2017、2、23

EL表达式快速入门

EL表达式用来取数据。

基本的样式：${obj}

（他等同于调用了pageContext.findAttribute(obj);）

以下列出使用的例子：

例一：

<%

String name = "majin";

request.setAttribute("name", name);

%>

${name}

<%--相当于调用了pageContext.findAttribute(name) --%>

<%--返回结果找到了返回结果，找不到返回"" 注意不是null，这就省了我们进行判断--%>

例二：

<%

PersonBean p = new PersonBean();

p.setName("majin");

request.setAttribute("person", p);

%>

${person.name}

<%-- 这句话看似简单但是 包含了两个调用过程--%>

<%-- pageContext.getAttribute("person") --%>

<%-- 使用反射来获取person中的name属性--%>

例三：

//当数据存放在list的集合中

<%

List<PersonBean> list = new ArrayList();

list.add(new PersonBean("aaa"));

list.add(new PersonBean("bbb"));

list.add(new PersonBean("ccc"));

request.setAttribute("list", list);

%>

${list[1].name}

//目前我们还无法单靠el表达式来迭代，还需要jstl

例四：

<%

Map map = new HashMap();

map.put("a", new PersonBean("a\_map"));

map.put("b", new PersonBean("b\_map"));

map.put("c", new PersonBean("c\_map"));

map.put("11", new PersonBean("111\_map"));

request.setAttribute("map", map);

%>

${map.a.name}

<br/>

<%-- ${map.11.name } --%>

${map['11'].name }

<%-- 这就说明，key、id等最好不要使用数字来表示 --%>

<%-- 取数据一般使用.号，当.号不能使用时，可以使用[]--%>

例五：

//比较重要的使用

${pageContext.request.contextPath}

//用来获取web应用的根目录

JSTL的快速入门

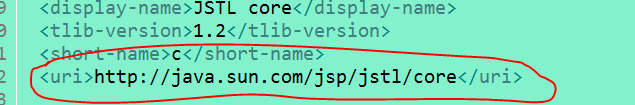
jsp的标准标签库

使用：

1. 导入jar包，jstl.jar和standard.jar这两个jar包(我的me不需要导...)
2. 使用taglib指令引入

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

这个uri怎么来的那。打开standard.jar中的META-INF中的c.tld,可以找到



c.tld文件中定义的所有标签都绑定到这个uri上面去了（和schema相似）

prefix的值最好取 文件的名字

1. 使用例子：

例一：

<%-- 使用forEach标签进行list集合的迭代 --%>

<%

List list = new ArrayList();

list.add(new PersonBean("a\_list"));

list.add(new PersonBean("b\_list"));

list.add(new PersonBean("c\_list"));

list.add(new PersonBean("d\_list"));

request.setAttribute("list", list);

%>

<c:forEach items="${list}" var="person">

${person.name}<br />

</c:forEach>

例二：

<%-- 使用forEach标签进行map集合的迭代 --%>

<%

Map map = new HashMap();

map.put("a", new PersonBean("a\_map"));

map.put("b", new PersonBean("b\_map"));

map.put("c", new PersonBean("c\_map"));

request.setAttribute("map", map);

%>

<c:forEach items="${map}" var="entry">

<%-- 这里的迭代方式是使用的map的entrySet进行迭代的--%>

${entry.key} : ${entry.value.name}

<br />

</c:forEach>

例三

<%--使用if标签进行判断 ，用户是否登录--%>

<c:if test="${username != null}">

欢饮您,${username}

</c:if>

<c:if test="${username == null}">

<form action="">

用户名：<input type="text" name="username"/>

密码:<input type="password" name="password" />

</form>

</c:if>

自定义标签

自定义标签的步骤：

1. 写一个实现tag接口的java类（标签处理器类），将jsp中的java代码移到doStartTag（）中。

public class viewIPTag extends TagSupport {

@Override

public int doStartTag() throws JspException {

HttpServletRequest request = (HttpServletRequest) pageContext.getRequest();

JspWriter out = pageContext.getOut();

