf-8

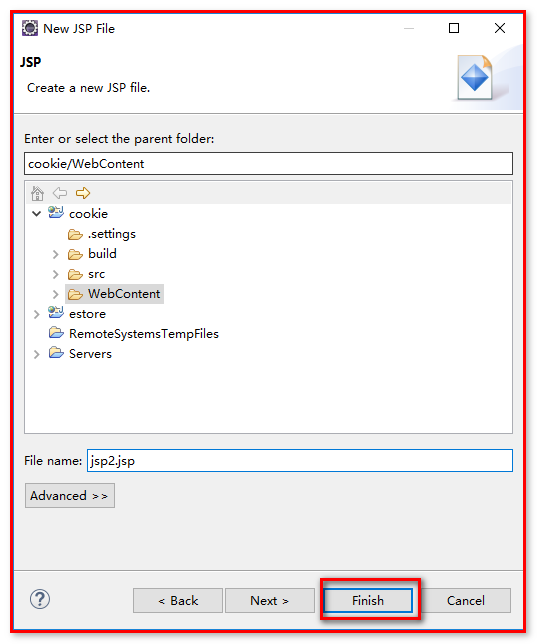




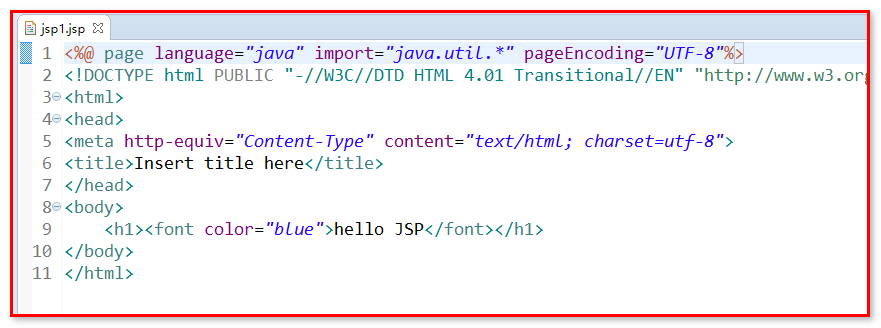
新建jsp文件：





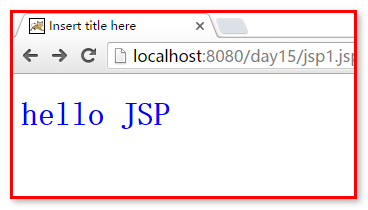


JSP页面代码：

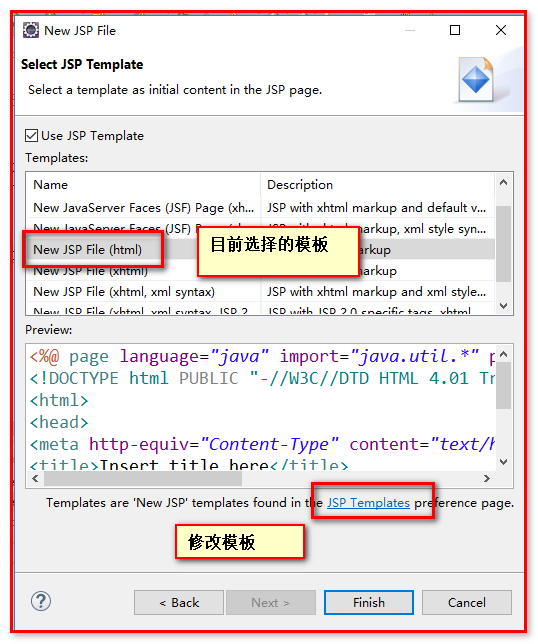


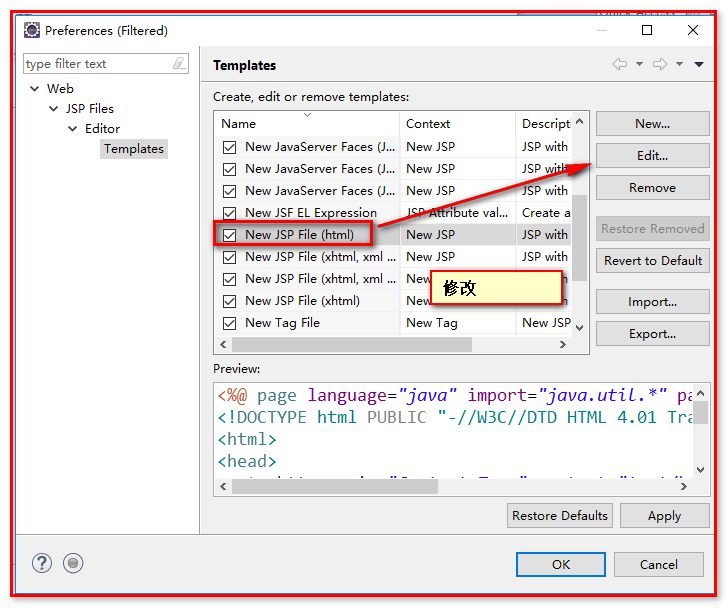
发布项目到tomcat服务器上，打开浏览器访问：看到如下效果，恭喜第一个JSP文件搞定。

