动态页面技术（JSP/EL/JSTL）

教学导航

教学目标	案例：完成商品的列表的展示
教学方法	

一、JSP 技术

1. jsp 脚本和注释

jsp 脚本：

1）<%java 代码%> ----- 内部的 java 代码翻译到 service 方法的内部

2）<%=java 变量或表达式> ----- 会被翻译成 service 方法内部 out.print()

3）<%!java 代码%> ---- 会被翻译成 servlet 的成员的内容

jsp 注释： 不同的注释可见范围是不同

1）Html 注释：<!--注释内容--> ---可见范围 jsp 源码、翻译后的 servlet、页面 显示 html 源码

2）java 注释：//单行注释 /*多行注释*/ --可见范围 jsp 源码 翻译后的 servlet

3）jsp 注释：<%--注释内容--%> ----- 可见范围 jsp 源码可见

2. jsp 运行原理-----jsp 本质就是 servlet（面试）

jsp 在第一次被访问时会被 Web 容器翻译成 servlet，在执行

过程：

第一次访问---->helloServlet.jsp---->helloServlet_jsp.java---->编译运行

PS：被翻译后的 servlet 在 Tomcat 的 work 目录中可以找到

3．jsp 指令（3 个）

jsp 的指令是指导 jsp 翻译和运行的命令，jsp 包括三大指令：

1）page 指令 --- 属性最多的指令（实际开发中 page 指令默认）

属性最多的一个指令，根据不同的属性，指导整个页面特性

格式：<%@ page 属性名 1= "属性值 1" 属性名 2= "属性值 2" ...%>

常用属性如下：

language：jsp 脚本中可以嵌入的语言种类

pageEncoding：当前 jsp 文件的本身编码---内部可以包含 contentType

contentType：response.setContentType(text/html;charset=UTF-8)

session：是否 jsp 在翻译时自动创建 session

import：导入 java 的包

errorPage：当当前页面出错后跳转到哪个页面

isErrorPage：当前页面是一个处理错误的页面

2）include 指令

页面包含（静态包含）指令，可以将一个 jsp 页面包含到另一个 jsp 页面中

格式：<%@ include file="被包含的文件地址"%>

3）taglib 指令

在 jsp 页面中引入标签库（jstl 标签库、struts2 标签库）

格式：<%@ taglib uri="标签库地址" prefix="前缀"%>

4．jsp 内置/隐式对象（9 个）----- 笔试

jsp 被翻译成 servlet 之后，service 方法中有 9 个对象定义并初始化完毕，我们在 jsp 脚本中可以直接使用这 9 个对象

名称	类型	描述
out	javax.servlet.jsp.JspWriter	用于页面输出
request	javax.servlet.http.HttpServletRequest	得到用户请求信息，
response	javax.servlet.http.HttpServletResponse	服务器向客户端的回应信息

config	javax.servlet.ServletConfig	服务器配置，可以取得初始化参数
session	javax.servlet.http.HttpSession	用来保存用户的信息
application	javax.servlet.ServletContext	所有用户的共享信息
page	java.lang.Object	指当前页面转换后的 Servlet 类的实例
pageContext	javax.servlet.jsp.PageContext	JSP 的页面容器
exception	java.lang.Throwable	表示 JSP 页面所发生的异常，在错误页中才起作用

(1) out 对象

out 的类型：JspWriter
out 作用就是想客户端输出内容----out.write()
out 缓冲区默认 8kb 可以设置成 0 代表关闭 out 缓冲区 内容直接写到 respons 缓冲器

(2) pageContext 对象

jsp 页面的上下文对象，作用如下：
page 对象与 pageContext 对象不是一回事
1）pageContext 是一个域对象
setAttribute(String name,Object obj)
getAttribute(String name)
removeAttrbute(String name)

pageContext 可以向指定的其他域中存取数据
setAttribute(String name,Object obj,int scope)
getAttribute(String name,int scope)
removeAttrbute(String name,int scope)
findAttribute(String name)

---依次从 pageContext 域，request 域，session 域，application 域中获取属性，在某个域中获取后将不在向后寻找

四大作用域的总结：

page 域：当前 jsp 页面范围
request 域：一次请求
session 域：一次会话
application 域：整个 web 应用

2）可以获得其他 8 大隐式对象

例如： pageContext.getRequest()
pageContext.getSession()

5. jsp 标签（动作）

1）页面包含（动态包含）：<jsp:include page="被包含的页面"/>
2）请求转发：<jsp:forward page="要转发的资源" />

静态包含与动态包含的区别？

二、EL 技术

1. EL 表达式概述

EL（Express Lanuage）表达式可以嵌入在 jsp 页面内部，减少 jsp 脚本的编写，EL 出现的目的是要替代 jsp 页面中脚本的编写。

2. EL 从域中取出数据(EL 最重要的作用)

jsp 脚本：<%=request.getAttribute(name)%>

EL 表达式替代上面的脚本：${requestScope.name}

EL 最主要的作用是获得四大域中的数据，格式**${EL 表达式}**
EL 获得 pageContext 域中的值：${pageScope.key};
EL 获得 request 域中的值：${requestScope.key};
EL 获得 session 域中的值：${sessionScope.key};
EL 获得 application 域中的值：${applicationScope.key};
EL 从四个域中获得某个值${key};
---同样是依次从 pageContext 域，request 域，session 域，application 域中获取属性，在某个域中获取后将不在向后寻找

