# 1 JSP

Java Server Pages，SUN公司提供的一种动态网页编程技术。

为什么要有JSP ？

1 Servlet输出页面很繁琐

2 Servlet生成的页面没有预览工具

JSP： JSP = HTML+java代码+JSP本身的标签

JSP是动态资源还是静态资源？

是动态资源 放的位置、编写和访问上很像静态资源。

**JSP本质上就是Servlet**

jsp如何运行？

一个jsp在第一次访问时，会被容器（服务器）中的jsp引擎（容器中编译jsp的软件模块）

编译成一个Servlet jsp名字\_jsp.java jsp名字\_jsp.class并且放在 Tomcat的work目录下。

JSP和Servlet表现形式上不同

a.编写jsp文件就是编写文本文件一样。编写servlet编写的是java程序。

b.编写jsp以后，直接将jsp拷贝到容器的部署目录就可以了。

c.编写后的servlet要通过web.xml文件加以配置。jsp直接访问就行了。

d.jsp和静态资源放在一起就可以了。-----jsp在使用上更像静态资源。

**jsp中嵌入java代码：**

jsp中三种脚本标签：

<%! %> 这个标签可以用来声明变量，方法 ，类

变量和方法是 成员变量和成员方法

类是成员内部类

一般不用该标签

<% %> 在这个标签中编写的java代码

将会出现在 \_jspService()

