# JSP

## JSP概述

* JSP全称是Java Server Pages，它和servle技术一样，都是SUN公司定义的一种用于开发动态web资源的技术。JSP/Servlet规范。JSP实际上就是Servlet。
* JSP这门技术的最大的特点在于，写jsp就像在写html，但它相比html而言，html只能为用户提供静态数据，而Jsp技术允许在页面中嵌套java代码，为用户提供动态数据。
* Jsp快速入门：在jsp页面中输出当前时间

|  |
| --- |
|  |
| **注：使用JSP页面，需要导入tomcat中jsp.jar和servlet-api.jar两个包** |

## JSP原理

* 当访问lesson1.jsp时，会先将其编译lession1\_jsp.java，然后再编译成lession1\_jsp.class

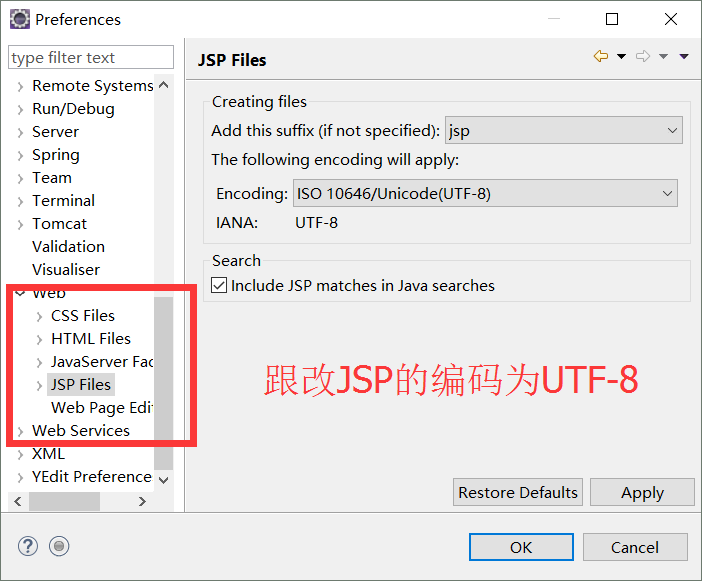
|  |
| --- |
|  |
|  |

* **JSP本质就是一个Servlet**

**注：**

**在Eclipse中tomcat运行项目时的Servlet上下文路径与手动运行tomcat的Servlet上下文件路径的不同**

## 修改Eclipse创建JSP的编码



## jsp的最佳实践

* Servlet：控制器。重点编写java代码逻辑 （获取表单数据、处理业务逻辑、分发转向）
* JSP：代码显示模板。重点在于显示数据

演示：JSP也可以处理表单请求，但一般不用这么用，由Servlet去处理表单

|  |
| --- |
|  |
|  |

**注：request,out,response称为JSP的内置对象，不需要声明直接使用**

# JSP的基本使用

## 1、JSP模版元素

网页的静态内容。如：html标签和文本。

## 2、JSP的脚本

2.1、小脚本 <% java代码 %>

2.1、表达式 <%= 2+3 %> 等价于out.print(2+3);

2.3、声明 <%! %> 表示在类中定义全局成员，和静态块。

## 3、JSP注释

JSP注释：<%-- 被注释的内容 --%> 特点：安全，省流量

网页注释：<!-- 网页注释 --> 特点：不安全，费流量

|  |
| --- |
|  |

# JSP的3个指令

JSP指令（directive）是为JSP引擎而设计的，它们并不直接产生任何可见输出，而只是告诉引擎如何处理JSP页面中的其余部分。

在JSP 2.0规范中共定义了三个指令：【page、Include、taglib】

|  |
| --- |
| 语法： |
| <%@ 指令名称 属性1=“属性值1” 属性2=“属性值2”。。。%>  或者：  <%@ 指令名称 属性1=“属性值1”%>  <%@ 指令名称 属性2=“属性值2”%> |
| 如：JSP的默认第一行 |
|  |