String ip = request.getRemoteAddr();

try {

out.print(ip);

} catch (IOException e) {

throw new RuntimeException(e);

}

return super.doStartTag();

}

}

1. 在tld文件中，将标签处理器类描述成标签（抄tomcat的examples中tld的文件格式，把标签绑定到uri上面去）。

<?xml version="1.0" encoding="UTF-8" ?>

<taglib xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-jsptaglibrary\_2\_0.xsd"

version="2.0">

<description>A tag library exercising SimpleTag handlers.</description>

<tlib-version>1.0</tlib-version>

<short-name>test</short-name>

<uri>http://www.test.cn</uri>

<tag>

<name>viewIP</name>

<tag-class>com.majin.web.tag.viewIPTag</tag-class>

<body-content>empty</body-content> **不是空标签时使用JSP**

</tag>

</taglib>

1. 在jsp文件中使用标签（导入和使用）。

<%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>

<%@ taglib uri="http://www.test.cn" prefix="test"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>

<title>使用自定义标签显示当前的ip地址</title>

</head>

<body>

您的IP地址为:

<%--

<%

String ip = request.getRemoteAddr();

out.print(ip);

%>

--%>

<test:viewIP/>

</body>

</html>

自定义标签的实现原理：

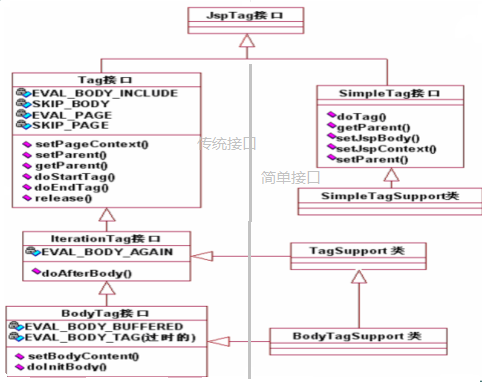
Tag接口中定义有6个方法

setPageContext()、setParent()、doStartTag()、doEndTag()、release()和getParent()

最重要说的就是setPageContext(PageContext page)，这个方法把jsp页面的pageContext对象传递给标签处理器类，而且pagecontext对象封装了其他八大隐式对象的引用。那么，jsp中要干的事都可以在我们的java类中干了。

自定义标签的其他的应用：

1. 控制一段内容显示
2. 控制整个页面显示
3. 控制某段内容循环显示
4. 修改某段内容



传统接口（jsp2.0之前，因为有些企业的框架是使用jsp2.0以前的技术开发的）

1. doStartTag() 返回EVAL\_BODY\_INCLUDE标签体显示,SKIP\_BODY标签体不显示
2. doEndTag() 返回EVAL\_PAGE后续页面继续显示，SKIP\_PAGE不显示
3. （使用IterationTag接口）doAfterBody()在标签体结束之后、结束标签开始之前调用，返回EVAL\_BODY\_AGAIN标签体再次执行，SKIP\_BODY标签体不再执行
4. （使用BodyTag接口）
   1. 在doStartTag中，返回EVAL\_BODY\_BUFFERED
   2. 在doEndTag中，得到BodyContext，修改后输出

传统接口流程:

Jsp被访问（访问到自定义标签）-->jsp遇到自定义标签<test:viewIP>-->根据test和taglib指令找到uri-->根据uri、viewIP和tld文件找到java类-->实例化这个java类-->依次调用setPageContext、setParent、doStartTag、doEndTag、release方法-->结束后继续执行jsp后面的代码。(注意：标签处理器的对象并不会立即销毁、和servlet生命周期一样)

简单接口（前面讲的都是传统接口）：

1. getJspBody().invoke(null)或getJspBody().invoke(this.getJspContext().getOut())，将内容 输出给浏览器。当然，你不这么做的话，不会向浏览器输出标签体
2. 控制后续页面不显示，只需要抛出SkipPageException即可。
3. 重复显示：在doTag()方法中，使用for循环重复调用 invoke()
4. 改变显示内容：使用invoke将内容先输出到带有存缓的writer中，在拿到存缓后，修改内容即可。

