JSP概述

2019年3月1日 10:25

1. JSP概述

JSP是javaee提供的动态web资源开发技术之一。

Servlet技术本质上是java代码，在其中编写HTML页面是比较困难的，为了简化这个过程，SUN公司提供了JSP技术，看起来非常类似于一个HTML页面，但是可以直接在其中编写java代码，从而将之前Servlet在java代码中嵌入html的过程，改为了JSP在HTML中嵌入java代码，便于开发动态页面。

2. jsp原理

JSP页面会在第一次被访问到时，会被JSP翻译引擎翻译成Servlet，本质上仍然是Servlet执行，向浏览器输出了数据。

实验：

开发jsp页面并进行访问，观察[tomcat]/work/Catalina/[虚拟主机]\[web应用]\org\apache\jsp\xxx.java文件，了解jsp被翻译成对应的servlet的细节。

JSP语法

2019年3月1日 10:42

1. 模板元素

jsp页面中的html内容，称之为模板元素。

```
<!DOCTYPE HTML>
<html>
  <head>
  </head>
  <body>
    this is first jsp!
  </body>
</html>
```

```
out.write("\r\n");
out.write("<!DOCTYPE HTML>\r\n");
out.write("<html>\r\n");
out.write("  <head>\r\n");
out.write("  </head>\r\n");
out.write("  <body>\r\n");
out.write("  \tthis is first jsp!\r\n");
out.write("  </body>\r\n");
out.write("</html>\r\n");
```

模板元素在被翻译到Servlet时，直接被out.write输出到了浏览器。

2. 脚本表达式

语法：<%= java表达式 %>

```
<%= new Date().toLocaleString() %>
```

```
out.print( new Date().toLocaleString() );
```

脚本表达式在被翻译到Servlet的过程中，直接计算表达式的值，通过out.print输出到浏览器。

3. 脚本片段

语法：<% 若干java语句 %>

脚本片段会被直接复制粘贴到翻译过来的Servlet的对应位置。

```
<%
    for(int i =0;i<5;i++){
%>
    <font color='red'>this is first jsp!</font><br>

<% } %>
```

```
      for(int i =0;i<5;i++){

    out.write("\r\n");
    out.write("  \t\t<font color='red'>this is
    first jsp!</font><br>\r\n");
    out.write("  \t\t\r\n");
    out.write("  \t");
}
```

jsp的某个脚本片段语法可以不完整，只要保证在翻译成Servlet时，整个Servlet语法完整即可。

4. JSP声明

基本语法：<%! 若干语句%>

JSP声明的内容，会被复制粘贴到翻译过来的Servlet的类的内部，成为类的成员。

主要用来为翻译过来的Servlet增加类的成员 - 包括成员属性 成员方法 静态成员 静态代码块 内部类 等等

5. JSP注释

基本语法：<%-- 被注释的内容 --%>

<!-- abc -->	html注释被当做模板元素输出到了浏览器，但浏览器不予展示
<% //ghi %>	java注释被当做脚本片段翻译到了Servlet中，但代码被注释不能执行
<%-- def --%>	jsp注释在翻译成Servlet过程中被抛弃，不会出现在翻译出来的Servlet中

6. JSP指令

基本语法：<%@jsp指令名称 属性名=属性值..%>

JSP指令不产生任何直接的输出，它是用来控制jsp翻译引擎如何将当前jsp页面翻译成Servlet的。

jsp指令一共有三种类型：

page指令：指定jsp页面的基本属性，决定了jsp翻译引擎在翻译jsp为servlet过程中的行为

include指令:在当前jsp页面中包含其它jsp页面

taglib指令:引入标签库

a. page指令

<%@ page 属性名=属性值 .. %>

声明当期jsp页面的基本属性。控制jsp翻译引擎如何翻译jsp到servlet。

language="java"	当前JSP使用的开发语言
extends="package.class"	当前jsp翻译成servlet后要继承的类，注意此值必须是一个servlet的子类，一般情况下不要改
import="{package.class | package.*}, ..."	导入需要使用到的包 java.lang.*;javax.servlet.*;javax.servlet.jsp.*;javax.servlet.http.*;
session="true | false"	用来指定当前页面是否使用session，如果设置为true，则翻译过来的servlet中将会有对session对象的引用，于是可以直接在jsp中使用session隐式对象。但是这将导致一旦访问jsp就会调用request.getSession()方法，可能导致不必要的空间浪费。如果确定jsp中不需要session可以设为false
[buffer="none | 8kb | sizekb"]	out隐式对象所使用的缓冲区的大小
[autoFlush="true | false"]	out隐式对象是否自动刷新缓冲区，默认为true，不需要更改
[isThreadSafe="true | false"]	翻译过来的servlet是否实现SingleThreadModel
[errorPage="relative_url"]	如果页面出错，将要跳转到的页面，除了在jsp中使用此属性指定错误页面外也可以在web.xml中配置整个web应用的错误页面，如果两个都设置则jsp中的此属性起作用
[isErrorPage="true | false"]	如果设置此属性为true,翻译过来的servlet中将含有Exception隐式对象,其中封装的就