<%= %> 在这个标签中写的java代码将会出现在

out() 中 向页面输出 该标签中的代码

一定不要加分号

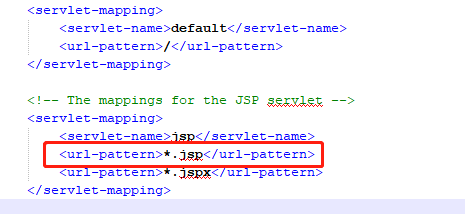
**JSP中的注释：**

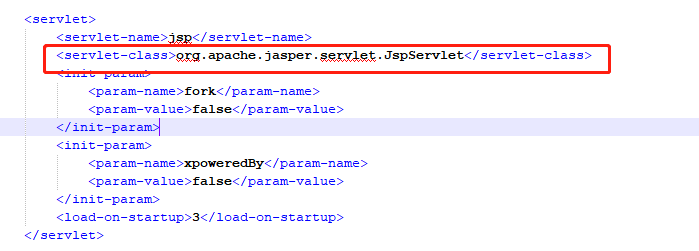
<!-- -->

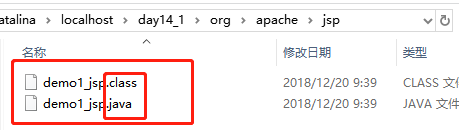
<%-- --%>

## JSP的执行原理

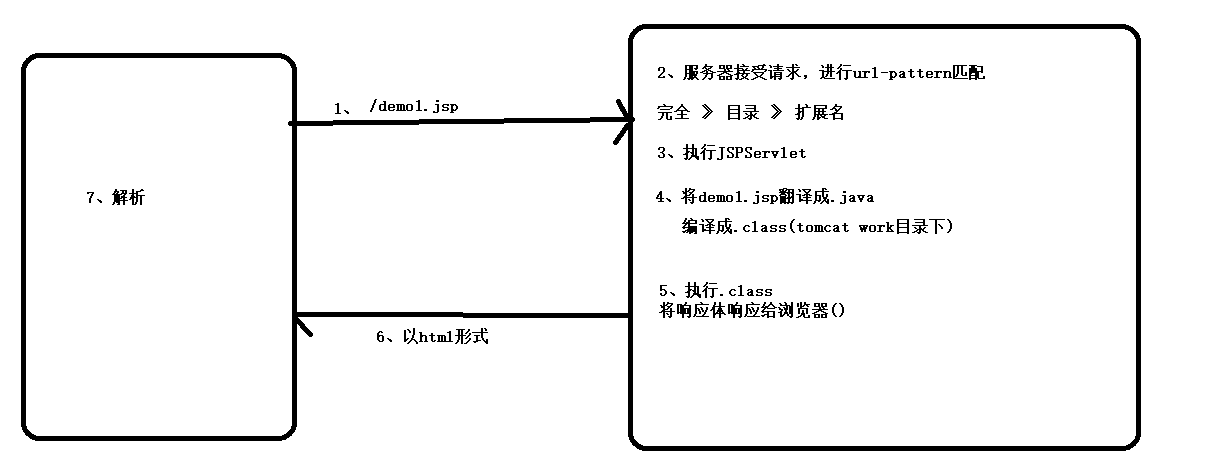
tomcat的conf下web.xml







翻译后的jsp类继承HttpJspBase，是HttpServlet的子类 jsp本质就是Servlet



# 2 JSP指令标签

jsp中的指令标签是用来控制整个jsp页面行为的

1 **<%@ page %>**

language jsp中可以嵌入的语言 目前为止只能填java

import 用来导包

pageEncoding 设置当前jsp页面的编码方式

contentType 设置浏览器查看这个页面时使用编码方式

session 默认是true 当前jsp中能否使用session(如果为true，则在这个jsp中就已经可以使用session对象)

isELIgnored 当前页面能否使用EL表达式

buffer 缓存区大小 默认是8kb

autoFlush 是否自动刷新

errorPage 指定错误页面

isErrorPage 设置为错误页面 (设置为错误页面，就有处理异常的功能)

以上两个属性，一般不用

一般错误处理是在web.xml中统一进行配置：

|  |
| --- |
| <error-page>  <error-code>404</error-code>  <location>/nofound.jsp</location>  </error-page>  <error-page>  <error-code>500</error-code>  <location>/error.jsp</location>  </error-page> |

|  |
| --- |
| <error-page>  <exception-type>java.lang.NullPointerException</exception-type>  <location>/error.jsp</location>  </error-page> |

**2 <%@ include file="" %>**  在当前页面包含其他页面

静态包含: 静态包含也叫做编译期包含，两个jsp将会被编译成一个Servlet文件

静态包含时，生成的html的结构被破坏

**3 <%@ taglib prefix="前缀" uri="标签库的路径" %>** 引入标签库

# 3 JSP内置对象（9个）

page Object jsp编译后生成的servlet的对象(一般不用)

pageContext PageContext

request HttpServletRequest

response HttpServletResponse

session HttpSession

application ServletContext

out JspWriter

config ServletConfig 获取初始化参数

exception Throwable（当jsp的isErrorPage="true"）

查看JSP生成的.java文件，即可看到这些对象。

## 3.1 out对象

是 JspWriter类型 字符流

**out和 response.getWriter() 不是同一个流**

out流 和 response.getWriter() 都是字符流 都有缓冲区

out流的执行是 先把 缓冲区中的内容 输出到 response.getWriter()的缓冲区中，再由response.getWriter() 这个流统一输出。

|  |
| --- |
| <%="aaa" %>  <%  out.write("bbb");  %>  <%  response.getWriter().write("ccc");  %> |

执行顺序为

ccc

aaa

bbb

## 3.2 pageContext对象

**pageContext(父类是JspContext):JSP的上下文对象 可以用来获得其他八个内置对象**

pageContext.getRequest();

pageContext.getResponse();

pageContext.getSession();

pageContext.getServletConfig();

pageContext.getServletContext();

pageContext.getOut();

pageContext.getException();

pageContext.getPage();

这些对象可以直接在JSP中获得，那么pageContext提供这些方法有什么用？

一般用于框架的编写，自定义标签的编写等。

**pageContext也是一个作用域 范围就是当前页面**

setAttribute("",Object);

getAttribute("")

removeAttribute

Object findAttribute(String name)

按照 pageContext request session application的顺序根据 name 查找作用域

setAttribute(String name, Object value, int scope)

|  |
| --- |
| pageContext.setAttribute("pageKey", "pageValue",PageContext.PAGE\_SCOPE);  // pageContext.setAttribute("pageKey", "pageValue")  pageContext.setAttribute("requestKey","requestValue",PageContext.REQUEST\_SCOPE);  //request.setAttribute("requestKey","requestValue");  pageContext.setAttribute("sessionKey","sessionValue",PageContext.SESSION\_SCOPE);  //sessionsetAttribute("sessionKey","sessionValue");  pageContext.setAttribute("applicationKey","applicationValue",PageContext.APPLICATION\_SCOPE);  //application.setAttribute("applicationKey","appicationValue"); |