简单接口的流程：

Jsp被访问（访问到自定义标签）-->....-->实例化java类-->调用setJspContext()、setJspBody()、setParent()、doTag()方法-->继续jsp的后续代码。（注意：标签处理器的对象会在doTag()方法调用结束后立即变成垃圾，等待jvm的回收）

同时，还要注意：

简单接口的tld中定义标签内容 与传统的方式不一样。

简单接口（scritpless/empty） 传统接口（JSP/empty）

自定义标签中定义属性：

只需要两步，

1. 标签处理器的属性设置setter方法（这个属性必须和标签的属性名字相同）

private int count = 0;

public void setCount(int count) {

this.count = count;

} ------------------------------正确

pivate int count1= 0 ;

public void setCount1(int count) {

This.count1 = count1;

}--------------------------------错误

1. 在tld文件中对属性进行描述

<tag>

<name>sDemo5</name>

<tag-class>cn.test.web.simpleTag.Demo5</tag-class>

<body-content>scriptless</body-content>

<attribute>

<name>count</name>

<required>true</required>

<rtexprvalue>true</rtexprvalue>

</attribute>

</tag>

用例：

1. 防盗链
2. If标签
3. If-else标签
4. forEach标签
5. forEach2标签
6. Html转义标签
7. 打包

1.防盗链（使用到request中的referer）

JSTL

out标签 value、default、escapeXML

案例：

foreach标签 items、var、begin、end、step、varStatus

案例：1.迭代集合

1. 输出1~9

<c:foreach begin="1" end="9" var="i">

${i}

</c:foreach>

1. 实现table的间色（使用varstatus属性）（注意：html的版本会影响tr是不是会有huvor属性）

if标签 test、var、scope（test为条件，var和scope仍是和域存储有关）

choice标签 when标签 otherwise标签

forTokens标签 items、delims、begin、end、step、var、varstatus

案例：

实现字符串的分割

<%

request.setAttribute("data","a,c,v,b");

%>

<c:forTokens items="${data}" delims="," var="sub">

${sub}

</c:forTokens>

catch标签 var （捕捉标签体中的异常，并把异常作为对象 以var值为关键字保存到域中）

param标签 name、value（配合url、redirect标签的使用，相当于在url地址后加上参数）

url标签 var、scope、value、context

案例：构造url、实现url的重写（回话标志号以参数的形式跟在url地址后面）

1. <a href='<c:url value="/index.jsp "/>' >点点</a>

（这样一来，a标签的href变成 /Day11/index.jsp ，比我们使用${pageContext.request.contextPath}再和/index.jsp合并要简单的多）

2. <c:url var="url" value="/Day11/index.jsp" />

<a href="${url}">点点</a>

1. 中文字符的url编码（不陌生，百度的搜索引擎）

<c:url var="url" value="index.jsp">

<c:param name="name" value="中国"></c:param>

</c:url>

<a href="${url}">点点</a>

redirect标签 url、context （重定向）

remove标签 var、scope （移除某个域中的属性）

set标签 var、value、target、property、scope

案例：1.设置各个域中的属性

<c:set var="" value="xx" scope="page|session|application|context"/>

1. 设置map集合中的属性

<%

Map map=new HashMap();

request.setAttribute("map",map);

%>

<c:set target="${map}" property="key" value="value" />

3.设置javaBean的属性

<%

Person p =new Person();

request.setAttribute("p",p);

%>

<c:set target="${p}" property="name" value="value"/>

el表达式

1. 获取数据
2. 执行运算

算数运算、关系运算、逻辑运算

empty运算---判断集合是null 或者 空

格式:${empty(list)}

二元运算符

格式:${user==null ? "" :user.name}

(注意el表达式不支持字符串的链接，即${"您好，"+username}不支持)

二元运算符还可以做 数据的回显

如：

<input type="radio" name="gender" value="male" ${gender=='male' ? 'checked':''}>男