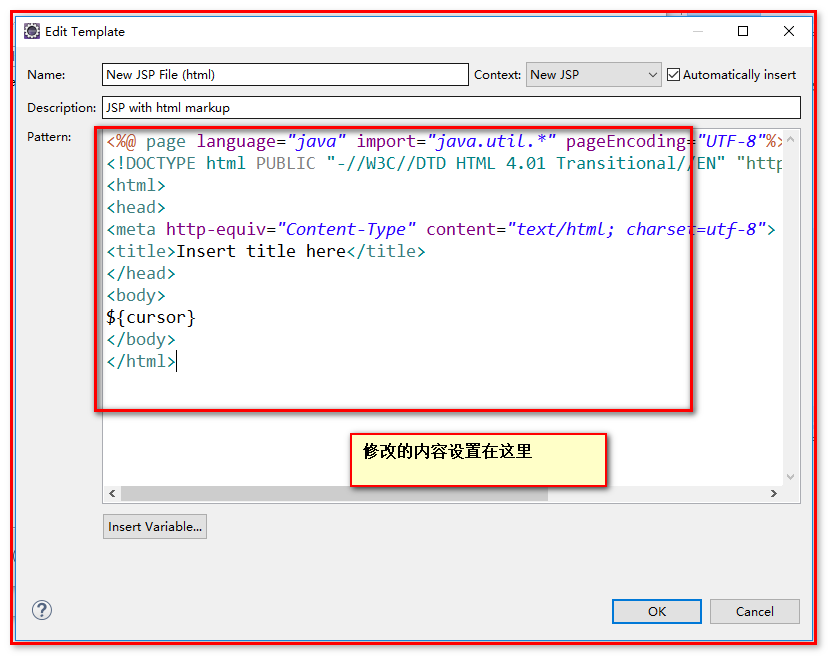
效果：



附：如果需要修改jsp模板：







模板代码：

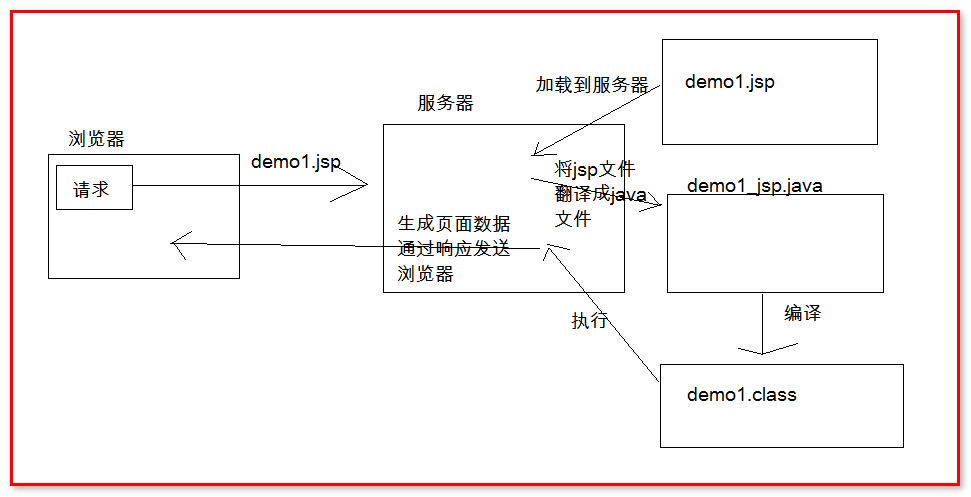
|  |
| --- |
| <%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"UTF-8"*%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"*>  <title>Insert title here</title>  </head>  <body>  ${cursor}  </body>  </html> |

修改完成确认即可

## JSP三种脚本元素（在jsp中使用java代码动态生成网页）

老师，jsp为什么可以执行java代码？

JSP执行原理：



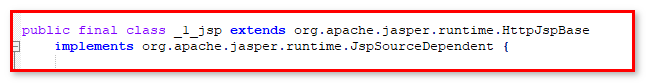
这些文件所在：



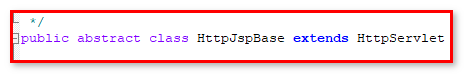
apache-tomcat-7.0.81\work\Catalina\localhost\day04\org\apache\jsp——jsp文件的存放目录

\_1\_jsp部分源码：

它继承HttpJspBase:



HttpJspBase继承HttpServlet:

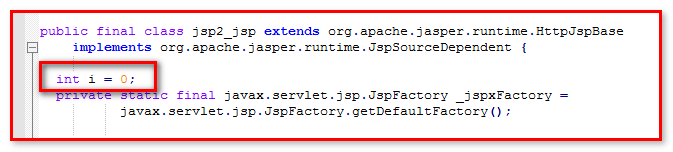


总结：jsp技术最终还是用java类，执行网页内容，jsp说到底还是一个Servlet.

既然是servlet，那么必然可以书写java代码，因此，JSP中提供了书写java代码的三种定义格式：（为了区别页面上的html代码）

## 脚本声明

格式：<%! 书写Java代码 %>

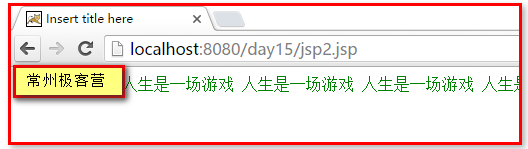


将书写的代码翻译在了类的成员位置。

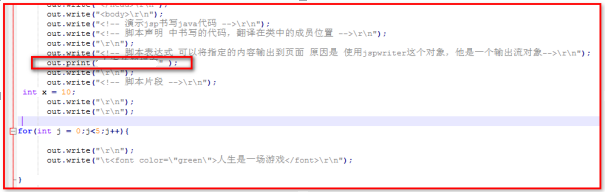
## 脚本表达式

格式：<%= java代码 %>

可以将数据输出到页面：



源代码：代码翻译在\_jspService方法中，而且，被输出流输出

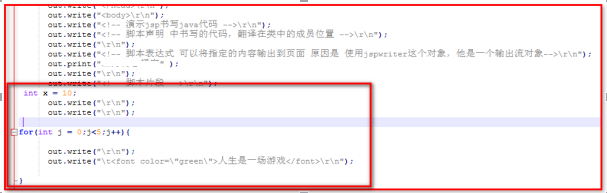


## 脚本片段

格式：<% Java代码片段 %>

源代码：代码翻译在\_jspService方法中

脚本片段，即使分开书写，也能最终组合在一起：



## 完成入门需求：

Html修改：

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>  <title>Insert title here</title>  </head>  <body>  <!-- 需求：根据不同请求参数，显示不同古代美女名称 -->  <a href=*"/cookie/demo?param=1"*>美女1</a>  <a href=*"/cookie/demo?param=2"*>美女2</a>  <a href=*"/cookie/demo?param=3"*>美女3</a>  <hr>  <a href=*"/cookie/jsp3.jsp?param=1"*>美女1</a>  <a href=*"/cookie/jsp3.jsp?param=2"*>美女2</a>  <a href=*"/cookie/jsp3.jsp?param=3"*>美女3</a>  </body>  </html> |