	是上一个页面中抛出的异常对象
[contentType="mimeType [;charset=characterSet]" | "text/html ; charset=ISO-8859-1"]	和jsp乱码相关的指令,用来指定jsp输出时,设置的Content-Type响应头用来指定浏览器打开的编码
[pageEncoding="characterSet | ISO-8859-1"]	服务器翻译jsp时使用的编码集.如果向防止jsp乱码,应该保证文件的保存编码和jsp翻译成servlet用的编码以及输

b. include指令

<%@include file=""%>

在当前jsp页面中引入其他jsp页面，组合成一个jsp进行输出，如果网页某部分重复的出现很多次，可以独立成一个单独的jsp而在其他jsp中进行引入即可。

c. taglib指令

<%@ taglib uri="" prefix=""%>

在当前页面中引入标签库。

7. 九大隐式对象

在jsp翻译成Servlet的过程中，JSP翻译引擎会自动在service方法中帮我们创建9个对象（需要开启isErrorPage才会有Exception），因此这9个对象可以直接在jsp页面中的java代码内直接访问，这9个对象就称之为jsp的9大隐式对象。

**jsp页面的9大隐式对象是常见面试题之一。

page - 代表当前Servlet
cofing - 代表当前Servlet的ServletConfig对象
request - 代表当前请求的HttpServletRequest对象
response - 代表当前响应的HttpServletResponse对象
application - 代表当前web应用的ServletContext对象
session - 代表当前会话的HttpSession对象
exception - 代表当前页面接收到的Throwable类型异常信息对象
out - 代表当前jsp页面输出流的JspWriter对象
pageContext - 代表当前jsp页面的PageContext类型对象

8. PageContext详解

代表当前jsp页面的执行环境的对象

a. PageContext功能1：作为入口获取其他八大隐式对象

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

b. 作为域对象使用

相关方法

```
setAttribute(Object name,Object value)
getAttribute(Object name)
removeAttribute(Object name)
getAttributeNames()
```

生命周期

当当前jsp被访问开始时，PageContext域生命周期开始，当当前jsp页面访问结束时，PageContext域生命周期结束

作用范围

当前jsp页面范围内共享数据

主要功能

在当前jsp页面内共享数据

c. 可以作为入口操作四大作用域

pageContext本身是一个域，但同时也是一个通用入口，可以通过该对象操作四大作用域中的属性。

相关方法

```
setAttribute(Object name,Object value,int scope)
getAttribute(Object name,int scope)
removeAttribute(Object name,int scope)
getAttributeNames()
```

其中scope可以取值如下，代表从指定域中操作域属性：

```
PageContext.APPLICATION_SCOPE
PageContext.SESSION_SCOPE
PageContext.REQUEST_SCOPE
PageContext.PAGE_SCOPE
```

另外PageContext提供了如下方法，可以无需指定作用域，可以自动按照由小到大搜寻查找四大作用域中指定名称的属性并返回该属性值，只要找到立即返回，如果都找不到，返回null。

```
findAttribute(String name)
```

d. 便捷的请求转发

```
pageContext.forward("path")
```

四大作用域

2019年3月1日 14:50

1. 什么是作用域

   在javaweb开发中，提供了四大作用域，在不同范围和不同生命周期内，共享数据。本质上就是在一个可以在一定时间和一定范围内被共享的对象身上的map中共享数据的过程。

2. 四大作用域

   由大到小排列：

   ServletContext域 - ApplicationScope

   作用范围：

   整个web应用

   生命周期

   web应用加载，立即创建代表当前web应用的ServletContext对象，生命周期开始，服务器关闭或web应用被移出出容器时，随着web应用的销毁，ServletContext对象被销毁，生命周期结束。

   主要功能

   在整个web应用范围内共享数据

   session域 - SessionScope

   作用范围:

   当前会话范围

   生命周期：

   当第一调用request.getSession()时创建session。

   当 超时、手动调用invalidata方法、服务器异外关闭时销毁

   主要功能:

   在会话范围内共享数据

   request域 - RequestScope

   作用范围：

   本次请求范围

   生命周期：

   请求开始，服务器创建代表请求的request对象，生命周期开始，请求结束，request销毁，生命周期结束

   主要功能：

   在请求转发过程中，由上游向下游传递域属性

   pageContext域 - pageScope

   作用范围：

当前jsp页面

生命周期：

当前jsp页面访问开始时创建，当前jsp页面访问结束时销毁

主要功能：

在当前jsp页面范围内共享数据。

且是四大作用域的通用操作对象，额可以通过此对象操作四大作用域。

JSP标签技术

2019年3月1日 15:16

1. JSP标签技术概述

JSP改变了Servlet在java中嵌入html的做法，改为了在html中嵌入java，编写页面确实更加简单了，但是在jsp页面中html中嵌入大量java代码一样会造成jsp页面内容混杂，难以理解和维护。

为了解决这样的问题，sun公司进一步提出了jsp标签技术。简单来说就是希望通过开发一系列的标签来通过标签替代页面中的java代码，最终的目标是jsp页面中不再出现任何一行java代码，都用标签来实现。

2. 常用的JSP标签技术

jsp标签

sun公司提供的标签技术 - 目前用的比较少

el表达式

便捷易用 - 使用非常广泛 - 但只是表达式无法实现复杂功能

jstl标签库

提供了大量强大的标签支持，是目前应用最广泛的标签库

自定义标签

sun提供的标签库开发接口，允许开发人员自己来开发特定功能的标签 - 通常用在框架级别的开发中，普通开发人员很少使用

其他第三方标签

以后了解

EL表达式

2019年3月1日 15:26

1. EL表达式概述

EL 全名为Expression Language，用来替代<%= %>脚本表达式

EL具有获取数据、执行运算、获取常用开发对象、调用java方法这四方面的功能

javaEE目前内置了EL表达式，可以在jsp页面中直接使用

2. EL表达式的基本形式

EL表达式的基本形式为 ${el表达式}

3. EL获取数据

a. 注意事项

el只能获取域中的数据

el只能获取不能设置

el只能获取不能遍历

b. 获取常量

el表达式支持 数字 字符串 布尔类型的常量，可以直接通过el获取

```
<hr><h1>EL获取常量</h1>
<%= 123 %>
${123 }
<%= 123.321 %>
${123.321 }
<%= "abc" %>
${"abc" }
<%= true %>
${true }
<%= false %>
${false }
```

c. 获取变量 - 自动搜寻域

当通过el获取变量时，el会自动在四大作用域中按照由小到大的

顺序搜寻指定名称的域属性，一旦获取直接返回，如果四大作用域中都没有找到，则返回一个空字符串。

```
<hr><h1>EL获取变量</h1>
<%
    application.setAttribute("name", "朴乾");
    session.setAttribute("name", "鑫三胖");
    request.setAttribute("name", "曹二胖");
    pageContext.setAttribute("name", "李一瘦");
%>
<%=pageContext.findAttribute("name") %>
${name}
```

d. 获取变量 - 获取指定域中的属性

在el中可以直接使用如下四个内置对象来操作指定域

applicationScope - 操作ServletContext域

sessionScope - 操作session域

requestScope - 操作request域

pageScope - 操作PageContext域

```
<hr><h1>EL获取指定域中的变量</h1>
${pageScope.name }
${requestScope.name }
${sessionScope.name }
${applicationScope.name }
```

e. 获取数组中的数据

```
<hr><h1>EL获取数组中的变量</h1>
<%
    String names [] = {"张三丰","张无忌","张翠山","谷丰硕"};
    pageContext.setAttribute("names", names);
%>
${names[3] }
```

f. 获取集合中的数据

```
<hr><h1>EL获取集合中的数据</h1>
<%
    List<String> list = new ArrayList<String>();
```

```
        list.add("周芷若");
        list.add("赵敏");
        list.add("殷素素");
        list.add("小昭");
        list.add("灭绝师太");
        pageContext.setAttribute("list", list);
%>
${list[3] }
```

g. 获取Map中的数据

```
<hr><h1>EL获取Map中的数据</h1>
<%
        Map<String,String> map = new HashMap<String,String>();
        map.put("name","韦小宝");
        map.put("job","爵爷");
        map.put("master","陈近南");
        map.put("gender","男");
        map.put("wife.first","双儿");
        pageContext.setAttribute("map", map);
        pageContext.setAttribute("job", "master");
 %>
 ${map["master"] }
 ${map[job] }
 ${map.gender }
 ${map.name }
 ${map["wife.first"] }
 ${map.wife.first }
```

h. 获取javabean的属性

```
<hr><h1>EL获取JavaBean中的属性</h1>
<%
        Person p = new Person("zs",19,"bj");
        pageContext.setAttribute("p", p);
%>
 ${p.namex }
 ${p.age }
 ${p.addr }
 ${p["age"] }
```

