JSP

课前默写

1、写出过滤器的使用步骤
2、写出过滤器的配置方式
3、写出过滤器的执行顺序
4、写出监听器的类型
5、写出过滤器和Servlet的区别
6、写出自己对监听器的理解

课程回顾

1、过滤器的生命周期
2、过滤器的执行步骤
3、过滤器的的应用场景
4、监听器的类别
5、监听器的应用场景

今日内容

1、什么是JSP
2、JSP的语法
3、JSP的指令
4、JSP的动作
5、JSP的内置对象
6、EL表达式的使用
7、域对象
8、什么是JSTL
9、JSTL的使用

教学目标

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

```
    2.1 jsp性能好，可以在html页面中动态嵌入元素
    2.2  服务器调用的是已经编译好的JSP文件
    2.3 JSP基于Java Servlet Api,有很多强大企业的支持。
    2.4 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 %> 声明局部变量： <% int i=10;%> 声明全局变量： <%! int i=10;%> 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： ,特点，不安全，耗流量

13.4.3 代码演示

```
<%--指令，页面的设置--%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" %>
    <%--1、模板元素 --%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>第一个JSP页面</title>
</head>
<body>
<%--我是jsp特有的注释 --%>
<!--我是HTML的注释  -->
<a href="http://www.baidu.com">百度一下</a>
<%--jsp脚本 --%>
<%! int c=10; %>
```

```
<!-- 4、jsp标签 -->
<jsp:include page=""></jsp:include>
</body>
</html>
```

```
<%--指令，页面设置信息 --%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!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>第二个JSP页面</title>
</head>
<body>
<%--我是JSP的注释，我是不会出现在对应的java类中的 --%>
<!--我是HTML的注释，会写出到页面  -->

<%-- <%!System.out.println(123); %>带! 的意思是声明 --%>
<!--jsp脚本之声明  -->
<!--声明一个属性  -->
<%!private int count; %>
<!--声明并且实现方法  -->
<%!public void show(){
    System.out.println(count);
} %>
<%!private int age=10; %>
<%-- <%!show(); %> --%>
<!--jsp脚本之方法内使用  -->
<!-- 局部变量 -->
<%int c=1;%>
<%count++;
    c++;%>
<%
System.out.println("局部变量: "+c);
System.out.println("全局变量: "+count);
%>
<%System.out.println("请求的IP地址: "+request.getRemoteAddr()); %>
<!--jsp脚本之三: 赋值、输出内容  -->
<h1>访问次数: <%=count %></h1>

</body>
</html>
```

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” 。。。。。%> 示例：<%@ page language="*java*" contentType="text/html; charset=UTF-8" pageEncoding="*UTF-8*"%>

13.5.1 page指令

作用：用于定义JSP页面的各种属性。

```
import和java代码的含义一样
       <%@ pageimport="java.util.Date,java.util.List"%>
             或者：
         <%@ pageimport="java.util.Date"%>
         <%@ pageimport="java.util.List"%>
session:是否会自动创建session对象，默认值为true;
buffer:JSP中有java.servlet.jsp.JspWriter输出字符流。设置输出数据的缓存大小。默认8kb.
errorPage:如果页面中有错误，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
isErrorPage:是否创建throwable对象。默认是false
contentType:等同于response.setContentType(“text/html”; charset=utf-8);服务器发送客户端的内容编码
pageEncoding: Jsp文件本身的编码
isELIgnored: 是否支持EL表达式。 默认是false,支持表达式，是true,不支持表达式，${1+1};false输出结果2
true按照原样输出
```

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" %>
          动态包含：  <jsp:include page=“header.jsp"></jsp:include>
     两者的区别：翻译的时间段不同(可在tomcat工作目录下查看)
                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

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	
pageContext	javax.servlet.jsp.JspWriter	
out	javax.servlet.jsp.PageContext	javax.servlet.jsp.JspWriter

代码演示:

```
<%@ 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对象

13.8.1 域对象

可以操作其他三个域对象(request,session,application)的数据

```
    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,Objecto,int Scope);
    Object getAttribute(String name,intScope);
    void removeAttribute(String name,intScope);
Scope作用域，值如下：
    PageContext.PAGE_SCOPE
    PageContext.REQUEST_SCOPE
    PageContext.SESSION_SCOPE
    PageContext.APPLICATION_SCOPE

findAttribute(Stringname)自动从page,request ， session ， application依次查找，
                          找到了就取值，结束查找   （作用域的范围由小到大）
```

13.8.2 它可以创建其它的8个隐式对象

在普通类中可以通过PageContext获取其它JSP隐式对象,具体如下:

```
getException方法返回exception隐式对象
getPage方法返回page隐式对象
getRequest方法返回request隐式对象
getResponse方法返回response隐式对象
getServletConfig方法返回config隐式对象
getServletContext方法返回application隐式对象
getSession方法返回session隐式对象
getOut方法返回out隐式对象
```

13.8.3 提供了简易方法

```
pageContext.forward(“2.jsp”);
pageContext.include(“2.jsp”);
```

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：${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

```
<% @  tagliburi="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

<c:if>

```
<!-- 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>
```

<c:choose>

```
<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

for基础遍历

```
<!-- 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

    for( int i=0;i<10;i++){
     System.out.println(i);
    }
     -->
```

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>
```

作业题

1、使用JSP实现登录注册和图书浏览功能

面试题

1、Jsp和Servlet的区别
2、Jsp的执行原理
3、说说Jsp的隐藏对象有哪些
4、说出Jsp内置对象以及方法