Jsp文件：

|  |
| --- |
| <%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"UTF-8"*%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"*>  <title>Insert title here</title>  </head>  <body>  <%  //为什么request对象直接可以用？    String param = request.getParameter("param");  %>  <%  **if**(param.equals("1")){  %>  <h1><font color=*"red"*>貂蝉</font></h1>  <%  }  %>  <%  **if**(param.equals("2")){  %>  王昭君  <%  }  %>  <%  **if**(param.equals("3")){  %>  西施  <%  }  %>  </body>  </html> |

# JSP技术入门和常用指令

## JSP中注释

JSP中可以写哪些注释？

1. html注释
2. css注释
3. java注释
4. js注释
5. jsp注释

问题：jsp可以写多种注释，什么时候，使用哪种注释？

由于JSP中可以书写html、Java代码，因此在JSP中可以嵌入html注释，还可以嵌入Java注释，还可以书写JSP自己的注释

1. html注释：<!—注释 -->
2. java注释：Java的注释必须嵌入在上面介绍的三个脚本中，不能在jsp其他位置书写。
3. jsp自己的注释：<%-- 注释--%>

注释使用总结：

|  |
| --- |
| <!--  jsp作用：简化书写html标签，动态的生成网页内容（jsp中可以书写java代码）  演示，jsp文件中的各种注释  -->  <%-- jsp文件中注释 ：不会翻译到 .java文件中--%>  <%-- html注释，无法注释掉java代码，但是，在html文件中依然起作用 --%>  <%-- java注释会原封不动的翻译到.java文件中 --%>  <!--11111 <% //int i = 0; %> -->  <%-- 总结：希望在哪个类型的文件中，看到注释，就使用哪种类型的注释 --%> |

## JSP中的page指令（重点：必须掌握）

**什么是指令？**

就是一段代码（代码就一段文本）

**什么样的代码是jsp中的指令？**

例子：

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

第一个记住：格式为：<%@ 指令的名字 key=value key=value …….%>就是jsp指令

Key:属性名称

Value：属性值

Key和value值是用来设置jsp页面（例子：<a href=”http://www.baidu.com”></a>）

**为什么要学习page指令，page指令的功能是什么？**

对jsp页面进行设置的指令（一段代码）。

Page指令：

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

学习page指令（主要是学习指令中的key和value，它们是用来设置jsp的）：

下一个问题：怎么设置jsp页面，设置jsp页面的那些东西？

|  |  |
| --- | --- |
| language="java" | 声明当前jsp使用的编程语言，默认值是java(它现在也只支持java)（工作的时候也是写java） |
| import="java.util.\*" | 导入要使用的包（工作的时候，需要导入类的时候使用）  注意jsp有部分包不用手动导入： |
| pageEncoding="UTF-8" | 设置当前jsp源文件的编码表（工作的时候，就使用UTF-8） |
| contentType="text/html; charset=utf-8" | 设置浏览器解析html的编码表，有pageEncoding的情况可以不设置。  相当于：  Response.setcontentType(“text/html;charset=utf-8”) （工作的时候，不写这个属性）  当设置过pageEncoding="UTF-8"之后，浏览器解析的时候，默认使用UTF-8，所以不再重新设置编码表。 |
| errorPage="500.jsp" | 设置在当前jsp页面(jsp3.jsp)发生异常(int I = 1/0;)后，跳转那个页面（500.jsp）。（工作的时候，如果页面有可能发生错误） |
| isELIgnored="true" | 是否解析jsp中的EL表达式（工作的时候，一般不写，使用默认的，默认为false解析El表达式）  isELIgnored="true"效果： |
| session="true" | 设置在当前的页面中是否可以直接使用session对象（工作时候一般不设置，默认为true）  session="true" 效果： |
| isErrorPage="true" | 设置当前的JSP页面(500.jsp)，是否是显示错误信息页面（500.jsp），如果是错误页面可以看到错误的信息（使用exception对象——jsp中的对象）  演示错误信息显示： |

