ORACLE WORKFORCE DEVELOPMENT PROGRAM

Java企业级应用开发

4

第4章：JSTL与EL表达式

CSG
CAMPUS SOLUTION GROUP

本章内容

- 第1节 EL表达式
- 第2节 EL表达式对象
- 第3节 JSTL标签库概述
- 第4节 core标签库
- 第5节 function函数库

本章目标

- 理解EL表达式特性和优点。
- 掌握EL表达式在Jsp页面中的应用过程。
- 掌握EL表达式作用域与隐式对象。
- 了解JSTL标签库的分类。
- 熟练应用core标签库。
- 了解掌握function函数库。

第1节 EL表达式

知识点预览

#	知识点	难点	重点	应用	说明
1	EL表达式概述				介绍EL表达式的用途和特点
2	获取作用域数据	√	√	√	掌握EL表达式作用域
3	使用运算符		√	√	EL表达式运算符的应用

EL表达式及其特性

- EL全名为Expression Language，用于**实现Jsp页面无脚本环**境。用于替代表达式脚本，也是代表一个值，可以用于结果的显示。另外，方便获得指定变量的值的同时，还可以自动类型转换。

EL主要作用

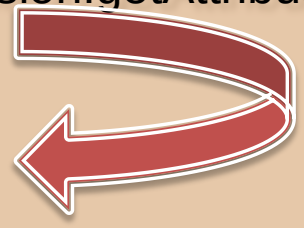
- 1、获取数据
- 2、执行运算
- 3、获取web开发常用对象
- 4、调用Java方法

EL表达式概述

EL表达式的基本语法格式

- EL表达式总是放在标记"${"和"}"之间。
  - EL表达式一般由两部分构成，如：${customer.name }，其中“.”称为点操作符，符号左边可以是对象，也可以是EL隐式对象，右边可以是对象属性，也可以是映射键。
  - 如果操作数组，则使用“.”操作就不能进行有效的操作，这时可以使用“[]”，如${list[0] }。但是并不代表“[]”只能操出作数组。“.”操作符可以操作的对象，都可以使用“[]”来进行操作，“[]”中应该含有“””” 。如${customer[“name”] }，当然操作数组可以写成${list[“0”] }。
- 如图：

```
<%
        Customer customer = (Customer)session.getAttribute(“customer”);
        String name = customer.getName();
%>
${customer.name}  或
${sessionScope.customer[“name”]}
```



1、获取数据

- 使用EL表达式获取数据语法："${标识符}"
- EL表达式语句在执行时，会调用pageContext.findAttribute方法，
- 用标识符为关键字，分别从page、request、 session、application四个域中查找相应的对象，找到则返回相应对象，找不到则返回”“（注意，不是null，而是空字符串）。

testel.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%@page import="gacl.javabean.study.Person"%>
<!DOCTYPE HTML>
<html>
 <head>
  <title>el表达式获取数据</title>
 </head>  <body>
  <%     request.setAttribute("name","孤傲苍狼");%>
     <%--${name}等同于pageContext.findAttribute("name") --%>
     使用EL表达式获取数据：${name}
  <hr>
  <!-- 在jsp页面中，使用el表达式可以获取bean的属性 -->
  <%
     Person p = new Person();
     p.setAge(12);
     request.setAttribute("person",p);
  %>
     使用el表达式可以获取bean的属性：${person.age}
  <hr>
  <!-- 在jsp页面中，使用el表达式可以获取bean中的。。。。。。。。的属性 -->
  <%
     Person person = new Person();
     Address address = new Address();
     person.setAddress(address);

     request.setAttribute("person",person);
  %>
    ${person.address.name}
   <hr>

</body>
```

Testlist.jsp

```
 <%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%@page import="gacl.javabean.study.Person"%>
<!DOCTYPE HTML>
<html>
  <head>
    <title>el表达式获取数据</title>
  </head>  <body>
      <!-- 在jsp页面中，使用el表达式获取list集合中指定位置的数据 -->
         <%
            Person p1 = new Person();
            p1.setName("孤傲苍狼");

            Person p2 = new Person();
            p2.setName("白虎神皇");

            List<Person> list = new ArrayList<Person>();
            list.add(p1);
            list.add(p2);
            request.setAttribute("list",list);
         %>

         <!-- 取list指定位置的数据 -->
          ${list[0].name}
         ${list[1].name}
         <!-- 迭代List集合 -->
</body>
</html>
```

2、执行运算

- 语法：${运算表达式}
- EL表达式支持如下运算符

  - 关系运算符
  - 逻辑运算符
  - empty运算符
  - 二元表达式
  - [ ] 和 . 号运算符

使用运算符

EL算术运算符

- 算术运算符规则举例

EL表达式算术运算符			
EL算术运算符	说明	范例	结果
+	加	${1+5 }	6
-	减	${10-1 }	9
*	乘	${2*5 }	10
/或div	除	${10/2 }或${10 div 2 }	5
%或mod	取余	${8%3}或${10 mod 2 }	2

- EL表达式中可以直接进行算术运算。

使用运算符

EL关系运算符

◆关系运算符规则

EL表达式关系运算符			
EL关系运算符	说明	范例	结果
==或eq	等于	${1==5 }或**${1 eq 5 }**	false
!=或**ne**	不等于	${1!=5 }或${1 ne 5 }	true
<或lt	小于	${1<5 }或${1 lt 5 }	true
>或**gt**	大于	${1>5 }或${1 gt 5 }	false
<=或le	小于等于	${1<=5 }或${1 le 5 }	true
>=或ge	大于等于	${1>=5 }或${1 ge 5 }	fasle

◆EL表达式中可以直接进行关系运算，结果为boolean类型。

使用运算符

EL逻辑运算符

- 逻辑运算符规则

EL表达式逻辑运算符			
EL逻辑运算符	说明	范例	结果
&&或and	交集	${1==5 && 2>3 }或${1==5 and 2>3 }	false
||或or	并集	${1==5 \|\| 3>2 }或${1==5 or 3>2 }	true
!或not	非运算	${!(1==5)}或${not(1==5)}	true

- EL表达式中可以直接进行逻辑运算，结果为boolean类型。

使用运算符

EL其他运算符

- 运算符规则

EL表达式其他运算符			
EL其他运算符	说明	范例	结果
empty	判断是否为空	${empty value}	true/false
条件运算符	条件成立取B不成立取C	${A ? B : C }	B/C
()	改变执行优先权	${2+(5*2) }	12

- EL表达式如表中的运算符在业务运算时也很有用。

testelbds.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@page import="me.gacl.domain.User"%>
<!DOCTYPE HTML>
<html>
  <head>
    <title>el表达式运算符</title>
  </head>

  <body>
  <h3>el表达式进行四则运算：</h3>
      加法运算：${365+24}<br/>
      减法运算：${365-24}<br/>
      乘法运算：${365*24}<br/>
      除法运算：${365/24}<br/>

    <h3>el表达式进行关系运算：</h3>
    <%--${user == null}和 ${user eq null}两种写法等价--%>
      ${user == null}<br/>
      ${user eq null}<br/>

      <h3>el表达式使用empty运算符检查对象是否为null(空)</h3>
</html>
```

