day13-JSP

1、熟悉什么是JSP

2、掌握JSP的语法

3、掌握JSP的指令

4、掌握JSP的动作

5、掌握JSP的内置对象

6、掌握EL表达式的使用

7、掌握域对象

8、熟悉什么是JSTL

9、掌握JSTL的使用

**第十三章 JSP**

13.1 jsp是什么

全称： Java Server Pages,java服务器页面。和Servlet一样，是sun公司定义的一种动态网页开发技术。

特点：基于html模版，可以在html模版嵌入java代码和jsp中的标签。

备注：html静态页面。

CSS:修饰美化html页面。

JavaScript:动态修改html页面和css样式。

Servlet：运行在服务器上的java程序。适合编写java代码，写网页困难

jsp:适合编写动态内容，不适合写java代码，主要用于显示页面

13.2 为什么要用jsp

* + 1. jsp性能好，可以在html页面中动态嵌入元素
    2. 服务器调用的是已经编译好的JSP文件
    3. JSP基于Java Servlet Api,有很多强大企业的支持。
    4. JSP可以与处理业务逻辑的Servlet一起使用，该模式被Java Servlet模版引擎所支持。

13.3 JSP优势

3.1 与纯 Servlet 相比：JSP可以很方便的编写或者修改HTML网页而不用去面对大量的println语句

3.2 与JavaScript相比：虽然JavaScript可以在客户端动态生成HTML，但是很难与服务器交互，因此不能提供复杂的服务，比如访问数据库和图像处理等等。

3.3 与静态HTML相比：静态HTML不包含动态信息

13.4 JSP语法

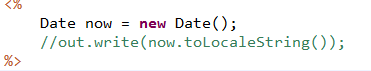
13.4.1 JSP语法格式

JSP页面中可以包含任意量的Java语句、变量、方法或表达式

​ 语法格式： <% java代码 %>

出现在JSP对应Servlet的service方法中。

<%声明的变量：局部%>



​ 声明全局变量： <%! int i=10;%>

定义JSP对应Serlvet的成员（变量和方法、静态方法）

注意：开发中很少用。面试和考试经常考。

​ 输出变量： <%=2+3%>等价于输出语句 (注意：不能使用分号来结束表达式)



13.4.2 JSP注释

格式： <%-- 网页注释 —>：安全，省流量

网页注释：<!-- 网页注释 --> ,特点，不安全，耗流量

13.5 JSP指令

告诉JSP引擎如何jsp文件中的内容

​ 语法：

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

等价于：

<%@指令 属性1=”值1” %>.

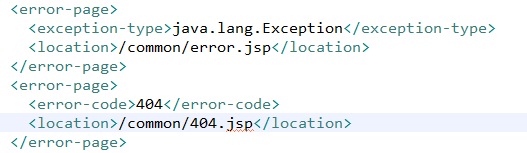
<%@指令 属性2=”值2”%>.

13.5.1 page指令

作用：用于定义JSP页面的各种属性。

|  |  |
| --- | --- |
| 属性 | 作用 |
| language | 告知引擎，脚本用的是java。默认java，支持java。不写也行 |
| extends | 告知引擎，JSP对应的Servlet的父类是哪个。不需要写，也不要改 |
| \*import | 告知引擎，导入哪些包  <%@page import=”java.util.Date**,**java.util.UUID”%>或者  <%@page import=”java.util.Date”%>  <%@page import=”java.util.UUID”%> |
| \*session | 告知引擎是否产生HttpSession对象，即是否在代码中调用request.getSession()。默认值true |
| buffer | JspWriter用于输出JSP的内容到页面上。告知引擎，设定他的缓存大小。默认8Kb |
| \*errorPage | 告知引擎，当前页面出现了异常，应该转发到哪个页面上（路径写法：/代表着当前应用） |
| \*isErrorPage | 告知引擎，是否抓住异常。如果该属性为true，页面中就可以使用exception隐含对象，打印异常的详细信息 |
| contentType | 告知引擎，响应正文的MIME类型 |
| pageEncoding=”UTF-8” | 相当于，告知了引擎用UTF-8读jsp，还response.setContentType(“text/html;charset=UTF-8”); |
| \*isELIgnored | 告知引擎，是否忽略EL表达式。默认值是false，不忽略 |