补充：一般开发的时候，会把整个项目中的常见的错误处理配置到web.xml文件中

测试统一错误配置的时候，需要将jsp page指令中 errorPage属性去掉。

<!-- 配置统一的错误页面 -->

<error-page>

<!-- 服务器的错误响应码 -->

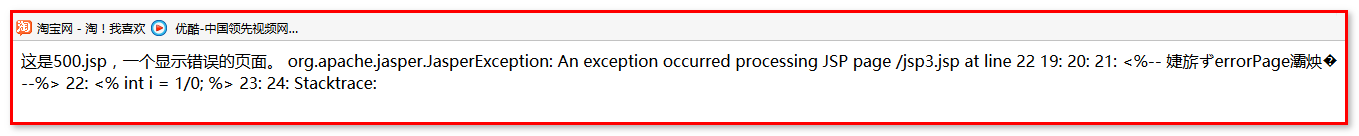
<error-code>500</error-code>

<!-- 跳转那个页面 -->

<location>/500.jsp</location>

</error-page>

效果：



### JSP中的include指令

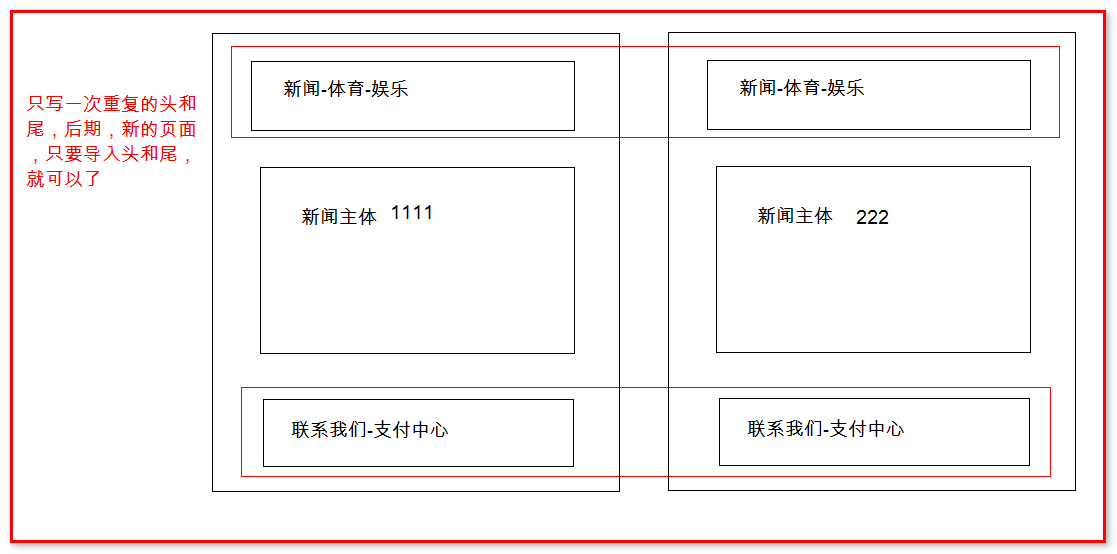
JSP中的include指令作用：引入其他的页面（头页面和尾），合并成一个页面，展示。这种引入方式称为静态引入。

资料演示：





需求分析：



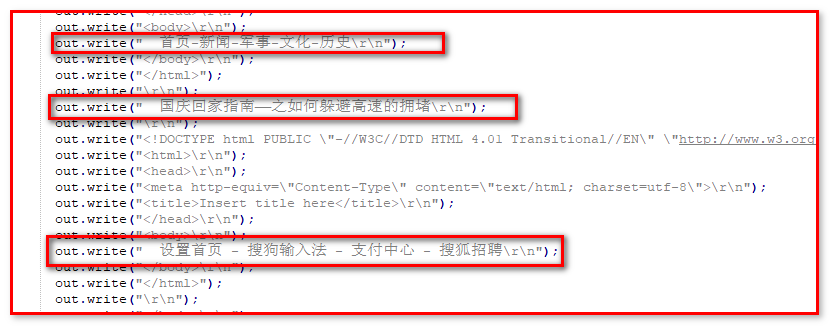
应用场景：

一个网页，一般它的最顶部是一个页面，而最底部是一个页面，主体部分是另外一个页面

|  |
| --- |
| <%@include file=*"header.jsp"* %><br>  这是新闻主体<br>  <%@include file=*"footer.jsp"* %> |

使用这个include指令三个jsp文件最终变成一个class文件，这样引入页面的方式叫做静态引入。

在运行之前，所有的代码已经写在.java源文件中



工作中：需要在当前页面引入其他页面，使用include指令。

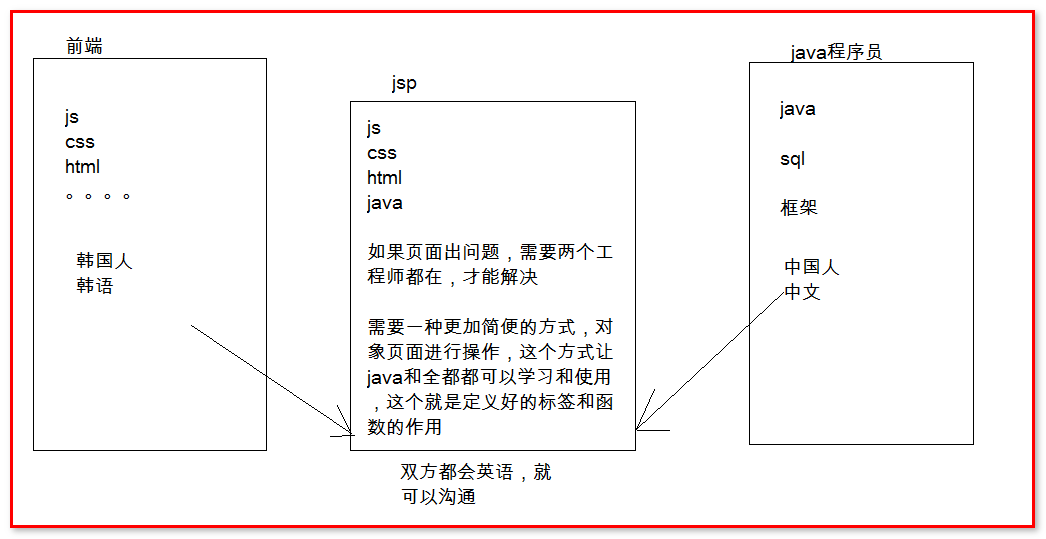
### JSP中的taglib指令（重点：必须掌握）

作用：taglib指令在jsp导入已经定义好的标签库或者函数库（与类库（java对象，一个一个类）不是一个概念），方便程序员使用定义好的标签和函数。

翻译：

将已经定义好的对象（标签和函数），导入到当前的jsp文件中，让我们可以使用。（相当于使用import关键字，可以导入其他已经定义好的java类）

为什么需要标签和函数？ 方便前端和后端开发人员学习，可以使用同一种方式，开发jsp页面



前端崛起；最大的功臣，json

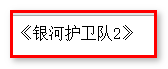
页面由美工维护，美工不懂java代码，需要在jsp页面将java代码剔除，使用标签代替java代码，使用这些标签需要使用taglib指令来引入标签库