1）获得普通字符串

2）获得 User 对象的值

3）获得 List<User>的值

3. EL 的内置对象 11 个

pageScope,requestScope,sessionScope,applicationScope
---- 获取 JSP 中域中的数据
param,paramValues - 接收参数.
相当于 request.getParameter() rrquest.getParameterValues()
header,headerValues - 获取请求头信息
相当于 request.getHeader(name)
initParam - 获取全局初始化参数
相当于 this.getServletContext().getInitParameter(name)
cookie - WEB 开发中 cookie
相当于 request.getCookies()---cookie.getName()---cookie.getValue()
pageContext - WEB 开发中的 pageContext.
pageContext 获得其他八大对象

${pageContext.request.contextPath}
相当于

<%=pageContext.getRequest().getContextPath%>　这句代码不能实现获得 WEB 应用的名称

4．EL 执行表达式

例如：

${1+1}

${empty user}

${user==null?true:false}

三、JSTL 技术

1．JSTL 概述

JSTL (JSP Standard Tag Library)，JSP 标准标签库，可以嵌入在 jsp 页面中使用标签的形式完成业务逻辑等功能。jstl 出现的目的同 el 一样也是要代替 jsp 页面中的脚本代码。JSTL 标准标准标签库有 5 个子库，但随着发展，目前常使用的是他的核心库

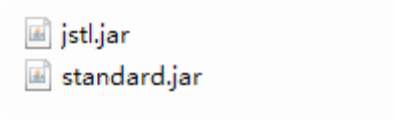
标签库	标签库的 URI	前缀
Core	http://java.sun.com/jsp/jstl/core	c
I18N	http://java.sun.com/jsp/jstl/fmt	fmt
SQL	http://java.sun.com/jsp/jstl/sql	sql
XML	http://java.sun.com/jsp/jstl/xml	x
Functions	http://java.sun.com/jsp/jstl/functions	fn

2．JSTL 下载与导入

JSTL 下载：

从 Apache 的网站下载 JSTL 的 JAR 包。进入“http://archive.apache.org/dist/jakarta/taglibs/standard/binaries/”网址下载 JSTL 的安装包。jakarta-taglibs-standard-1.1.2.zip，然后将下载好的 JSTL 安装包进行解压 此时 ,在 lib 目录下可以看到两个 JAR 文件 ,分别为 jstl.jar 和 standard.jar。

其中，jstl.jar 文件包含 JSTL 规范中定义的接口和相关类，standard.jar 文件包含用于实现 JSTL 的.class 文件以及 JSTL 中 5 个标签库描述符文件（TLD）



将两个 jar 包导入我们工程的 lib 中

使用 jsp 的 taglib 指令导入核心标签库

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

3. JSTL 核心库的常用标签

1）<c:if test=”” >标签

其中 test 是返回 boolean 的条件

2）<c:forEach>标签

使用方式有两种组合形式：

```
<!-- forEach模拟
    for(int i=0;i<=5;i++){
        syso(i)
    }
-->
<c:forEach begin="0" end="5" var="i">
   ${i }<br/>
</c:forEach>
```

示例：

1）遍历 List<String>的值

2）遍历 List<User>的值

3）遍历 Map<String,String>的值

4）遍历 Map<String,User>的值

5）遍历 Map<User,Map<String,User>>的值
entry.key-----User
entry.value------List<String,User>

四、javaEE 的开发模式

1．什么是模式

模式在开发过程中总结出的“套路”，总结出的一套约定俗成的设计模式

2．javaEE 经历的模式

model1 模式：
技术组成：jsp+javaBean
model1 的弊端：随着业务复杂性 导致 jsp 页面比较混乱
model2 模式
技术组成：jsp+servlet+javaBean
model2 的优点：开发中 使用各个技术擅长的方面
servlet：擅长处理 java 业务代码
jsp：擅长页面的现实

MVC：---- web 开发的设计模式
M：Model---模型 javaBean：封装数据
V：View-----视图 jsp：单纯进行页面的显示
C：Controller----控制器 Servelt：获取数据--对数据进行封装--传递数据--指派显示的 jsp 页面

3．javaEE 的三层架构

服务器开发时 分为三层

web 层：与客户端交互

service 层：复杂业务处理

dao 层：与数据库进行交互

开发实践时 三层架构通过包结构体现

MVC 与三层架构有什么关系？

总结：

EL 表达式

从域中取出数据 ${域中存储的数据的 name}

${pageContext.request.contextPath}

JSTL 标签(核心库)

<%@ taglib uri=”” prefix=”c”%>

<c:if test=”” >

<c:forEach items=”数组或集合” var=”数组或集合中的每一个元素” >

javaEE 三层架构+MVC

web 层：收集页面数据，封装数据，传递数据，指定响应 jsp 页面

service 层：逻辑业务代码的编写

dao 层：数据库的访问代码的编写