el表达式进行四则运算：

加法运算：389
减法运算：341
乘法运算：8760
除法运算：15.208333333333334

el表达式进行关系运算：

false
false

Testeleq.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@page import="me.gacl.domain.User"%>
<!DOCTYPE HTML>
<html>
  <head>
    <title>el表达式运算符</title>
  </head>
<body>
    <% List<String> list = new ArrayList<String>();
       list.add("gacl");
       list.add("xdp");
       request.setAttribute("list",list);
    %>
    <%--使用empty运算符检查对象是否为null(空)--%>
    <c:if test="${!empty(list)}">
       <c:forEach var="str" items="${list}">
          ${str}<br/>
       </c:forEach>
    </c:if>
    <br/>
    <%
       List<String> emptyList = null;
    %>
    <%--使用empty运算符检查对象是否为null(空)--%>
    <c:if test="${empty(emptyList)}">
       对不起，没有您想看的数据
    </c:if>
    <br/>
    <h3>EL表达式中使用二元表达式</h3>
    <%
       session.setAttribute("user",new User("孤傲苍狼"));
    %>
    ${user==null? "对不起，您没有登陆" : user.username}
    <br/>
    <h3>EL表达式数据回显</h3>
    <%  User user = new User();
       user.setGender("male");
       request.setAttribute("user",user);
    %>
      <input type="radio" name="gender" value="male" ${user.gender=='male'?'checked':''}>男
      <input type="radio" name="gender" value="female" ${user.gender=='female'?'checked':''}>女
  <br/>65  </body>
</html>
```

3、获得web开发常用对象

- EL表达式语言中定义了11个隐含对象，使用这些隐含对象可以很方便地获取web开发中的一些常见对象，并读取这些对象的数据。
- 语法：${隐式对象名称}：获得对象的引用

序号	隐含对象名称	描　　述
1	pageContext	对应于JSP页面中的pageContext对象（注意：取的是pageContext对象。）
2	pageScope	代表page域中用于保存属性的Map对象
3	requestScope	代表request域中用于保存属性的Map对象
4	sessionScope	代表session域中用于保存属性的Map对象
5	applicationScope	代表application域中用于保存属性的Map对象
6	param	表示一个保存了所有请求参数的Map对象
7	paramValues	表示一个保存了所有请求参数的Map对象，它对于某个请求参数，返回的是一个string[]
8	header	表示一个保存了所有http请求头字段的Map对象，注意：如果头里面有“-”，例Accept-Encoding，则要header[“Accept-Encoding”]
9	headerValues	表示一个保存了所有http请求头字段的Map对象，它对于某个请求参数，返回的是一个string[]数组。注意：如果头面有“-”，例Accept-Encoding，则要headerValues[“Accept-Encoding”]
10	cookie	表示一个保存了所有cookie的Map对象
11	initParam	表示一个保存了所有web应用初始化参数的map对象

从对象中获取数据

EL表达式从对象中获取数据

- 通过EL表达式从对象中获取数据
  - EL可以操作变量，数组、映射、表达式的运算以及逻辑运算。还可以操作对象。
  - 对象可能来自某一个EL隐式作用域对象。在获得对象时可以采用映射的方式，进而获得对象中的属性值时，可以通过“.”运算符实现逐层定位的方式。

```
${sessionScope.customer.age}
或
${sessionScope.customer[“age”]}
```

  - 通过EL表达式获取对象的属性值相当于如下Java服务程序脚码业务。

```
<%
    Customer customer = (Customer)session.getAttribute(“customer”);
    int age = customer.getAge();
%>
${sessionScope.customer.age}  或
${sessionScope.customer[“age”]}
```

提高EL效率

提高EL效率

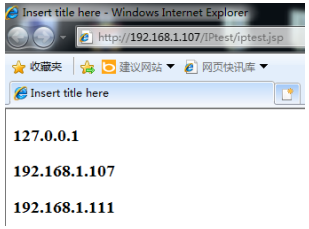
- EL表达式取得变量的值的方式。
  - ${customer }表达的意思是取出某一范围的名字为”customer”的变量的值。默认查找顺序是按照pageScope->requestScope-sessionScope->applicationScope的顺序，并且从查找到的位置回传，不再继续。
- 提高EL表达式取得变量值的效率
  - 采用${XXXScope.customer }在指定的作用域范围获得以customer为key的属性值对象，从而减少了逐层寻找匹配的麻烦。
  - 如果customer是一个引用型对象，获得该对象的属性值可以采用如下语法格式：
    - 例如：${XXXScope.customer.attributeName }。

testelobject.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<!DOCTYPE HTML>
<html>
  <head>
    <title>el隐式对象</title>
  </head>

  <body>
   <br/>----------1、pageContext对象：获取JSP页面中的pageContext对象----------<br/>
       ${pageContext}
    <br/>----------2、pageScope对象：从page域(pageScope)中查找数据----------<br/>
    <%
       pageContext.setAttribute("name","孤傲苍狼");  //map
    %>
       ${pageScope.name}
      <br/>----------3、requestScope对象：从request域(requestScope)中获取数据----------<br/>
      <%
         request.setAttribute("name","白虎神皇");  //map
      %>
         ${requestScope.name}
      <br/>----------4、sessionScope对象：从session域(sessionScope)中获取数据----------<br/>
      <%
         session.setAttribute("user","xdp");  //map
      %>
         ${sessionScope.user}
      <br/>----------5、applicationScope对象：从application域(applicationScope)中获取数据----------<br/>
      <%
         application.setAttribute("user","gacl");  //map
      %>
         ${applicationScope.user}

  </body>
</html>
```

练习

- 掌握通过EL隐式作用域对象获取数据，并通过pageContext获取访问请求数据
- 实现步骤
- 可使用list存放访问的IP地址，将list设置为page范围属性，使用EL表达式输出list中的前三个IP。



访问请求数据

Jsp与EL隐式公共对象

- Jsp和EL有一个公共对象:pageContext,EL通过pageContext来访问Jsp中其他的隐式对象（request，session等）。
  - 通过它可以取得有关客户要求或页面的详细信息。如表所示：

pageContext获取客户请求数据示例表	
范例	解释说明
${pageContext.request.queryString}	取得请求的参数字符串
${pageContext.request.requestURL}	取得不包括请求参数的请求URL
${pageContext.request.contextPath}	取得服务的Web Application名称
${pageContext.request.method}	http的请求方式(GET,POST)
${pageContext.request.protocol}	取得使用的协议
${pageContext.request.remoteAddr}	取得用户的ip
${pageContext.session.new }	判断session是否为新的
${pageContext.session.id }	取得session的id值
${pageContext.servletContext.serverInfor}	取得主机端的服务信息

访问请求参数

EL表达式参数访问对象

- 在Jsp页面中，经常会进行接收其他页面或者Servlet传递过来的参数。EL为我们提供参数访问对象来获取这些参数值，如${param.name}以及${paramValues.name }。各对象的具体说明如表：

pageContext获取客户请求数据示例表	
对象名称	解释说明
param	获得客户端的请求参数的字符串值
paramValues	返回映射至客户端的请求参数的一组值