扩展：配置全局错误页面：web.xml



13.5.2 inclue指令

include指令：静态包含，把其它资源包含到当前页面中

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

jsp:include动作: 动态包含

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

两者的区别：

静态包含：在翻译时就把两个文件进行合并

动态包含：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

13.5.3 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" %>

13.6 JSP的6个动作

使用标签的形式表示一段java代码，例如：

jsp:include 动态包含

jsp:forward请求转发

jsp:param 请求设置参数

jsp:useBean 创建一个对象

jsp:setProperty给指定的对象属性赋值

jsp:getProperty取出指定的对象属值

13.7 内置对象

JSP对应的Servlet的service方法中的局部变量

|  |  |  |
| --- | --- | --- |
| JSP隐含对象名称 | 类型 | 备注 |
| request | javax.servlet.http.HttpServletRequest |  |
| response | javax.servlet.http.HttpServletResponse |  |
| session | javax.servlet.http.HttpSession | page指令有开关 |
| application | javax.servlet.ServletContext |  |
| config | javax.servlet.ServletConfig |  |
| page | javax.servlet.http.HttpServlet | 当前jsp对应Servlet的实例引用 |
| exception | java.lang.Throwable | page指令有开关 |
| out | javax.servlet.jsp.JspWriter | 字符输出流。相当于PrintWriter |
| pageContext | javax.servlet.jsp.PageContext | 很重要 |

|  |
| --- |
| <%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*  pageEncoding=*"UTF-8"* isErrorPage=*"true"*%>  <!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>JSP内置对象</title>  </head>  <body>  <%  //页面上下文对象  pageContext.setAttribute("msg", "我是PageContext对象");  System.out.println(pageContext.getAttribute("msg"));  //请求对象  System.out.println("请求对象：" + request);  //响应对象  response.getWriter().print("OK");  //当前页面对象  System.out.println(page.getClass());//输出到控制台  //输出对象  out.print(48);//输出到浏览器：48  out.write(48);//输出到浏览器：0  **int** no = 101;  //会话对象  System.out.println(session);  //全局对象  System.out.println(application.getRealPath("/"));  //配置对象--初始化参数  System.out.println(config.getServletName());  //异常对象  System.out.println(exception);  System.out.println(application);  application.setAttribute("author", "1711");  %>  <h1>  jsp脚本：<%=no%></h1>  <h1>  jsp脚本：<%  out.print(no);  %>  </h1>  </body>  </html> |

13.8 PageContext抽象类（很重要）

1、本身是一个域（范围）对象，还能操作其他3个域对象中的属性

* 页面范围：

void setAttribute(String name,Object value);

void removeAttribute(String name);

Object getAttribute(String name);

* 操作其他域对象中的属性：

void setAttribute(String name,Object value,int scope);

void removeAttribute(String name,int scope);

Object getAttribute(String name,int scope);

int scope:常量，PageContext PAGE\_SCOPE(页面) REQUEST\_SCOPE（请求）SESSION\_SCOPE（会话）APPLICATION\_SCOPE（应用）

* 在四个范围中依次查找指定名称对象的方法：

PAGE(页面) REQUEST（请求）SESSION（会话）APPLICATION（应用）

Object findAttribute(String name);

2、获取其他八个JSP隐含对象(自定义标签时用)

PageContext.getRequest();

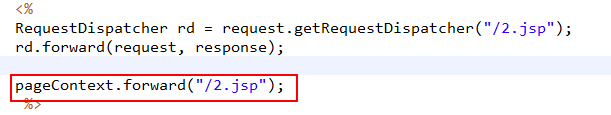
getServletContext();

getServletConfig();

getException();

get...