## Page

属性：

|  |
| --- |
| **少用：自学** |
| session: 是否会自动创建session对象。默认值是true; |
| buffer: JSP中有javax.servlet.jsp.JspWriter输出字符流。设置。输出数据的缓存大小，默认是8kb |
| isErrorPage: 是否创建throwable对象。默认是false; |
| **常用：演示** |
| import 和java代码中的import是一样的  <%@ page import="java.util.Date,java.util.List"%>  或者：  <%@ page import="java.util.Date"%>  <%@ page import="java.util.List"%>  **注：JSP会自动导入以下的包：**  **import java.lang.\*;**  **import javax.servlet.\*;**  **import javax.servlet.http.\*;**  **import javax.servlet.jsp.\*;** |
| errorPage: 如果页面中有错误，则跳转到指定的资源  errorPage="/uri" 如果写“/”则代表当前应用的目录下，绝对路径。  如果不写“/”则代表相对路径。 |
| contextType: 等同于response.setContextType("text/html;charset=utf-8"); |
| pageEncoding: 告诉JSP引擎要翻译的文件使用的编码。 |
| isELIgnored: 是否支持EL表达式。 默认是false |

## include

静态包含：把其它资源包含到当前页面中。

<%@ include file="/include/header.jsp" %>

动态包含：

<jsp:include page="/include/header.jsp"></jsp:include>

两者的区别：翻译的时间段不同

前者：在翻译时就把两个文件合并

后者：不会合并文件，当代码执行到include时，才包含另一个文件的内容。

原则：能用静的就不用动的。

|  |
| --- |
| 静态： |
|  |
|  |
| 动态 |
|  |

## Taglib 【后面再学】

作用：在JSP页面中导入JSTL标签库。替换jsp中的java代码片段。

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

# JSP的6个常用动作

* <jsp:include > 动态包含
* <jsp:forward> 请求转发
* <jsp:param> 设置请求参数
* <jsp:useBean> 创建一个对象
* <jsp:setProperty> 给指定的对象属性赋值
* <jsp:getProperty> 取出指定对象的属性值

|  |
| --- |
|  |

# JSP的9大内置对象

* 内置对象是指在JSP的<%=%> 和<% %>中可以直接使用的对象

|  |  |  |
| --- | --- | --- |
| 对象名 | 类型 | 说明 |
| request | javax.servlet.http.HttpServletRequest |  |
| response | javax.servlet.http.HttpServletResponse |  |
| session | javax.servlet.http.HttpSession | 由session="true"开关 |
| application | javax.servlet.ServletContext |  |
| exception | java.lang.Throwable | 由isErrorPage="false"开关 |
| page | java.lang.Object当前对象this | 当前servlet实例 |
| config | javax.servlet.ServletConfig |  |
| out | javax.servlet.jsp.JspWriter | 字符输出流，相当于 printWriter对象 |
| pageContext | javax.servlet.jsp.PageContext |  |

## pageContext(重要)

### 概述

**本身也是一个域对象：它可以操作其它三个域对象（request session application）的数据**

### 常用方法

* void setAttribute(String name,Object o);
* Object getAttribute(String name);
* void removeAttribute(String name);

### 操作其它域对象的方法

* void setAttribute(String name,Object o，int Scope);
* Object getAttribute(String name,int Scope);
* void removeAttribute(String name,int Scope);

### scpoe的值：

* PageContext.PAGE\_SCOPE
* PageContext.REQUEST\_SCOPE
* PageContext.SESSION\_SCOPE
* PageContext.APPLICATION\_SCOPE

|  |
| --- |
|  |

### findAttribute(String name);

自动从page request session application依次查找，找到了就取值，结束查找。

|  |
| --- |
|  |

### pageContext获取其它的8个隐式对象

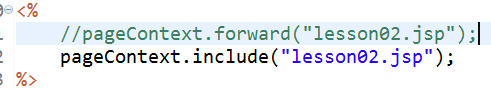
在普通类中可以通过PageContext获取其他JSP隐式对象。

|  |
| --- |
|  |

### pageContext的简易方法

pageContext.forward("2.jsp");

pageContext.include("2.jsp");