  - param以及paramValues是EL表达式有关输入隐式对象。可以获取客户端传来的参数数据。功能特征如同request获取客户参数，request调用getParameter(String name)获得客户端name参数对应的参数数据，调用getParameterValues(String name)获得客户端name参数的一组值。如图：

request.getParameter(**name**) == ${param.name }

request.getParameterValues(**name**) == ${paramValues.name }

```
<%@ 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=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form name="loginForm" method="POST" action="login_auth.jsp">
<table>
<tr>
<td>账号:</td>
<td><input type="text" name="loginName" value=""></td>
</tr>
<tr>
<td>密码:</td>
<td><input type="password" name="loginPasswd" value=""></td>
</tr>
<tr>
<td><input type="submit" name="commit" value="提交"></td>
</tr>
</table>
</form>
</body>
</html>
```

```
<%@ 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=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<!-- 使用EL表达式获取请求参数进行判断 -->
${(param.loginName=="CSG")&&(param.loginPa
sswd=="CSG")?"登录成功":"登录失败" }<br/>
</body>
</html>
```

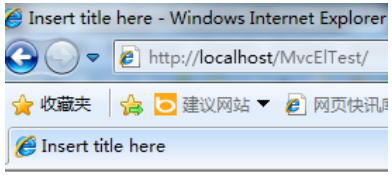
练习

- **利用MVC模式，从mysql数据库中搜索所有数据，并使用EL表达式将数据输出到jsp页面中。**

数据库创建脚本如下：请先创建CSG用户，密码为CSG12345(如果存在，则跳过)

此处请使用CSG用户登陆

```
--删除TEST_USER表
DROP TABLE TEST_USER;
--删除序列
DROP SEQUENCE userseq;
--清空回收站
PURGE RECYCLEBIN;
--创建序列
CREATE SEQUENCE userseq;
--创建表
CREATE TABLE TEST_USER(
   USERID NUMBER PRIMARY KEY,
   USERNAME VARCHAR2(50) NOT NULL,
   USERJOB VARCHAR2(50) NOT NULL,
   CONSTRAINT NK_LOGIN_NAME UNIQUE(USERNAME)
);
--插入测试数据
1. INSERT INTO TEST_USER(USERID,USERNAME,USERJOB)VALUES
(userseq.nextval,'CSG','Teacher');
2. INSERT INTO TEST_USER(USERID,USERNAME,USERJOB)VALUES
(userseq.nextval,'TOM','FARMER');
3. INSERT INTO TEST_USER(USERID,USERNAME,USERJOB)VALUES
(userseq.nextval,'JERRY','CEO');
4. INSERT INTO TEST_USER(USERID,USERNAME,USERJOB)VALUES
(userseq.nextval,'MARY','HR');
```



第3节JSTL标签库概述

知识点预览

#	知识点	难点	重点	应用	说明
1	服务端标签				了解服务端标签的特征和用途
2	标签库				了解标签库的构成
3	配置JSTL标签库	√	√	√	掌握在Java web应用中配置JSTL标记库

JSTL标签库介绍

- JSTL标签库的使用是为弥补html标签的不足，规范自定义标签的使用而诞生的。使用JSLT标签的目的就是不希望在jsp页面中出现java逻辑代码

服务端标签

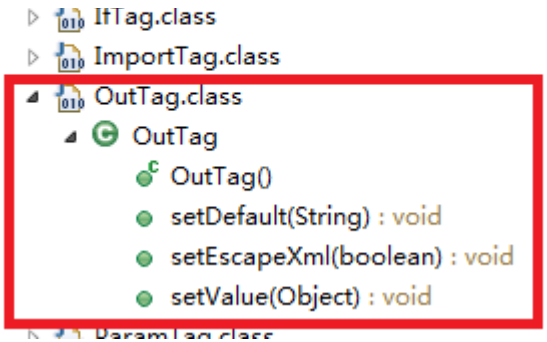
什么是服务端标签？

- 服务端标签是相对于静态标签来界定的。
  - 在Jsp页面中，**除HTML标签（静态标签）**以及Jsp指令元素外，替代Java业务逻辑语句，并且在服务端编译执行的标签，就叫做服务端标签。
- 服务端标签产生的原因：
  - Jsp页面一般构成，包括HTML标签（静态标签），Jsp指令元素，Java服务脚码程序，这些元素参杂在一起，使Jsp内容比较混乱，难以维护和更新，不便于开发。
  - 对于Java服务端业务语句，如果能够使用标签实现，可为用户提供一个无脚本环境，在此环境中，用户可以采用标签编写代码，而无需使用Java脚本。易于动静分离，易于维护更新。
- 服务端标签的优点：
  - 提供无脚本环境，易于维护更新。
  - 优于JavaBean，可以获取所在环境信息，使业务和表示分离，增强可读性，重复功能的可复用性。
  - 甚至可以自定义服务端标签，用于实现更复杂的业务。

标签库

- 什么是标签库？
  - 标签
    - 以具体到某制等。
    - 以实现最基问数据库业
  - 标签
    - 处理器。实现的接口，实现对标签的生命周期的管理。
    - XXX.jar：它 r包。
    - taglibname.t 中定义了标签库的名称、版本信息、 应的标签处理器，标签的属性等。

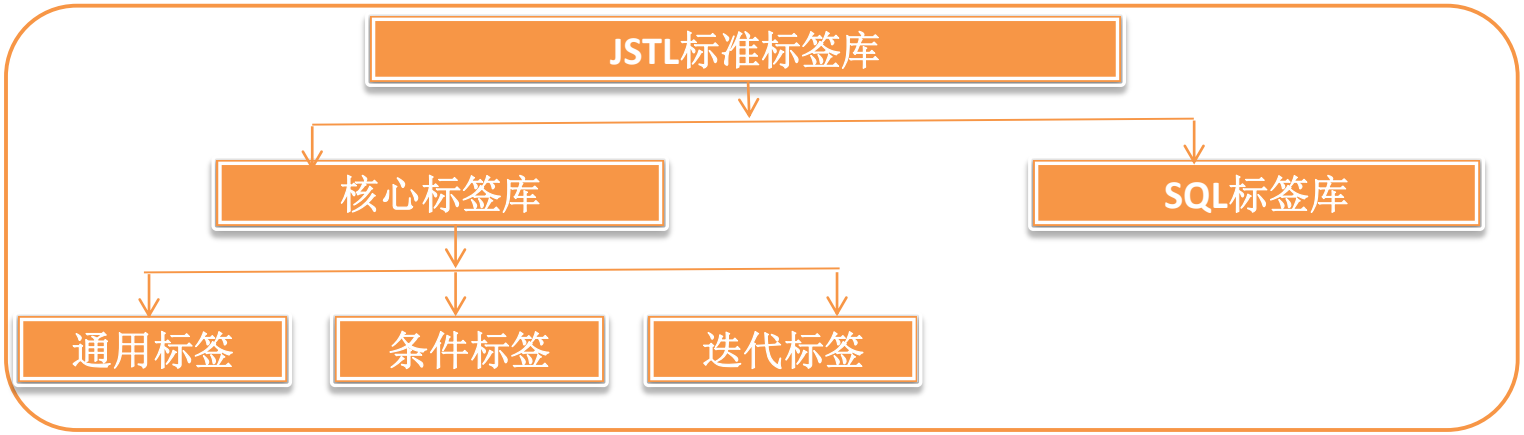
```
<tag>
    <description>
        Like &lt;%= ... &gt;, but for expressions.
    </description>
    <name>out</name>
    <tag-class>org.apache.taglibs.standard.tag.rt.core.OutTag</tag-class>
    <body-content>JSP</body-content>
    <attribute>
        <description>
Expression to be evaluated.
        </description>
        <name>value</name>
        <required>true</required>
        <rtexprvalue>true</rtexprvalue>
    </attribute>
```