工作的时候：需要用到标签库和函数库，都会使用taglib标签，明天就要学习核心标签库的使用。

简单演示：

|  |
| --- |
| <%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"UTF-8"*%>  <%@taglib prefix=*"c"* uri=*"http://java.sun.com/jsp/jstl/core"* %>  <%--  prefix="c" :设定使用标签库或则函数库的短名称，取个小名  uri="http://java.sun.com/jsp/jstl/core" 相当于 ：java.util.\*  设置要导入的标签的全路径，只是这个路径，他长得像地址  --%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"*>  <title>Insert title here</title>  </head>  <body>  <c:out value=*"《银河护卫队2》"*></c:out>  </body>  </html> |

效果：



在taglib指令中的属性：

uri：是当前标签对应的Java代码封装之后的名称空间——指定了一个网址，这个网址是用来确定，我们要引入的是那个标签库或者函数库

prefix：它是当前在页面上可以使用的标签的短名称——小名。

以下为了解：

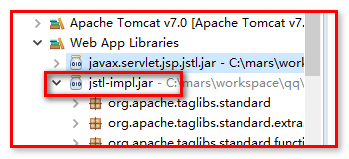
Taglib指令原理：本身jsp可以书写java代码，html、js、java在一个源文件上混杂一起维护不方便，所以将页面的java代码抽取成java类，然后将类配置到文件中，再在jsp页面上导入配置的java类，这样就可以以标签的方式使用定义好的java功能

1 页面的java代码抽取成java类

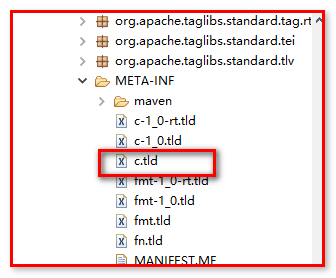
2 将类配置到文件中

3 jsp页面上导入配置的java类

标签库主要Jar包：



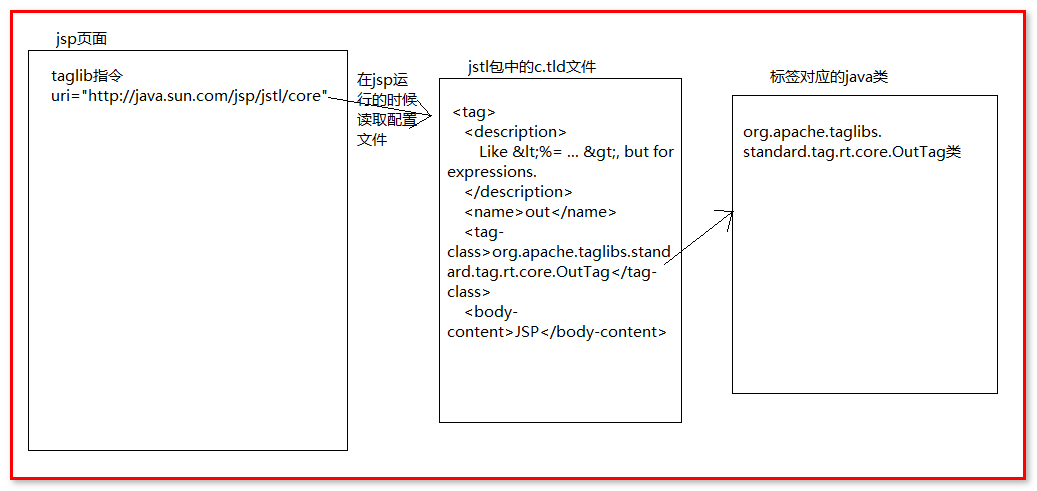
标签库主要配置文件：



配置文件源代码：c.tld



标签执行机制：



# JSP的内置对象&动作标签介绍

什么是内置对象：在jsp中能直接使用的对象就是jsp的内置对象。（request response session exception）

内置对象在哪里？

Jsp底层实际上还是一个java类，可以在jsp中直接使用的，必然存在在jsp翻译后的java类中

JSP的内置对象9个：

这9个内置对象，已经掌握5个，可以在JSP翻译之后的对应的Java源代码的service方法中找到：

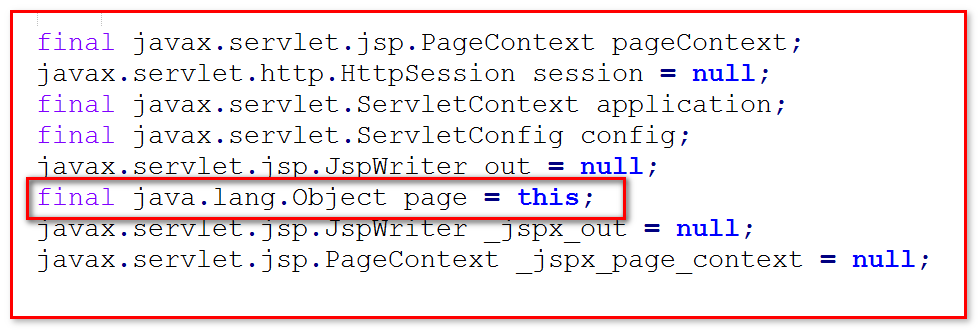
|  |  |
| --- | --- |
| HttpServletRequest request | 请求 |
| HttpServletResponse response | 响应 |
| HttpSession session session | 会话 |
| ServletContext application | 表示当前项目对象 |
| ServletConfig config | 专门获取当前这个Servlet的配置信息 |