3、提供转发和包含的简易方法



pageContext.include(String url);

|  |
| --- |
| 四大作用域对象  实际开发中：存放数据的。  PageContext:页面范围。最小。开发中很少用。演示代码。  ServletRequest:请求范围。不同的资源，只有用转发。  HttpSession:会话范围。多次请求共享数据，但不同的客户端不能共享。  ServletContext:应用范围。最大。尽量少用。用时要做同步处理。 |

**第十四章 EL表达式**

14.1 EL概述和基本语法

​ EL表达式:expression language表达式语言

​ 目的: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

​ 格式如： ${EL表达式} 等价于findAttribute(name)

14.2 EL的具体功能

​ 案例1：获取实体类中的属性值

|  |
| --- |
| <%  User user = **new** User();  user.setName("gggg");  Address address = **new** Address();  address.setAddr("北京市海淀区");  user.setAddress(address);  session.setAttribute("user", user);  /\* request.getRequestDispatcher("testEl.jsp").forward(request, response); \*/  response.sendRedirect("testEl.jsp");  %>  <!-- EL表达式中的.表示调用该属性的get方法 -->  <%-- 我是：$住在{user.name } , 住在 ${user.address.addr } --%> |

14.3 使用list和map

|  |
| --- |
| <%  List<String> list = **new** ArrayList<String>();  list.add("aa");  list.add("bb");  list.add("cc");  pageContext.setAttribute("list", list);  Map<String, String> map = **new** HashMap<String, String>();  map.put("aa", "11");  map.put("bb", "22");  map.put("cc", "33");  pageContext.setAttribute("map", map);  %>  <h1>1.以数组的下标形式获取list值</h1>  ${list[0]}  <br> ${list[1]}  <br> ${list[2]}  <br>  <h1>2.以提供的方法get(index)</h1>  ${list.get(0)}  <h1>3.使用El表达式获取map中的值(以key的形式获取)</h1>  ${map.aa}  <br> ${map.bb}  <br> ${map.cc}  <br>  <h1>4.使用El表达式获取map中的值</h1>  ${map['aa']} |

14.4 使用EL表达式的empty关键

|  |
| --- |
| <%  String s1 = "";  pageContext.setAttribute("s1", s1);  String s2 = **null**;  pageContext.setAttribute("s2", s2);  String s3 = "122222";  pageContext.setAttribute("s3", s3);  List list1 = **new** ArrayList();  pageContext.setAttribute("list1", list1);  %>  <!-- empty关键只要内容是空true -->  ${**empty** s1}  <br> ${**empty** s2}  <br> ${**empty** s3}  <br> ${**empty** list1}  <br> |

**第十五章 JSTL**

15.1 什么是JSTL

JSL:全称JavaServerPages Standard TagLibrary,JSP标准标签库

15.2 JSTL的作用

实现JSP页面中逻辑处理。如判断， 循环等；

15.3 使用JSTL

必须在JSP页面添加tablib指令库

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

15.4 常用标签介绍

15.4.1 通用标签set,out,remove

|  |
| --- |
| <!-- 基础标签:声明变量，输出变量，移除变量，变量默认值 -->  <h3>基础标签:声明变量，输出变量，移除变量，变量默认值</h3>  <!--1. var：是变量名 value:变量的值(必须是EL表达式)-->  <p:set var=*"k"* value="${1+1}"></p:set>  <!--2. 输出变量k value:使用EL表示表示变量-->  移除前输出的内容：<p:out value="${k}"></p:out><br>  <!--3.移除指定变量 -->  <p:remove var=*"k"*/>  移除后输出的内容：<p:out value="${k}"></p:out><br>  <!--4.给指定变量赋默认值 -->  默认值：<p:out value="${m}" default=*"123456"*></p:out> |