配置JSTL标签库

JSTL标签库概述

- JSTL（Java Server Pages Stardard Tag Library）包含用于编写和开发Jsp页面的一组标准标签。
  - 可为用户提供一个无脚本环境，可以使用标签编写代码，无需Java脚本。
  - JSTL包含各种标签，如核心标签、迭代标签、条件标签和SQL标签等。
- JSTL提供的主要标签库如图：



配置JSTL标签库

Java web项目中配置JSTL标签库

- 在创建好的Java web项目中，向WEB-INF目录下的lib目录中导入jstl.jar以及standard.jar文件，这两个文件中是字节码文件打包文件。
- 在Jsp页面代码中，通过taglib指令导入标签库描述符文件。
  - taglib指令的语法如图：

```
<%@ taglib uri="标签库描述符文件"  prefix="前缀名" %>
Eg:
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

  - ✓ uri：标签库描述符文件映射路径，可以自行设定，不过此时要修改web.xml文件配置。
  - ✓ prefix：标签前置名称，可以自行设定，不过此时uri必须沿用默认。

配置JSTL标签库

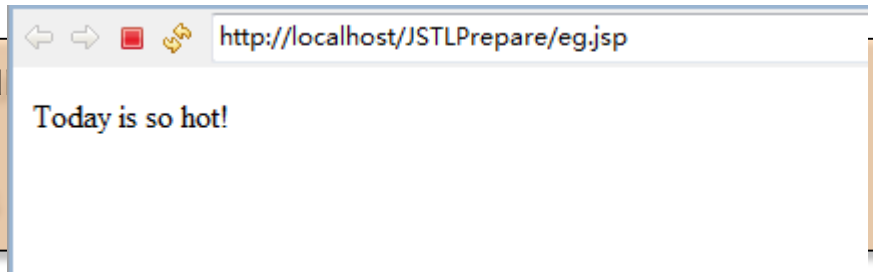
Java web项目中配置JSTL标签库

- taglib指令导入标签库时修改uri的方法：
  - uri：标签库描述符文件映射路径，可以自行设定，不过此时要修改web.xml文件配置，修改操作如图所示：

```
<jsp-config>
 <taglib>
 <taglib-uri>http://www.baidu.com</taglib-uri>
 <taglib-location>/WEB-INF/c.tld</taglib-location>
 </taglib>
</jsp-config>
```

  - prefix：标签前置名称可以自行设定，不过此时uri必须沿用默认。例如：

```
<%@ taglib prefix="d" u
应用标签时：
<d:out  value="Today is
```



第4节core标签库

知识点预览

#	知识点	难点	重点	应用	说明
1	out标签			√	掌握表达式标签out用于实现输出业务
2	if标签		√	√	掌握if标签用于条件分支结构的表达
3	choose系列标签		√	√	掌握chooce标签用于实现等值判断
4	forEach标签	√	√	√	掌握forEach标签实现循环迭代操作业务

out标签

- 核心标签库（Core tag library）
  - Core核心标签库包括：
    - 输入输出标签。
      - out标签：用于表达式的输出。
    - 流程控制标签。
      - If标签：if条件结构。
      - choose标签：多分支等值判断结构。
        - ✓ when标签：相当于Java语法中，swtich结构的case语句。
        - ✓ otherwise：相当于Java语法中，swtich结构的default语句。
    - 迭代标签。
      - forEach：相当于Java语法中的循环迭代结构。
  - Jsp页面中使用Core核心标签库的语法：

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

<c:out>

- 输出数据对象（字符串、表达式）的内容或结果
- 在使用Java脚本输出时常使用的方式为：
- <% out.println(“字符串”)%> 或者 <%=表达式%> ，在web开发中，为了避免暴露逻辑代码会尽量减少页面中的Java脚本，使用<c:out>标签就可以实现以上功能。

out标签

out标签

- out标签是core核心标签库的输出标签，用于显示数据信息。
- out标签的语法形式：
  - 语法1：out标记中没有body（本体）内容。

```
<c:out  value="需要显示的信息"  [default="defaultValue"]  />
```

  - 语法2：out标记中有body（本体）内容。

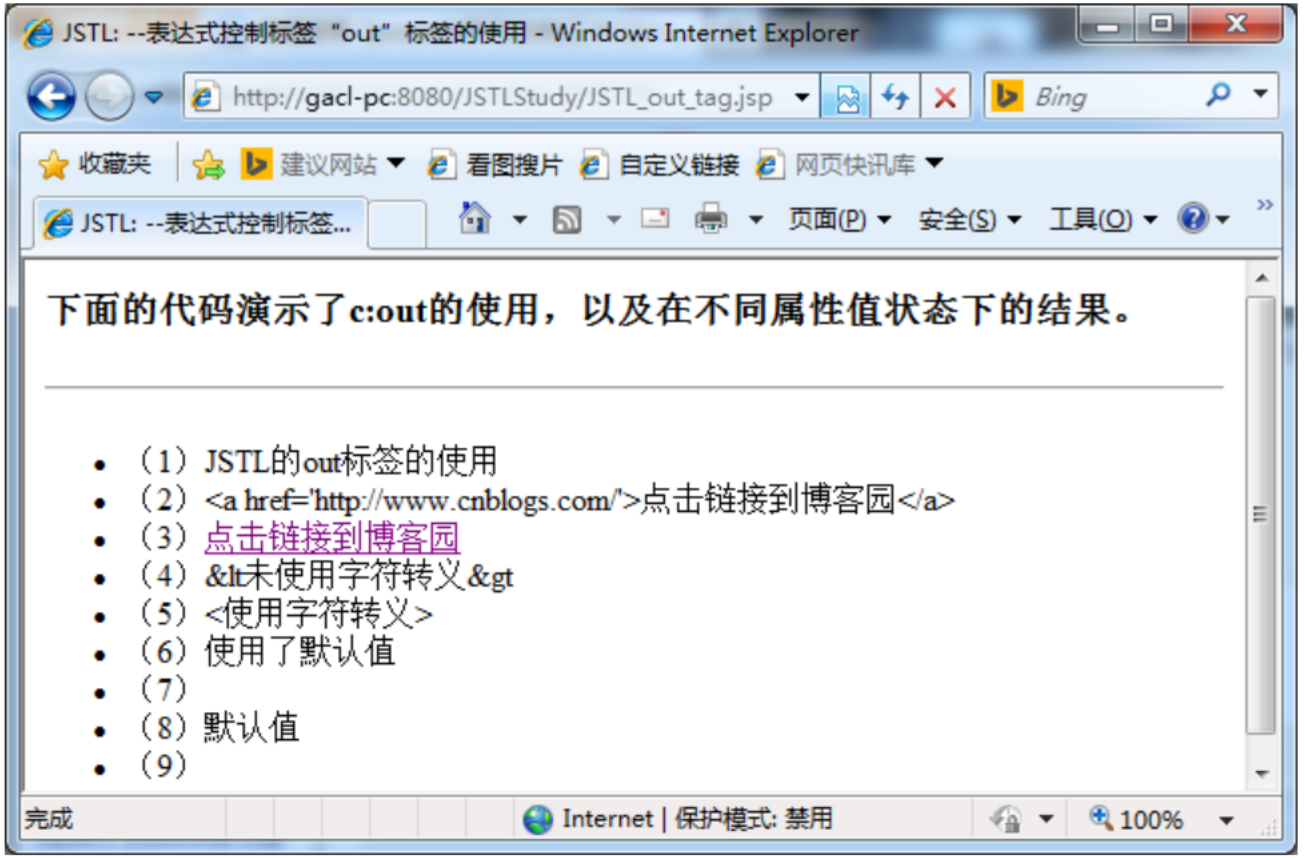
```
<c:out  value="需要显示的信息" >
        default value
</c:out>
```

- out标签中属性解释，如图表：

out标签属性介绍					
名称	说明	EL	类型	必须	默认值
value	需要显示的值	Y	Object	是	无
default	如果value为null，显示default的值	Y	Object	否	无

out标签实现JSP页面输出