新接触的：（要学习的部分）

|  |  |
| --- | --- |
| Object page = this | 它的表示是当前那个JSP页面对象。 |
| PageContext pageContext | 它表示的是当前jsp页面的上下文对象  作用：它的主要功能就是可以获取到JSP页面上的其他八个内置对象。 |
| Throwable exception | 主要是保存JSP页面上的异常信息的对象 |
| JspWriter out | 它相当于我们在Servlet中使用的response.getWriter |

### JSP的内置对象——page对象使用

在JSP页面中，由于page对象被提升成了Object类型。这个page在赋值的时候，赋值的this。



JSP页面中我们经常把page当做javaweb中的第4个容器使用。

Session:生，第一次调用getSession方法 死，有效时间30分钟结束，非正常关闭服务器，invalidate（）；

将什么样的数据存入session中，在一次会话中，需要存活的数据，就存入session中。

例子：用户的昵称，不管用户做什么样的操作（在淘宝上买东西，肯定有多次在服务器上的操作），昵称，始终需要显示在页面的左上方，那么，就应该讲数据存入session中。

Request：生，请求到服务器 死，请求结束，发出响应。

一次请求中需要使用的数据，存入request对象中。

例子：查看商品的时候，商品数据，只需要在请求完成之后，页面显示即可，不需要再下次请求的时候使用，这样的商品数据，就放入request对象。

ServletContext 生：服务器启动 死，服务器关闭

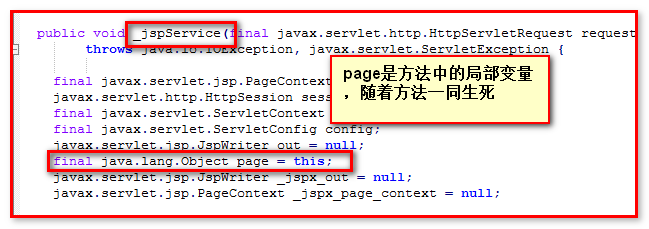
当前应用程序范围，都需要访问的数据，存入ServletContext中。

例子：在线人数。不管是A用户进入网站，还是B用户进入网站，都需要可以访问

注意：如果把page当做容器使用的时候，这时它中保存的内容只能在这个JSP页面中使用，在其他的地方无法使用。

**问题：为什么数据保存在page容器中，只能在当前jsp页面使用？**

Page对象定义在——\_JspService（）方法中，当方法进栈的时候，对象创建可以存取数据，当方法出栈的时候，page对象也随之消失，所以page中存放的数据只能在当前页面使用。



理解：因为page对象定义在\_JSPService方法中，当方法执行完成之后，所有的局部变量都消失，所在page对象中保存的数据，只能在当前的jsp中使用。

Page容器中，应该存什么样的数据？

只在当前页面使用的数据，就应该存在page容器中。

例子：页面使用java代码做循环，比如：i<6，这样的6这个数据，不会再其他页面使用，那么就存入page容器。

总结：web中的四个域对象（容器对象）

Page(当前页面) （当前项目的根路径） < request（一次请求）（商品） < session（一次会话，多次请求）（昵称） < ServletContext(整个项目）在线人数

**为什么request比page范围大？**

Request.getrequestdispather(“/index.jsp”).forward(request,response);获取转发器之后，可以将存入请求对象的数据转发到index.jsp这个页面上。Request中存贮的数据，可以跨越一个servlet和一个jsp,因为page中存贮数据，只能在当前页面使用。

简单总结：一次请求，不止一个页面，一次会话，不止一次请求。

工作的时候：不用page的方法，使用PageContext对象来操作page。

### JSP内置对象——pageContext使用（重点：必须掌握）

作用：1 获取其他八个内置对象（9个） 2 向四个web容器（page request session application ）设置数据 3 向四个web容器（page request session application ）获取数据

学习一个类：

1. 通过API文档找到相应的类
2. 第一个要看的是构造方法（可以自己获取对象——注意：有些对象是服务器提供不能自己new）
3. pageContext对象的创建不需要程序员自己动手，源代码中已经创建好了



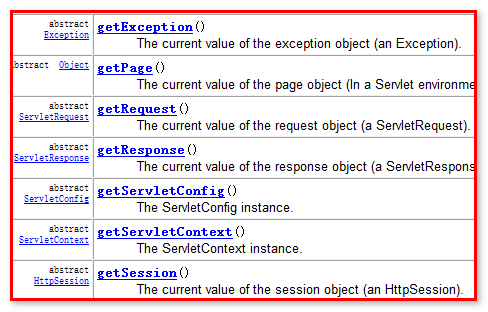
人话：提供了获取其他八个内置对象的方法，可以对四个web容器进行存取数据操作。

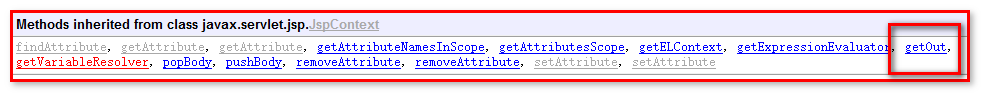
创建PageContext对象：



pageConext它是一个工具类，有三个功能：

1、获取jsp页面中的其他的8个内置对象





代码演示：

<%-- 获取其他八个内置对象 --%>

<%= pageContext.getException() %><br/>

<%= pageContext.getOut() %><br/>

<%= pageContext.getPage() %><br/>

<%= pageContext.getRequest() %><br/>

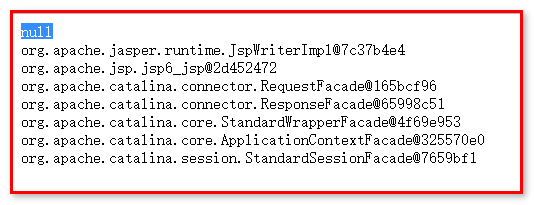
<%= pageContext.getResponse() %><br/>

<%= pageContext.getServletConfig() %><br/>

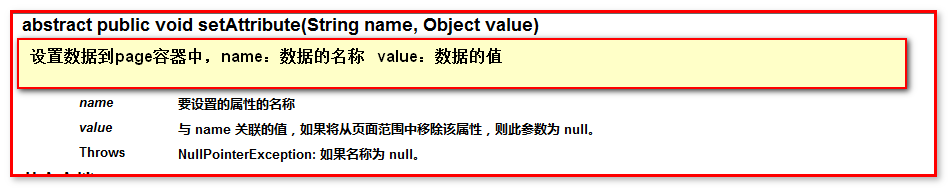
<%= pageContext.getServletContext() %><br/>