4. EL执行运算

a. 算数运算

```
<hr><h1>EL执行运算 - 算术运算</h1>
${3 + 2}
${3 - 2}
${3 * 2}
${3 / 2}
${3 % 2 }
${"3" + 2 }
${"3" + "2" }
<%-- ${"a" + 2 } --%>
```

b. 关系运算

```
<hr><h1>EL执行运算 - 关系运算</h1>
${3 == 2 }
${3 > 2 }
${3 >= 2 }
${3 < 2 }
${3 <= 2 }
${3 != 2 }
```

c. 逻辑运算

```
<hr><h1>EL执行运算 - 逻辑运算</h1>
${true and false }
${true && false }
${true or false }
${true || false }
${not (3>2) }
${! (3>2) }
```

d. Empty运算

```
<hr><h1>EL执行运算 - empty运算</h1>
<h1>Empty运算规则：如果对象为null 字符串为空 集合数组没有任何元素 empty操作都会返回true，否则false</h1>
<%
    String country = "China";
    pageContext.setAttribute("country", country);
    String province = "";
```

```
        pageContext.setAttribute("province",province);
        List<String> listx = new ArrayList<String>();
        pageContext.setAttribute("listx",listx);
 %>
${empty country}
${empty province}
${empty listx}
```

e. 三元表达式

```
<hr><h1>EL执行运算 - 三元表达式</h1>
<%
        Map<String,String> map = new HashMap<String,String>();
        map.put("name", "杨过");
        map.put("武功", "黯然销魂掌");
        map.put("wife", "姑姑");
        map.put("宠物", "雕");
        pageContext.setAttribute("map", map);
 %>
 ${empty map.wife ? "小龙女" : map.wife }
```

5. EL获取常用开发对象 - 11个内置对象

EL内置了11个内置对象，不需要提前存入域就可以直接在EL中使用。注意这和jsp的9打隐式对象没有任何关系。

a. 代表作用域的EL内置对象

pageScope	代表PageContext域
requestScope	代表Request域
sessionScope	代表Session域
ApplicationScope	代表ServletContext域

b. 代表请求参数的隐式对象

param	代表当前请求中全体请求参数组成的Map<String,String>
paramValues	代表当前请求中全体请求参数组成的Map<String,String[]>

c. 代表web应用的初始化参数的EL内置对象

initParam	代表当前web应用初始化参数的内置对象

d. 代表请求头的EL内置对象

header	代表请求中所有请求头组成的Map<String,String>的内置对象
headerValues	代表请求中所有请求头组成的Map<String,String[]>的内置对象

e. 代表请求中所有cookie组成的Map的内置对象

cookie	代表请求中所有cookie组成的Map<String,Cookie>

f. 代表当前页面的PageContext对象

pageContext	代表当前页面的PageContext对象，通过它可以获取所有jsp9大隐式对象

6. EL调用java方法

略

JSTL标签库

2019年3月2日 9:27

1. JSTL标签库概述

JSTL全称为JavaServerPages Standard Tag Library。

由JCP（Java Community Process）指定标准。

是提供给 Java Web 开发人员一个标准通用的标签函数库。

可以和 EL 配合来取代传统直接在页面上嵌入 Java 程序（Scripting）的做法，以提高程序可读性、维护性和方便性。

2. 在jsp引入jstl

从javaee5开始已经内置了jstl标签库，不需要导入任何开发包就可以直接在jsp页面中通过<%@taglib uri="" prefix=""%>来引入jstl标签库。

3. JSTL标签库的子库

核心标签库 core - c

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

国际化标签 fmt

数据库标签 sql

XML标签 xml

JSTL函数(EL函数)