```
<%@ page language="java" pageEncoding="UTF-8"%>
<%--引入JSTL核心标签库 --%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE HTML>
<html>
<head>
  <title>JSTL: --表达式控制标签 “out” 标签的使用</title>
</head>

<body>
  <h3><c:out value="下面的代码
  <hr/>
  <ul>
    <%--（1）直接输出了一个
    <li>（1）<c:out value="JSTL

    <li>（2）<c:out value="<a h
    <%--escapeXml="false"表示
    <li>（3）<c:out value="<a h

    <%--(4）字符串中有转义字
    <li>（4）<c:out value="&lt;
    <%--（5）使用了转义字符
    <li>（5）<c:out value="&lt(

    <%--（6）设定了默认值，
    <li>（6）<c:out value="${nu
    <%--（7）未设定默认值，
    <li>（7）<c:out value="${nu

    <%--（8）设定了默认值，从EL表达式${null}得到空值，所以直接输出设定的默认值。 --%>
    <li>（8）<c:out value="${null}" default="默认值"/></li>
    <%--（9）未设定默认值，输出结果为空。 --%>
```



if标签

if标签

- if标签是core核心标签库的条件标签，用于条件判断结构的表达。
- if标签的语法形式：
  - 语法1：if标记中没有body（本体）内容。

```
<c:if  test="testCondition"  var="varName"
        [scope="{page|request|session|application}"] />
```

  - 语法2：if标记中有body（本体）内容。

```
<c:if  test="testCondition"  [var="varName" ]
        [scope="{page|request|session|application}"] >
        本体内容
        …
</c:if>
```

if标签

- if标签中属性解释，如图表：

if标签属性介绍					
名称	说明	EL	类型	必须	默认值
test	条件表达式，如果为true，则执行本体内容，否则，为false时不执行本体内容。	Y	boolean	是	无
var	**用来存储test运算后的结果，即true或者false。**	N	String	否	无
scope	var变量的JSP作用域范围	N	String	否	page

- If标签的举例：

```
<c:if  test="${6>0 }"  var="result"
          scope="page"] >
          <c:out value="${result }">
</c:if>
<c:out value="${result }">
```

if标签实现JSP条件业务判断

- testjstlif.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%--引入JSTL核心标签库 --%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE HTML>
<html>
<head>
    <title>JSTL: --流程控制标签 if标签示例</title>
</head>

<body>
    <h4>if标签示例</h4>
    <hr>
    <form action="JSTL_if_tag.jsp" method="post">
        <input type="text" name="uname" value="${param.uname}">
        <input type="submit" value="登录">
    </form>
    <%--使用if标签进行判断并把检验后的结果赋给adminchock，存储在默认的page范围中。 --%>
    <c:if test="${param.uname=='admin'}" var="adminchock">
    <%--可以把adminchock的属性范围设置为session，这样就可以在其他的页面中得到adminchock的值，
    使用<c:if text=”${adminchock}”><c:if>判断，实现不同的权限。 --%>
        <c:out value="管理员欢迎您！ "/>
    </c:if>
    <%--使用EL表达式得到adminchock的值，如果输入的用户名为admin将显示true。 --%>
    ${adminchock}
</body>
</html>
```

choose系列标签

choose标签

- choose标签本身只当作when标签和otherwise标签的父标签。
- choose标签的语法形式：

```
<c:choose >
        本体内容（<c:when>和<c:otherwise>）
 </c:choose>
```

- choose标签的本体内容只能有：
  - 空白。
  - 1个或者多个<c:when>标签。
  - 0个或者多个<c:otherwise>标签。
- choose标签的本体内容要求细节：
  - <c:when>和<c:otherwise>标签必须是完整的包含在choose标签中。
  - <c:when>必须在<c:otherwise>之前。
  - 如果有<c:otherwise>时，必须放在choose标签结构的最后。

choose系列标签

choose标签的子标签when

- when标签的用途，和Java语法**switch结构中case语句一样**。
- when标签的语法形式，如图：

```
<c:when  test="testCondition">
        本体内容
 </c:when>
```

- when标签中属性解释，如图表：

when标签属性介绍					
名称	说明	EL	类型	必须	默认值
test	条件表达式，如果为true，则执行本体内容，否则，为false时不执行本体内容。	Y	boolean	是	无

choose系列标签

choose标签的子标签otherwise

- otherwise标签的用途，和Java语法switch结构中default语句一样。
  - 当choose标签中的所有when标签都没有成立时，执行otherwise标签。
- otherwise标签的语法形式，如图：

```
<c:otherwise >
        本体内容
 </c:otherwise>
```

- otherwise标签使用时要点：
  - otherwise必须是在choose内部。
  - otherwise在choose标签中，必须放在最后。

choose标签实现JSP条件业务

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%--引入JSTL核心标签库 --%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE HTML>
<html>
  <head>
    <title>JSTL: -- choose及其嵌套标签标签示例</title>
  </head>