## 3.3 request对象

\_jspService方法的参数 HttpServletRequest类型。用法与Servlet中的request

用法相同

## 3.4 response对象

\_jspService方法的参数 HttpServletResponse类型。用法与Servlet中的response用法相同

## 3.5 session对象

HttpSession类型

## 3.6 application对象

ServletContext类型，用法和Servlet中的ServletContext用法相同

## 3.7 config对象

ServletConfig类型，用法和Servlet中的ServletConfig用法相同

## 3.8 page对象

Object类型，Object page = this;

## 3.9 exception对象

Throwable类型，只有当jsp的isErrorPage="true"时，才会出现该对象。

# 4 EL表达式

EL是Expression Language的缩写，它是一种简单的数据访问语言。本节将针对EL表达式进行详细的讲解。

EL（Expression Language） 目的：为了使JSP写起来更加简单。表达式语言的灵感来自于 ECMAScript 和 XPath 表达式语言，它提供了在 JSP 中简化表达式的方法，让Jsp的代码更加简化

**EL表达式的作用**

1获取作用域中的值

2做运算 （数学运算 逻辑运算 关系运算...）

3 EL 提供常用的操作web对象 （11 个）

4 访问java的方法

## 4.1 获取作用域中的值

**例如：${name}**

按照 page，request，session，application(ServletContext)的顺序,

查找作用域 找 key是name的 value值 。

也可以指定查找的范围

${pageScope.key}

${requestScope.key}

${sessionScope.key}

${applicationScope.key}

getAttribute(key); 当key不存在时返回 null

${key} 当key不存在时返回空字符串""

**User user = new User(1,"zhangsan",20);**

**request.setAttribute("user", user);**

${user.id}

${user.name}

${user.age}

user----> 保存在request中时的key值

id，name ，age 对象的属性名

**要求： 属性必须有get方法 EL表达式才能访问到**

**EL表达式也可以用来访问集合结构**

|  |
| --- |
| <%  List<String> list = new ArrayList<String>();  pageContext.setAttribute("list", list);  %>  ${empty list } 判断list中是否有元素    数组 List Map    <%  String[] ss = {"aaa","bbb","ccc"};  pageContext.setAttribute("ss", ss);  List<String> list = new ArrayList<String>();  list.add("xxx");  list.add("yyy");  list.add("zzz");  request.setAttribute("list",list);  List<User> list2 = new ArrayList<User>();  User u1 = new User(1,"aaa",20);  User u2 = new User(2,"bbb",20);  User u3 = new User(3,"ccc",20);  list2.add(u1);  list2.add(u2);  list2.add(u3);  request.setAttribute("list2",list2);      Map<String,String> map = new HashMap<String,String>();  map.put("aaa-a", "111");  map.put("bbb", "222");  map.put("ccc", "333");    pageContext.setAttribute("map", map);    %>  ${pageScope.ss[1] }<br>  ${requestScope.list[2] }<br>  ${list2[0].name}<br>  ${map.bbb } <br>  ${map["aaa-a"] } |

**EL不能遍历集合结构**

## 4.2 做运算

${10+2 } ${10 / 2} ${5==5 } ${5<3 }

${list2[0].name=="aaa" }

${list2[0].age==20 }

${list2[0].age == list2[1].age }

|  |  |  |  |
| --- | --- | --- | --- |
| **操作** | **描述** | **示例** | **结果** |
| **==（eq）** | 是否相等 | ${5==5} | True |
| **!=（ne）** | 是否不等 | ${5!=5} | false |
| **<（lt）** | 是否小于 | ${5<7} | True |
| **>（gt）** | 是否大于 | ${5>7} | false |
| **<=（le）** | 是否小于等于 | ${5 le 5} | true |
| **>=（ge）** | 是否大于等于 | ${5 ge 6} | false |

|  |
| --- |
| **${user.weight gt user.IQ}（绝对不能写成：${user.weight} gt ${user.IQ}或者${${user.weight} gt ${user.IQ}}** |

|  |
| --- |
| EL表达式中使用逻辑运算符：逻辑运算符有and or not  例如：  ${1==1 and 2==2}  ${param.month==5 and param.day==25}  为了增强优先级使用括号：  ${(param.month==5 or param.month==6) and param.day==25} |