15.4.2 条件标签if,choose

|  |
| --- |
| <!-- 条件标签：if choose -->  <!-- test属性中是条件，但是条件需要使用EL表达式来书写 -->  <h3>条件标签：if</h3>  <c:if test="${8>2 }">  8大于2是成立的  </c:if>  <c:if test="${8<2 }">  8小于2是成立的  </c:if>  <br>  <%-- 如果只是一个基本数据类型直接书写不需要${} --%>  <c:set var=*"m"* value="${5}"></c:set>  <c:if test="${m>3}">  5大于3是成立的  </c:if> |

|  |
| --- |
| <h3>条件标签：choose(等价于java中switch)</h3>  <%-- 测试成绩等级 >90 优秀 >80 良好 >70 中等 >60及格--%>  <c:set var=*"score"* value="${80}"></c:set>  <c:choose>  <c:when test="${score>=90 }">优秀</c:when>  <c:when test="${score>=80 }">良好</c:when>  <c:when test="${score>=70 }">中等</c:when>  <c:when test="${score>=60 }">及格</c:when>  <c:otherwise>不及格</c:otherwise>  </c:choose> |

15.4.3 迭代标签foreach

<!-- 遍历for:输出1到10 的值 -->

<!--var: 变量，把遍历的每一个值都存储在变量中进行输出

begin：开始 如果是变量使用EL表达式表示

end:结束 如果是变量使用EL表达式表示

step:间隔的长度

-->

示例代码：

|  |
| --- |
| <c:forEach var=*"i"* begin=*"1"* end=*"10"* step=*"2"*>  ${i}<br>  </c:forEach> |

foreach遍历

|  |
| --- |
| <h3>测试list集合遍历获取学生列表</h3>  <table border=*"1"* width=*"80%"* bordercolor=*"red"* cellspacing=*"0"*  align=*"center"*>  <tr>  <th>学号</th>  <th>姓名</th>  <th>成绩</th>  <th>班级</th>  <th>是否是第一个</th>  <th>是否是最后一个</th>  <th>计数count</th>  <th>索引index</th>  </tr>  <!-- varStatus:变量状态：遍历出的每一项内容的状态：  isFirst() first  isLast() last  getCount() count 计数 重要的  getIndex() index  -->  <!-- var :遍历出的每一项使用变量先存储  items：集合(使用El表达式)  -->  <c:forEach var=*"stu"* items="${students}" varStatus=*"vs"*>  <tr>  <td>${stu.id}</td>  <td>${stu.name}</td>  <td>${stu.score}</td>  <td>${stu.classes}</td>  <td>${vs.first}</td>  <td>${vs.last}</td>  <td>${vs.count}</td>  <td>${vs.index}</td>  </tr>  </c:forEach>  </table> |

## **JSTL中的Functions**

taglib指令导入：

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

* fn:contains函数：用于判断在源字符串中是否包含目标字符串。
* fn:containsIgnoreCase函数：用于判断在源字符串中是否包含目标字符串，并且在判断时忽略大小写。
* fn:startsWith函数：用于判断源字符串是否以指定的目标字符串开头。
* fn: endsWith函数：用于判断源字符串是否以指定的目标字符串结尾。
* fn:indexOf函数：用于在源字符串中查找目标字符串，并返回源字符串中最先与目标字符串匹配的第一个字符的索引。
* fn:replace函数：用于把源字符串中的一部分替换为另外的字符串，并返回替换后的字符串。
* fn:substring函数：用于获取源字符串中的特定子字符串。
* fn:substringBefore函数：用于获取源字符串中指定子字符串之前的子字符串。
* fn: substringAfter函数：用于获取源字符串中指定子字符串之后的子字符串
* fn:split函数：用于将源字符串拆分为一个字符串数组。
* fn:join函数：用于将源字符串数组中的所有字符串连接为一个字符串。
* fn:toLowerCase函数：用于将源字符串中的所有字符改为小写。
* fn: toUpperCase函数：用于将源字符串中的所有字符改为大写。
* fn:trim函数：用于将源字符串中的开头和末尾的空格删除。
* fn:escapeXml函数：用于将源字符串中的字符“<”、“>”、“””和“&”等转换为转义字符。
* fn:length函数：用于返回字符串中的字符的个数，或者集合和数组的元素的个数