  <body>
    <h4>choose及其嵌套标签示例</h4>
    <hr/>
    <%--通过set标签设定score的值为85 --%>
    <c:set var="score" value="85"/>
    <c:choose>
    <%--使用<c:when>进行条件判断。
        如果大于等于90，输出“您的成绩为优秀”；
        如果大于等于70小于90，输出“您的成绩为良好”；
        大于等于60小于70，输出“您的成绩为及格”；
        其他（otherwise）输出“对不起，您没能通过考试”。
     --%>
      <c:when test="${score>=90}">
        你的成绩为优秀！
      </c:when>
      <c:when test="${score>70 && score<90}">
        您的成绩为良好!
      </c:when>
      <c:when test="${score>60 && score<70}">
        您的成绩为及格
      </c:when>
      <c:otherwise>
        对不起，您没有通过考试！
      </c:otherwise>
    </c:choose>
  </body>
</html>
```

forEach标签

forEach标签

- forEach标签的用途，为循环控制。
- forEach标签的语法形式，有两种：
  - 语法1：forEach标记可以迭代一集合对象的所有成员。

```
<c:forEach  var="varName" items="collecitons"
 [varStatus="vsName"]  [begin="begin"]  [end="end"] [step="step"]>
        本体内容
 </c:forEach>
```

  - 语法2： forEach标记迭代指定次数。

```
<c:forEach  [var="varName"]  [items="collecitons"]
 [varStatus="vsName"]   begin="begin"  end="end"  [step="step"]>
        本体内容
 </c:forEach>
```

forEach标签

forEach标签

- forEach标签中有关属性解释，如图表：

forEach标签属性介绍					
名称	说明	EL	类型	必须	默认值
var	用于存放现在指到的成员	N	String	否	无
items	被迭代的集合对象	Y	Arrays Collections String iterator Map Enum	否	无
varStatus	用于存放所指成员的相关信息，比如索引等。	N	String	否	无
begin	开始的位置	Y	int	否	0
end	结束的位置	Y	int	否	最后1个
step	每次迭代的步伐	Y	int	否	1

forEach标签

forEach标签

- forEach标签中varStatus属性特性解释；
  - varStatus属性可以获得有关集合项信息。
- varStatus属性有关集合项的信息特性，如图表：

forEach标签varStatus属性介绍		
名称	说明	类型
count	所有迭代项从1开始的当前迭代项计数	String
index	所有迭代项从0开始的当前索引	int
first	判断当前迭代项是否第一项	boolean
last	判断当前迭代项是否最后一项	boolean
current	当前迭代集合中的项	Object

forEach标签实现JSP迭代业务

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%--引入JSTL核心标签库 --%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@page import="java.util.ArrayList"%>
<!DOCTYPE HTML>
<html>
  <head>
    <title>JSTL: -- forEach标签实例</title>
  </head>

  <body>
   <h4><c:out value="forEach实例"/></h4>
   <%
         List<String>list = new ArrayList<String>();
         list.add(0, "贝贝");
         list.add(1, "晶晶");
         list.add(2, "欢欢");
         list.add(3, "莹莹");
         list.add(4, "妮妮");
         request.setAttribute("list", list);
    %>
    <B><c:out value="不指定begin和end的迭代：" /></B><br>
    <%--不使用begin和end的迭代，从集合的第一个元素开始，遍历到最后一个元素。 --%>
    <c:forEach var="fuwa" items="${list}">
        <c:out value="${fuwa}"/><br/>
    </c:forEach>

    <B><c:out value="指定begin和end的迭代：" /></B><br>
    <%--指定begin的值为1、end的值为3、step的值为2，
       从第二个开始首先得到晶晶，每两个遍历一次，
       则下一个显示的结果为莹莹，end为3则遍历结束。 --%>
    <c:forEach var="fuwa" items="${list}" begin="1" end="3" step="2">
        <c:out value="${fuwa}"/><br/>
```

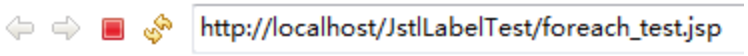
课堂练习

课堂练习4： forEach标签的使用

- 任务描述
  - 使用forEach标签输出字符串数组names。
  - String[] names={"TOM","JERRY","MARY","JACK","ROSE","DICK"};
  - 输出形式为：
    - 1.输出全部。
    - 2.输出全部(其中的间隔为2)。
    - 3.输出前三个。
- 关联知识点
  - EL表达式和forEach标签的使用。
- 实现步骤
  - 设置page范围属性。
  - 使用EL表达式取得属性值。
  - 使用forEach标签按照相应形式输出。

课堂练习

页面运行效果



输出全部

TOM

JERRY

MARY

JACK

ROSE

DICK

输出全部(间隔2个输出)

TOM

MARY

ROSE

输出前三个

TOM

JERRY

MARY

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!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=ISO-8859-1">
<title>Insert title here</title>
</head>
<%!
                String[] names={"TOM","JERRY","MARY","JACK","ROSE","DICK"};
%>
<%
                //将names加入page范围属性
                pageContext.setAttribute("names", names);
%>
<%--使用forEach标签输出循环 --%>
<body>
<h3>输出全部</h3>
<c:forEach items="${names }" var="name" >
<h3>${name }</h3>
</c:forEach>
<h3>输出全部(间隔2个输出)</h3>
<c:forEach items="${names }" var="name" step="2" >
<h3>${name }</h3>
</c:forEach>
<h3>输出前三个</h3>
<c:forEach items="${names }" var="name" begin="0" end="2" >
<h3>${name }</h3>
</c:forEach>
</body>
</html>
```

URL操作标签

- <c:url>标签用于在JSP页面中构造一个URL地址，其主要目的是实现URL重写。
- 【语法1】：指定一个url不做修改，可以选择把该url存储在JSP不同的范围中。

```
<c:url
   value=”value”
   [var=”name”]
   [scope=”page|request|session|application”]
   [context=”context”]/>
```

【语法2】：配合 <c:param>标签给url加上指定参数及参数值，可以选择以name存储该url。

```
<c:url
   value=”value”
   [var=”name”]
   [scope=”page|request|session|application”]
   [context=”context”]>
        <c:param name=”参数名” value=”值” >
</c:url>
```

testjstlurl.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%--引入JSTL核心标签库 --%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE HTML>
<html>
<head>
   <title>JSTL: -- url标签实例</title>
</head>

<body>
   <c:out value="url标签使用"></c:out>
   <h4>使用url标签生成一个动态的url，并把值存入session中.</h4>
   <hr/>
   <c:url value="http://www.baidu.com" var="url" scope="session">
   </c:url>
   <a href="${url}">百度首页(不带参数)</a>
   <hr/>
   <h4>
      配合 &lt;c:param&gt;标签给url加上指定参数及参数值，生成一个动态的url然后存储到paramUrl变量中
   </h4>
   <c:url value="http://www.baidu.com" var="paramUrl">
      <c:param name="userName" value="孤傲苍狼"/>
      <c:param name="pwd">123456</c:param>
   </c:url>
   <a href="${paramUrl}">百度首页(带参数)</a>
</body>
</html>
```

课堂练习

课堂练习5：从数据库读入信息

- 任务描述
  - 利用MVC模式，从oracle数据库中的CSG. TEST_DEPARTMENTS表中读取所有部门名称。
  - 在JSP网页中只能使用EL表达式和core库标签
- 关联知识点
  - CORE标签库中常用标签的使用。
  - 配合EL表达式处理进行简单的数据处理。
- 实现步骤
  - 使用servlet从数据库中读取所有信息传输到JSP页面中。
  - 在JSP页面使用EL表达式和CORE标签输出信息。