<input type="radio" name="gender" value="female" ${gender=='female'?'checked':''}>女

1. 获取web开发的常用对象

${name}这el表达式执行时，首先会查找是不是有name的隐式对象，没有才会调用pageContext.findAttribute(name)来查找，还找不到返回""

el表达式有 11个隐式对象

a)pageContext 取到jsp中的pageContext对象

b)pageScope 取到PageContext所有属性的map集合

c)sessionScope

d)requestScoope

e)applicationScope

<%

pageContext.setAttribute("name","mm");

%>

${name}

${pageScope.name}

f)param 所有请求参数的map集合

用做数据的回显

g)paramValues 对应一个name有多个值的情况，返回的是String[]

http://localhost:8080/Day12/6.jsp?name=aaa&&name=bbb

${paramValues.name} //这打得到的是数组

${paramValues.name[1]}

h)header 得到所有请求头的map集合

i)headerValues

j)cookies 保存了所有cookie的map对象

${cookies}//得到集合

${cookies.JSESSIONID} //得到session对象

${cookies.JSESSIONID.name}//得到session对象的名字

${cookies.JSESSIONID.value}//session的值

k)initParam web应用的初始化参数的map集合

web.xml下

<web-app>

<context-param>

<name></name>

<value></value>

</context-param>

</web-app>

前六个对象比较重要

EL函数

el表达式可以直接调用java的函数，注意：

1. 只能调用静态的函数
2. 在tld文件中将函数描述成el函数

如：

<function>

<name>filter</name> el函数的名字

<function-class>cn.web.utils.HtmlFilter</function-class> 函数的类的位置

<function-signature>java.lang.String filter( java.lang.String )</function-signature> 函数的签名

</function>

调用的格式：

<%@ taglib uri="http://www.test.cn" prefix="d" %>

....

${d:filter("<a href='http://www.baidu.com'>点点</a>")}

sun公司定义的el函数库（略）

国际化（internationalization 简称i18n）

哪个地区的人来，显示哪个地区的显示风格。

合格的国际化软件：

1. 固定文本，如菜单、导航等，错误信息等 需要国家化
2. 动态数据，如日期、货币等 国际化

固定文本的国际化：

文本信息保存到properties中，一组properties成为一个资源包

一组资源包拥有相同的基名，如：message.properties、message\_zh.properties和message\_en.properties等

资源包中只能保存ascii型，不能保存中文（使用native2ascii命令，可将一切文本转换成ascii）

javaAPI中，ResourceBundle类的getBundle(baseName，locale)可以根据来访者自动获取相应的资源包进行显示。而后，使用getString(key)从资源包中拿取字符串。

国际化相关的类

Locale类、DateFormat类、NumberFormat类、MessageFormat类

2017/3/3

preparedStatement与Statement的区别：

1. 两个均是接口，PreparedStatement 是Statement的孩子
2. PreparedStatement可以预防SQL注入(使用占位符 和 转义)
3. PreparedStatement 对sql语句进行预编译，可以减缓数据库的压力

java程序 --> (预编译) class文件 --> jvm执行

sql语句 -->(预编译) 关系代数表达式 --> DBMS执行

2017/3/16

元数据：数据的定义信息

数据库元数据 -- DataBaseMetaData

参数元数据 -- ParameterMetaData

结果元数据 -- ResultSetMetaData

woc,爆炸! TM卖批！

java.lang.IllegalArgumentException: Can not set int field cn.majin.domain.Account.id to java.math.BigDecimal

在Account类中id声明为int类型，

在Account表中id生命为int/number(4,2)/number(1)/Integer 都不行

百度出来，蛇皮！

1）如果不指定number的长度，或指定长度n>18

id number not null，转换为pojo类时，为java.math.BigDecimal类型

2）如果number的长度在10 <= n <= 18

id number(n) not null，转换为pojo类时，为java.lang.Long类型

3）如果number的长度在1 <= n <= 9

id number(n) not null，转换为pojo类时，为java.lang.Integer类型

阿萨