## 4.3 EL 提供常用的操作web对象（11个）

|  |
| --- |
| pageScope  requestScope  sessionScope  applicationScope  param 相当于 request.getParameter()  paramValues 相当于 request.getParameterValues()  header 相当于 request.getHeader(); 一个key对应一个value  headerValues 相当于 request.getHeaders() 一个key对应多个value  initParam 获取初始化参数（全局初始化参数）  cookie 获取Cookie的  pageContext 获取其他八个内置对象 访问这个八个对象下的get...方法 |

|  |
| --- |
| ${header[“User-Agent”]} |

|  |
| --- |
| Cookie c = new Cookie("aaa","111");  response.addCookie(c);  Cookie c2 = new Cookie("bbb","222");  response.addCookie(c2);  response.sendRedirect("/1204el/el2.jsp");  ${cookie.aaa.value } ----111  ${cookie.bbb.value } ----222  ${cookie.ccc.value } ----“”  ${cookie.aaa.name } -----aaa |

|  |
| --- |
| ${pageContext.request.contextPath }  ${pageContext.session.id }  ${pageContext.request.method}  ${pageContext.session.creationTime } |

# 5 JSTL标签库

## 5.1 JSTL介绍

Jsp Standard Tag Library （jsp标准标签库）

主要作用用来替换传统页面中<%%>

**版本：** 1.0

1.1

1.2

JSTL1.0 J2EE1.4年代以前的版本，EL表达式没有纳入官方的规范.(JSTL中不能使用EL表达式)

JSTL1.1 JSTL1.2 J2EE1.4版本以后 EL都已经纳入官方规范. 1.1 和 1.2两个版本没有本质区别.

**Jar包：**

standard.jar

jstl.jar

## 5.2 JSTL标签库分类

c(core) 核心标签库

fmt 格式化（国际化）

fn 函数 （JSTL提供的EL函数）

xml

Sql

## 5.3 页面中引入JSTL标签库

如何使用JSTL标签库：

在jsp页面中导入：

<%@taglib prefix="" uri="" %>

**<%@ taglib uri=”http://java.sun.com/jsp/jstl/core” prefix=”c”%>**

## 5.4 c(core)核心标签库

|  |
| --- |
| if 没有 c:else  test = "${user.name=='aaa'}"  如果test中的表达是为true 则 if中夹含的代码会执行  var test运算后的值（true/false）  scope var中定义的变量存放在哪个作用域 |

|  |
| --- |
| choose 类似于 switch  when test  otherwise |

|  |
| --- |
| forEach:  items 要被遍历的集合结构  var 每取出集合中的一个对象 赋给 var变量  begin  end  step  varStatus 用来产生序号  varStatus="abc" abc.index 从零开始  abc.count 从1开始  abc.first true/false  abc.last true/false  <c:forEach items="${list }" var="u">  ${u.id }--${u.name }--${u.sex }<br>  </c:forEach> |

## 5.5 fmt 格式化标签库

**<%@ taglib uri=”http://java.sun.com/jsp/jstl/fmt” prefix=”fmt”%>**

<fmt:formatDate>标签：格式化时间和日期

|  |
| --- |
| 例如：  <fmt:formatDate value="${current}"/>  通过type格式指定具体的格式形式。  <fmt:formatDate value="${d}"type="date"/>只格式成日期  <fmt:formatDate value="${d}"type="time"/>只格式成时间  <fmt:formatDate value="${d}"type="both"/>格式成日期+时间  通过pattern属性指定具体的格式要求  <fmt:formatDate value="${d}"pattern="yyyy-MM-dd"/> |