课堂练习

数据库创建：

请先创建CSG用户，密码为CSG12345(如果存在，则跳过)

使用CSG用户登陆

```
--删除TEST_DEPARTMENTS表
DROP TABLE TEST_DEPARTMENTS;
--清空回收站
PURGE RECYCLEBIN;
--创建表
CREATE TABLE TEST_DEPARTMENTS(
    DE_ID NUMBER PRIMARY KEY,
    DE_NAME VARCHAR2(50) NOT NULL
);
```

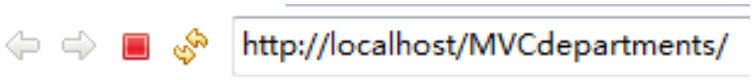
课堂练习

- 插入测试用数据：

```
INSERT INTO TEST_DEPARTMENTS(DE_ID,DE_NAME)VALUES
  (1,'HR');
INSERT INTO TEST_DEPARTMENTS(DE_ID,DE_NAME)VALUES
  (2,'PROC');
INSERT INTO TEST_DEPARTMENTS(DE_ID,DE_NAME)VALUES
  (3,'SALE');
INSERT INTO TEST_DEPARTMENTS(DE_ID,DE_NAME)VALUES
  (4,'EXECUTIVE');
INSERT INTO TEST_DEPARTMENTS(DE_ID,DE_NAME)VALUES
  (5,'IT');
INSERT INTO TEST_DEPARTMENTS(DE_ID,DE_NAME)VALUES
  (6,'ADMINISTRATION');
```

课堂练习

页面运行效果



以下为所有部门名称

部门名:HR
部门名:PROC
部门名:SALE
部门名:EXECUTIVE
部门名:IT
部门名:ADMINISTRATION

第5节function函数库

知识点预览

#	知识点	难点	重点	应用	说明
1	函数库概述			√	了解函数库的本质和构成
2	length函数		√	√	掌握length函数的应用
3	trim函数		√	√	掌握trim函数的应用

- 由于在JSP页面中显示数据时，经常需要对显示的字符串进行处理，SUN公司针对于一些常见处理定义了一套EL函数库供开发者使用。
- 这些EL函数在JSTL开发包中进行描述，因此在JSP页面中使用SUN公司的EL函数库，需要导入JSTL开发包，并在页面中导入EL函数库

函数库概述

函数标签库

- 什么是函数标签库？
  - JSTL函数标签库就是一些常用的函数，在JSTL中把这些常用的函数封装成标签的形式，然后可以在JSP页面上进行方便的调用。
- 函数标签库的真实形象
  - 称呼Functions标签库为标签库，倒不如称呼其为函数库来得更容易理解些。因为Functions标签库**并没有提供传统的标签**来为JSP页面的工作服务，而是被**用于EL表达式语句中**。
    - 使用函数标签的语法：

```
${fn:functionName(paramValue) }
```

  - Jsp页面导入函数库的语法，如图：

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

函数库概述

函数标签库分类

- 函数标签库分为两类
  - 长度函数：fn:length。
  - 字符串处理函数表（15个）：

字符串处理函数介绍	
名称	说明
fn:contains(String, substring)	如果参数String中包含参数substring，返回true
fn:trim(String)	去除参数String 首尾的空格，并将其返回
fn:toLowerCase(String)	返回参数字符串的小写形式
fn:toUpperCase(String)	返回参数字符串的大写形式
fn:indexOf(String, substring)	返回在String中的第一次出现substring的位置
fn:containsIgnoreCase(string, substring)	字符串string中判断是否有substring（忽略大小写）
fn:endsWith(string, suffix)	如果参数 string 以参数suffix结尾，返回true

函数库概述

函数标签库

◆ 字符串处理函数：

字符串处理函数介绍	
名称	说明
fn:join(array, separator)	将一个给定的数组array用给定的间隔符separator串在一起，组成一个新的字符串并返回
fn:replace(string, before, after)	用参数after字符串替换参数string中所有出现参数before字符串的地方,并返回替换后的结果
fn:split(string, separator)	返回一个数组，以参数separator 为分割符分割参数string，分割后的每一部分就是数组的一个元素
fn:substring(string, begin, end)	返回参数string部分字符串, 从参数begin开始到参数end位置，包括end位置的字符
fn:substringAfter(string, substring)	返回参数substring在参数string中后面的那一部分字符串
fn:substringBefore(string, sbstring)	返回参数substring在参数string中前面的那一部分字符串
fn:escapeXml(string)	将有特殊意义的XML (和HTML)转换为对应的XML character entity code，并返回
fn:startsWith(string, prefix)	如果参数string以参数prefix开头，返回true

length函数

长度函数

- 长度函数原型和功能：
  - 原型： int fn:length(item) 。Item是集合或者字符串。
  - 功能：返回参数item中包含元素的数量。参数Item类型是数组、collection或者String。如果是String类型,返回值是String中的字符数。
- 长度函数的由来
  - 长度函数fn:length的出现有重要的意义。在JSTL1.0中，有一个功能被忽略了，那就是对集合的长度取值。
  - 虽然java.util.Collection接口定义了size方法，但是该方法不是一个标准的JavaBean属性方法（没有get,set方法），因此，无法通过EL表达式“${collection.size}”来轻松取得。
  - fn:length函数正是为了解决这个问题而被设计出来的。它的参数为input，将计算通过该属性传入的对象长度。该对象应该为集合类型或String类型。其返回结果是一个int类型的值。

trim函数

trim函数

- trim函数原型和功能：
  - 原型： String fn:trim(string) 。参数string是字符串。
  - 功能：返回一个字符串，字符串的内容为，去掉前后空格后的字符串。
- trim函数的常用场景
  - 通过客户端控件获取用户信息并传递数据到服务程序时。
- trim函数处理字符串的原因
  - 由于空格在客户端程序的控件中，不易察觉，而空格本身也是字符，所以，经常会被当作字符串内容的一部分发送，而易造成错误录入信息。