4. JSTL标签库中常用标签

a. <c:set>

设置/修改域中属性。

设置/修改域中的java.util.Map类型的属性。

设置/修改域中的JavaBean类型的对象的属性。

```
<hr><h1>c:set 设置/修改域属性 </h1>
<c:set scope="request" var="name" value="zs"></c:set>
${name}
<c:set scope="request" var="name" value="ls"></c:set>
${name}
```

```
<hr><h1>c:set 设置/修改域中Map属性 </h1>
<%
    Map<String,String> map = new HashMap<String,String>();
    pageContext.setAttribute("map", map);
 %>
<c:set target="${map }" property="addr" value="bj"></c:set>
${map.addr }
<c:set target="${map }" property="addr" value="sh"></c:set>
${map.addr }

<hr><h1>c:set 设置/修改域中javabean属性 </h1>
<%
    Person p = new Person("ww",19,"gz");
    pageContext.setAttribute("p", p);
 %>
<c:set target="${p }" property="age" value="20"></c:set>
${p.age}
```

b. <c:remove>

从所有域或指定域中删除指定属性。

```
<hr><h1>c:remove 删除所有域或指定域中的属性 </h1>
<c:set var="namex" value="zl1" scope="page"></c:set>
<c:set var="namex" value="zl2" scope="request"></c:set>
<c:set var="namex" value="zl2" scope="session"></c:set>
<c:set var="namex" value="zl2" scope="application"></c:set>
<c:set var="agex" value="18" scope="page"></c:set>
${pageScope.namex }
${requestScope.namex }
${sessionScope.namex }
${applicationScope.namex }
${agex }
<c:remove var="namex" scope="page"/>
${pageScope.namex }
${requestScope.namex }
${sessionScope.namex }
${applicationScope.namex }
${agex }
<c:remove var="namex"/>
${pageScope.namex }
${requestScope.namex }
```

```
${sessionScope.namex }
${applicationScope.namex }
${agex }
```

c. <c:catch>

捕获异常

```
<hr><h1>c:catch 捕获异常</h1>
<c:catch var="e">
<%
    String str = null;
    str.toUpperCase();
 %>
 </c:catch>
 ${e}
```

d. <c:if>

实现判断

```
<hr><h1>c:if 实现判断</h1>
<%
    pageContext.setAttribute("num", 18);
 %>
<c:if test="${num > 20}" scope="page" var="flag">
    yes~
</c:if>
<c:if test="${num <= 20}">
    no~
</c:if>
${flag}
```

e. <c:choose><c:when><c:otherwise>

实现多重判断

```
<hr><h1>c:choose 实现多重判断</h1>
<%
    pageContext.setAttribute("num", 18);
 %>
<c:choose>
    <c:when test="${num<10 }">小于10</c:when>
    <c:when test="${num<100 }">大于10小于100</c:when>
    <c:when test="${num<1000 }">大于100小于1000</c:when>
    <c:otherwise>大于1000</c:otherwise>
```

```
</c:choose>
```

f. <c:foreach>

实现循环、实现遍历

```
<hr><h1>c:foreach 实现循环遍历</h1>
<%
	List<String> list = new ArrayList<String>();
	list.add("aa");
	list.add("bb");
	list.add("cc");
	list.add("dd");
	list.add("ee");
	pageContext.setAttribute("list", list);
%>
<c:forEach items="${list }" var="x">
	${x }
</c:forEach>
<c:forEach begin="0" end="100" step="2" var="i">
	${i }
</c:forEach>

<hr><h1>c:foreach案例：遍历10到100的偶数，如果数字所在的位置是3的倍数，显示成红色
</h1>
<c:forEach begin="10" end="100" step="2" var="i" varStatus="stat">
	<c:if test="${stat.count % 3 == 0 }">
		<font color="red">${i }</font>
	</c:if>
	<c:if test="${stat.count % 3 != 0 }">
		<font color="blue">${i }</font>
	</c:if>
</c:forEach>
```

g. <c:fortoken>

根据给定分隔符切割指定字符串，将得到的字符串数组进行遍历

```
<hr><h1>c:fortoken </h1>
<c:forTokens items="www.tedu.cn" delims="." var="str">
	${str}<br>
</c:forTokens>
```

改造EasyMall - 利用JSP、EL、JSTL来开发页面

2019年3月2日 10:46

1. 标签技术改造jsp页面中的java代码

<%=request.getContextPath() %>	${pageContext.request.contextPath}
a. <%= request.getAttribute("msg") == null ? "" : request.getAttribute("msg") %>	${requestScope.msg}
a. <%=request.getParameter("username") == null ? "" : request.getParameter("username")%> ${requestScope.username}	${param.username}

<%
Cookie usernameCookie = null;
Cookie[] cs = request.getCookies();
if(cs != null){
for(Cookie c : cs){
if("username".equals(c.getName())){
usernameCookie = c;
}
}
}
String username = "";
if(usernameCookie != null){
username =
URLDecoder.decode(usernameCookie.getValue(),
"utf-8");
}
%>
	${cookie["username"].value}

<%=usernameCookie == null?"":"checked='checked'" %>	<c:if test="${empty cookie['username'] }" > checked='checked' </c:if>