## 四大域对象：实际开发

* PageContext : pageConext 存放的数据在当前页面有效。**开发时使用较少。**
* ServletRequest: request 存放的数据在一次请求（转发）内有效。使用非常多。
* HttpSession: session 存放的数据在一次会话中有效。使用的比较多。如：存放用户的登录信息，购物车功能。
* ServletContext: application 存放的数据在整个应用范围内都有效。因为范围太大，**应尽量少用。**

# JSP的EL表达式

## EL概述

* EL表达式：expression language 表达式语言,用于简化jsp中java代码开发。
* 它不是一种开发语言，是jsp中**获取数据**的一种规范

## EL的具体功能

### 获取数据

* EL表达式只能获**取存在4个作用域中**的数据
* ${u} 原理： pageContext.findAttribute("u");
* EL获取对于null这样的数据，在页面中表现为空字符串
* ${u.name} == u.getName()方法
* 点（.） 运算符相当于调了getter方法，点后页面跟的是属性名。

|  |
| --- |
|  |

### 属性导航

**点点语法，可以获取属性对象的属性**

|  |
| --- |
|  |

### []运算符

* []运算符：点能做的，它也能做; 它能做的，点不一定能做
* 举例: ${user.name}== ${user['name']} == ${user["name"]}

|  |
| --- |
|  |

### 空运算empty

用于判断null，空字符串和没有元素的集合(即使集合对象本身不为null)都返回true

|  |
| --- |
|  |

### 三元运算符

|  |
| --- |
|  |
|  |

## EL的隐式对象：11个

|  |  |  |  |
| --- | --- | --- | --- |
| EL隐式对象引用名称 | 类型 | JSP内置对象名称 | 说明 |
| pageContext | javax.servlet.jsp.PageContext | pageContext | 一样的 |
| pageScope | java.util.Map<String,Object> | 没有对应的 | pageContext范围中存放的数据,页面范围 |
| requestScope | java.util.Map<String,Object> | 没有对应的 | 请求范围数据 |
| sessionScope | java.util.Map<String,Object> | 没有对应的 | 会话范围数据 |
| applicationScope | java.util.Map<String,Object> | 没有对应的 | 应用范围数据 |
| param | java.util.Map<String,String> | 没有对应的 | 一个请求参数 |
| paramValues | java.util.Map<String,String[]> | 没有对应的 | 重名请求参数 |
| header | java.util.Map<String,String> | 没有对应的 | 一个请求消息头 |
| headerValues | java.util.Map<String,String[]> | 没有对应的 | 重名请求消息头 |
| initParam | java.util.Map<String,String> | 没有对应的 | web.xml中全局参数 |
| cookie | java.util.Map<String,Cookie> | 没有对应的 | key:cookie对象的name值 |

|  |
| --- |
|  |

# JSTL

## 什么是JSTL

JSTL（JavaServerPages Standard Tag Library）JSP标准标签库

## JSTL的作用

使用JSTL实现JSP页面中逻辑处理。如判断、循环等。

## JSTL的使用

**常用标签介绍**

### **通用标签**: set、 out、 remove

|  |
| --- |
|  |

### **条件标签**：if choose

|  |
| --- |
|  |

### 迭带标签：foreach

**普通循环**

|  |
| --- |
|  |

**迭代器**

* c:forEach中的varStatus属性。
* 指向一个字符串，该字符串引用一个对象。 map.put("vs",一个对象);
* 这个对象记录着当前遍历的元素的一些信息：
* getIndex():返回索引。从0开始
* getCount():返回计数。从1开始
* isLast():是否是最后一个元素
* isFirst():是否是第一个元素

|  |
| --- |
|  |
|  |