课堂练习

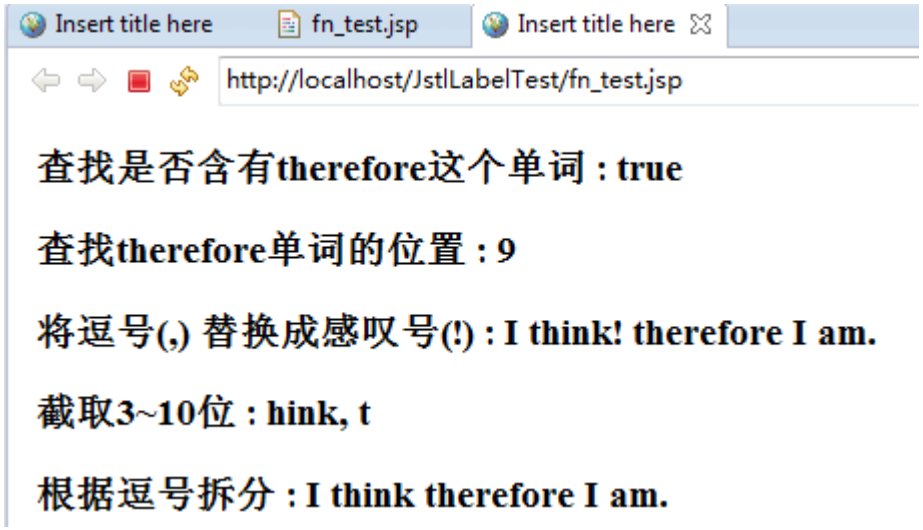
课堂练习1：函数标签库的使用

- 任务描述
  - 使用函数标签库中的标签处理info字符串。
  - 要求：对info字符串做一下处理：
    - 1.String info="I think, therefore I am. “
    - 2.查找info中是否含有therefore这个单词
    - 3.查找therefore单词的位置
    - 4.将逗号(,)替换成感叹号(!)
    - 5.将info截取3~10位
    - 6.根据逗号(,)拆分
- 关联知识点
  - EL表达式和函数标签库中的标签的使用
- 实现步骤
  - 设置page范围属性,使用EL表达式取得属性值,使用函数标签库中的标签对字符串进行处理并输出结果。

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
    <%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>
<!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=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
            String info="I think, therefore I am.";
            pageContext.setAttribute("info", info);
%>
<h3>查找是否含有therefore这个单词： ${fn:contains(info,"therefore") }</h3>
<h3>查找therefore单词的位置： ${fn:indexOf(info,"therefore") }</h3>
<h3>将逗号(,) 替换成感叹号(!)： ${fn:replace(info,",","!")}</h3>
<h3>截取3~10位： ${fn:substring(info,"3","10")}</h3>
<h3>根据逗号拆分： ${fn:split(info,",")[0]}  ${fn:split(info,",")[1]}</h3>
</body>
</html>
```

课堂练习

页面运行效果


Insert title here
fn_test.jsp
Insert title here
http://localhost/JstlLabelTest/fn_test.jsp
查找是否含有therefore这个单词 : true
查找therefore单词的位置 : 9
将逗号(,) 替换成感叹号(!) : I think! therefore I am.
截取3~10位 : hink, t
根据逗号拆分 : I think therefore I am.

自定义标签

- 自定义标签主要用于移除Jsp页面中的java代码

自定义标签开发和使用

- 1、编写一个实现Tag接口的Java类(标签处理器类)
- 2、在WEB-INF/目录下新建tld文件，在tld文件中对标签处理器类进行描述

在Jsp页面中使用自定义标签

- 1、使用"<%@taglib uri="标签库的uri" prefix="标签的使用前缀"%>"指令引入要使用的标签库

自定义标签的执行流程

- JSP引擎遇到自定义标签时，首先创建标签处理器类的实例对象，然后按照JSP规范定义的通信规则依次调用它的方法。
- 1、public void setPageContext(PageContext pc)， JSP引擎实例化标签处理器后，将调用setPageContext方法将JSP页面的pageContext对象传递给标签处理器，标签处理器以后可以 通过这个pageContext对象与JSP页面进行通信。
- 2、public void setParent(Tag t)，setPageContext方法执行完后，WEB容器接着调用的setParent方法将当前标签的父标 签传递给当前标签处理器，如果当前标签没有父标签，则传递给setParent方法的参数值为null。
- 3、public int doStartTag()，调用了setPageContext方法和setParent方法之后，WEB容器执行到自定义标签的开始标记时，就会调用标签处理器的doStartTag方法。
- 4、public int doEndTag()，WEB容器执行完自定义标签的标签体后，就会接着去执行自定义标签的结束标记，此时，WEB容器会去调用标签处理器的doEndTag方法。
- 5、public void release()，通常WEB容器执行完自定义标签后，标签处理器会驻留在内存中，为其它请求服务器，直至停止web应用时，web容器才会调用release方法。

```
package me.gacl.web.tag;
import java.io.IOException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.PageContext;
import javax.servlet.jsp.tagext.Tag;
public class ViewIPTag implements Tag {    //接收传递进来的PageContext对象
    private PageContext pageContext;

    @Override
    public int doEndTag() throws JspException {
        System.out.println("调用doEndTag()方法");
        return 0;
    }

    @Override
    public int doStartTag() throws JspException {
        System.out.println("调用doStartTag()方法");
        HttpServletRequest request =(HttpServletRequest) pageContext.getRequest();
        JspWriter out = pageContext.getOut();
        String ip = request.getRemoteAddr();
        try {
            //这里输出的时候会抛出IOException异常
            out.write(ip);
        } catch (IOException e) {
            //捕获IOException异常后继续抛出
            throw new RuntimeException(e);
        }
        return 0;
    }

    @Override
    public Tag getParent() {
        return null;
    }

    @Override
    public void release() {
        System.out.println("调用release()方法");
    }

    @Override
    public void setPageContext(PageContext pageContext) {
        System.out.println("setPageContext(PageContext  pageContext)");
        this.pageContext = pageContext;
    }

    @Override
    public void setParent(Tag arg0) {

    }

}
```

gacl.tld

```
<?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用来添加对taglib(标签库)的描述 -->
    <description>孤傲苍狼开发的自定义标签库</description>
    <!--taglib(标签库)的版本号 -->
    <tlib-version>1.0</tlib-version>
    <short-name>GaclTagLibrary</short-name>
    <!--
       为自定义标签库设置一个uri，uri以/开头，/后面的内容随便写，如这里的/gacl，
       在Jsp页面中引用标签库时，需要通过uri找到标签库
       在Jsp页面中就要这样引入标签库：<%@taglib uri="/gacl" prefix="gacl"%>
    -->
    <uri>/gacl</uri>

    <!--一个taglib(标签库)中包含多个自定义标签，每一个自定义标签使用一个tag标记来描述 -->
    <!-- 一个tag标记对应一个自定义标签 -->
     <tag>
       <description>这个标签的作用是用来输出客户端的IP地址</description>
       <!--
          为标签处理器类配一个标签名，在Jsp页面中使用标签时是通过标签名来找到要调用的标签处理器类的
          通过viewIP就能找到对应的me.gacl.web.tag.ViewIPTag类
        -->
       <name>viewIP</name>
       <!-- 标签对应的处理器类-->
       <tag-class>me.gacl.web.tag.ViewIPTag</tag-class>
       <body-content>empty</body-content>
    </tag>

</taglib>
```

jspTag_Test1.jsp

```
<%@ page language="java" pageEncoding="UTF-8"%>
<!-- 使用taglib指令引用gacl标签库，标签库的前缀(prefix)可以随便设置，如这里设置成 prefix="xdp"
-->
<%@taglib uri="/gacl"  prefix="xdp"%>
<!DOCTYPE HTML>
<html>
  <head>
    <title>输出客户端的IP</title>
  </head>

  <body>
    你的IP地址是(使用java代码获取输出)：
    <%
        //在jsp页面中使用java代码获取客户端IP地址
        String ip = request.getRemoteAddr();
        out.write(ip);
    %>
    <hr/>
     你的IP地址是(使用自定义标签获取输出)：
     <%--使用自定义标签viewIP --%>
     <xdp:viewIP/>
  </body>
</html>
```