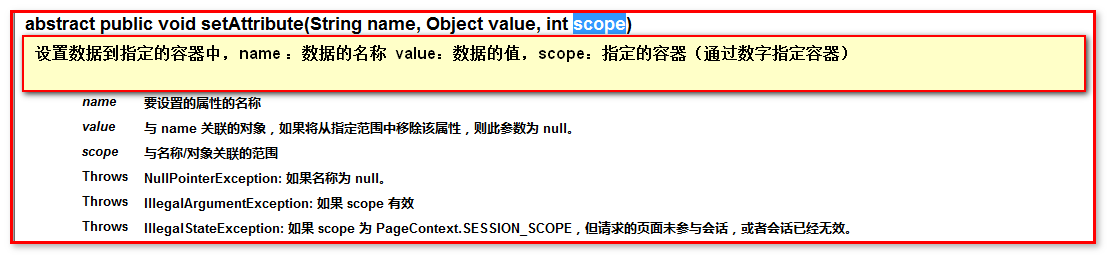
<%= pageContext.getSession() %><br/>

效果：

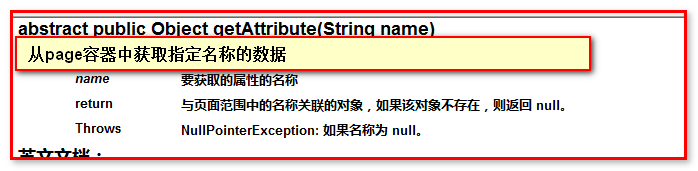


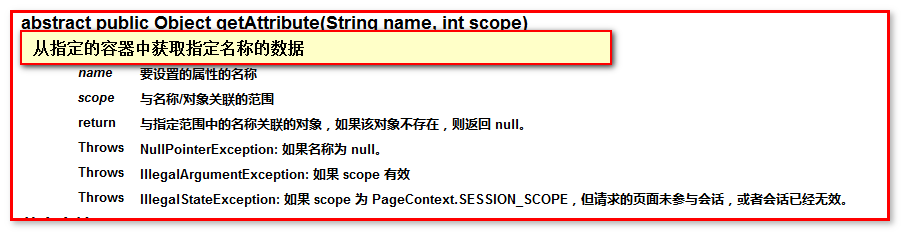
2、给其他4个域对象==（容器）中的设置数据





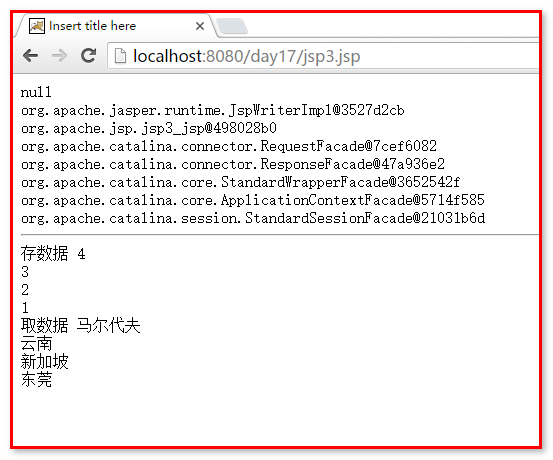
3、从4个容器中取出数据





|  |
| --- |
| <%-- 演示pageContext获取其他8个内置对象 --%>  <%= pageContext.getException() %><br>  <%= pageContext.getOut() %><br>  <%= pageContext.getPage() %><br>  <%= pageContext.getRequest() %><br>  <%= pageContext.getResponse() %><br>  <%= pageContext.getServletConfig() %><br>  <%= pageContext.getServletContext() %><br>  <%= pageContext.getSession()%><br>  <hr>  <%-- 演示设置数据和取出数据 --%>  存数据  <%  pageContext.setAttribute("addr", "马尔代夫", pageContext.APPLICATION\_SCOPE);  pageContext.setAttribute("addr", "云南", pageContext.SESSION\_SCOPE);  pageContext.setAttribute("addr", "新加坡", pageContext.REQUEST\_SCOPE);  pageContext.setAttribute("addr", "东莞", pageContext.PAGE\_SCOPE);  %>  <%=pageContext.APPLICATION\_SCOPE %><br>  <%=pageContext.SESSION\_SCOPE %><br>  <%=pageContext.REQUEST\_SCOPE %><br>  <%=pageContext.PAGE\_SCOPE %><br>  取数据  <%=pageContext.getAttribute("addr", pageContext.APPLICATION\_SCOPE) %><br>  <%=pageContext.getAttribute("addr", pageContext.SESSION\_SCOPE) %><br>  <%=pageContext.getAttribute("addr", pageContext.REQUEST\_SCOPE) %><br>  <%=pageContext.getAttribute("addr", pageContext.PAGE\_SCOPE) %><br> |

效果：



工作的时候，需要获取其他八个内置对象的时候和操作page容器存取数据的时候，使用pageContext。

例如：获取项目根路径。

pageContext获取request对象，getContextPath（）——获取项目根路径

### JSP内置对象——out的使用

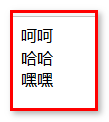
作用：out它主要是用来把数据输出到页面（相当于response.getWriter（））。

