

jsp的最佳实践:

Servlet: 控制器。重点编写java代码逻辑 (1、获取请求参数 2、调用业务 3、分发转向)

JSP: 代码显示模板。重点在于显示数据 xxx.jsp页面

JSP基本语法:

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。

<%! //声明

```
int num1 = 10;
public void ff() {
}
```

%>

声明 成员变量静态代码块

<% //小脚本

```
int num2 = 10;
num1++;
num2++;
```

%>

脚本里一般写JAVA代码处理业务逻辑

```
<%=num1 %> <!-- 表达式 -->
<%=num2 %>
```

表达式主要用于HTML里的一些赋值

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

4、3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指令

作用: 用于定义JSP页面的各种属性

Include指令 原则: 能使用静的就不使用动的

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>

taglib指令

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" %>

语法：

<%@ 指令名称 属性1= “属性值1” 属性2= “属性值2” 。。。 %>

<%@ 指令名称 属性1= “属性值1” %>

<%@ 指令名称 属性2= “属性值2” %>

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  定义JSP页面相关属性

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

<%
 if(5>3) {
 out.print("aaaaaaa");
 }
%>

java代码脚本

<c:if test="\${5>3}">
 gggggggggg
</c:if>

用导入的标签库替换 java脚本的 if语句

<%@ include file="/demo1/8.jsp" %> 页面包含的内容

5、6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

```

<%
    User user = new User();
    user.setUsername("tom");
    out.print(user.getUsername());
%>

<jsp:useBean id="user1" class="com.itheima.domain.User"></jsp:useBean>    //相当于 User user = new User();
<jsp:setProperty property="username" name="user1" value="jerry"/>    // 相当于user.setUsername("tom");
<jsp:getProperty property="username" name="user1"/>    // 相当于out.print(user.getUsername());

<!-- request.getRequestDispatcher("2.jsp?username=tom").forward(request, response); -->
<jsp:forward page="2.jsp">
    <jsp:param value="tom123" name="username"/>
    <jsp:param value="18" name="age"/>
</jsp:forward>

```

6、9内置对象

```

<%
    //pageContext.setAttribute("p", "pc", PageContext.PAGE_SCOPE);
    pageContext.setAttribute("p", "pp");

    //pageContext.setAttribute("p", "request", PageContext.REQUEST_SCOPE);
    request.setAttribute("p", "request");

    //pageContext.setAttribute("p", "session", PageContext.SESSION_SCOPE);
    session.setAttribute("p", "session");

    //pageContext.setAttribute("p", "application", PageContext.APPLICATION_SCOPE);
    application.setAttribute("p", "application");

    request.getRequestDispatcher("3.jsp").forward(request, response);
%>

```

最重要的一个：自动从page request session application依次查找取出，找到了就取值，结束查找。

```

<%--
    out.print(pageContext.findAttribute("p"));
--%>

```

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。

EL概述和基本语法

EL表达式: expression language 表达式语言

要简化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

jsp中的 `${pageContext.request.contextPath}` 返回的是 `/项目名`
使用的是绝对路径

`${pageContext.request.contextPath}` 相当于 `/CommodityInfo`

具体功能

a、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 (EL有很好的容错性)

`${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。

```
<!-- jsp脚本 -->
<%--
    User u = (User)request.getAttribute("user");
    out.print(u.getUsername());
--%>
<!-- EL表达式 -->
    ${u.username} == u.getUsername();
```

属性导航

`${u.address.city}`

[]运算符: 点能做的, 它也能做; 它能做的, 点不一定能做

`${u.username}` == `${u['username']}` == `${u["username"]}`

```
<%
    List list = new ArrayList();
    list.add("aaa");
    list.add("bbb");
    list.add("ccc");

    request.setAttribute("list", list);
%>
    ${list[1]}

<%
    Map map = new HashMap();
    map.put("a", "aaaaa");
    map.put("b", "bbbbb");
    map.put("c", "ccccc");

    request.setAttribute("m", map);
%>
    ${m["a"]} == ${m.a} 此处的a不代表属性了, 代表map中的key
```

b、运算

empty

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

```
<%
String str1 = null;
request.setAttribute("str1", str1);
String str2 = "";
request.setAttribute("str2", str2);
String str3 = "abc";
request.setAttribute("str3", str3);

List list1 = new ArrayList();
request.setAttribute("list1", list1);

List list2 = new ArrayList();
list2.add("asdf");
request.setAttribute("list2", list2);

User user = new User();
request.setAttribute("user", user);
%>
${empty str1 } == true<br/>
${empty str2 } == true<br/>
${empty str3 } == false<br/>
${empty list1 } == true<br/>
${empty list2 } == false<br/>
${empty user } == false<br/>

true == true
true == true
false == false
true == true
false == false
false == false
```

三元运算符

```
${ empty list3 ? "你还没有买任何商品": "你买的商品有" }
```

c、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 e
--------	-------------------------------	-------	------------------------

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。

使用JSTL

0)导入jar包



1) 在JSP页面添加taglib指令

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

2) 使用JSTL标签

```
<c:if test=""></c:if>
```

4、常用标签介绍

核心标签库：

> 通用标签: set、 out、 remove

设置变量: `<c:set var="num" value="${10+5}" scope="page"></c:set>
`

输出数据: `<c:out value="${num}"></c:out>
`

移除变量: `<c:remove var="num"/>
`

移除后输出: `<c:out value="${num}" default="aaaa"></c:out>`

> 条件标签: if choose

```
<c:set var="num" value="10"></c:set>
<c:if test="${ num <20 }">
    aaaaaaaa
</c:if>
```

```
<c:set var="num1" value="3"></c:set>
<c:choose>
    <c:when test="${num1==1 }">
        第一名
    </c:when>
    <c:when test="${num1==2 }">
        第二名
    </c:when>
    <c:when test="${num1==3 }">
        第三名
    </c:when>
    <c:otherwise>
        没获选
    </c:otherwise>
</c:choose>
```

> 迭带标签: foreach

普通循环

```
<!--
    var ---- 声明变量
    begin ---- 初始化
    end----- 结束条件
    step----- 步长
-->
<c:forEach var="i" begin="2" end="10" step="2" >
    <hr/>${i }
</c:forEach>
```

迭带器

```
//遍历: list set map和数组
List list = new ArrayList();
list.add("aaa");
list.add("bbb");
list.add("ccc");
list.add("ddd");
list.add("eee");
list.add("fff");
pageContext.setAttribute("list", list);
%>
<table border="1">
    <tr>
        <th>数据</th>
        <th>索引</th>
        <th>计数</th>
        <th>第一个</th>
        <th>最后一个</th>
    </tr>
```

for(类型 变量名: 数组或集合)


```

<!-- 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
-->
<c:forEach items="${list}" var="l" varStatus="vs">
    <tr class="${vs.index%2==0?'odd':'even'}">
        <td>${l}</td>
        <td>${vs.index}</td>
        <td>${vs.count}</td>
        <td>${vs.first}</td>
        <td>${vs.last}</td>
    </tr>
</c:forEach>
</table>

<style type="text/css">
    .odd{
        background-color: #c3f3c3;
    }
    .even{
        background-color: #f3c3f3;
    }
</style>

```

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