<% out.write("哈哈"); %><br>

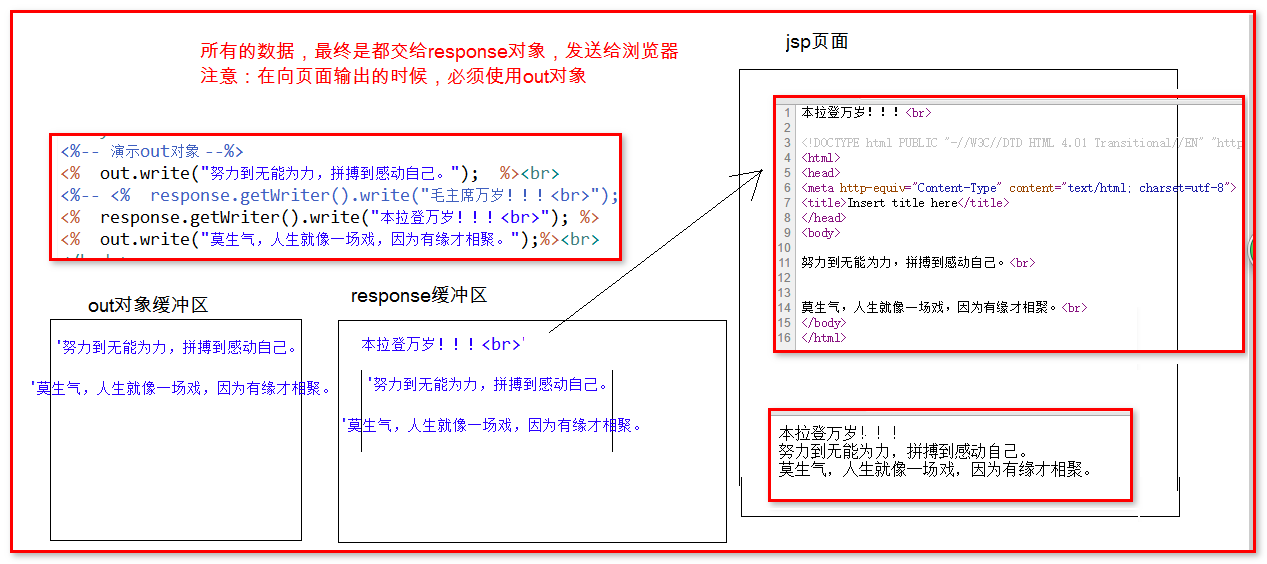
<% response.getWriter().write("呵呵<br>"); %>

<% out.write("嘿嘿"); %><br>

效果：



原理：



数据最终是通过response对象输出到页面中，所以out缓冲区中的数据最终还是到response中，通过response对象输出。

### JSP内置对象——exception对象

内置对象exception，在jsp页面上要能够使用exception内置对象，在jsp的page指令中，必须书写

isErrorPage=true，否则在页面上是无法使用exception内置对象。



谨记：不要把异常信息显示到JSP页面(不能让用户看到)。一般异常需要在后台的Java代码处理完成， 如果真的有异常，就使用友好页面处理。





### JSP的动作标签

JSP页面中是可以嵌入Java代码完成我们的功能，但是后期开发中JSP页面中是不写任何Java代码的（页面上本来，就应该没有java代码，页面本来就是前端工程师开发）。

而在JSP中给我们提供了一些内置的标签，可以代替Java代码。

### jsp:include标签 动态引入

作用：引入其他页面，合并成一个页面，展示。（动态引入）

<%--

JSP中的内置标签：

jsp:include 它和 jsp的include指令功能相同

jsp:include内置的标签，它是动态的包含，当使用它引入多个JSP页面的时候，

在翻译引入的所有JSP为Java文件的时候，会翻译成不同的Java文件，只有在

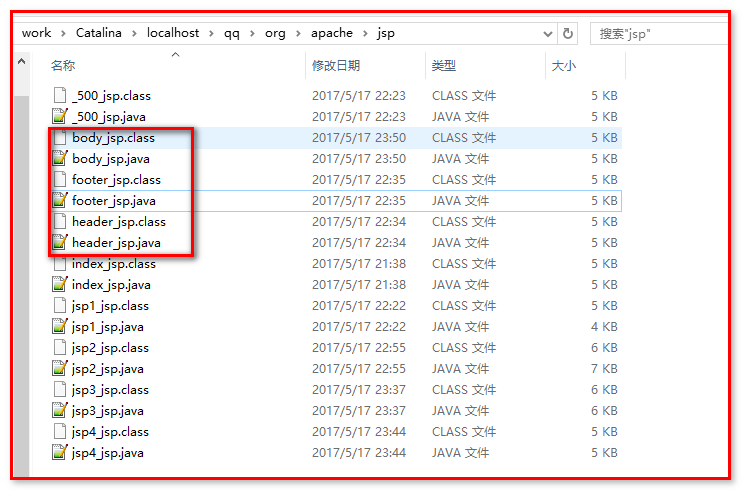
程序运行的时候，才会把需要包含的内容引入进来。

jsp的include指令 它是静态的包含 在翻译的时候就把所有的JSP合并成一个Java程序

--%>

|  |
| --- |
| <%-- <%@include file="header.jsp" %> --%>  <jsp:include page=*"header.jsp"*/><br>  这是新闻主体<br>  <%-- <%@include file="footer.jsp" %> --%>  <jsp:include page=*"footer.jsp"*/><br> |

执行后的效果：



分析原理：



什么时候用静态引入(include指令)，什么时候使用动态引入（include jsp 动作标签）？

随便用，没关系。

### jsp:forward标签

它主要是起到我们在Servlet中学习的request的请求转发的作用。

<%-- 演示jsp froward标签 --%>

<jsp:forward page=*"index.jsp"*></jsp:forward>

