**EL表达式和JSTL标签**

拓薪教育

刘洋

1. EL表达式(熟练)
   1. EL表达式概述

EL（Expression Language）是一门表达式语言，它对应<%=…%>。我们知道在JSP中，表达式会被输出，所以EL表达式也会被输出。

EL表达式的格式：${…}，例如：${1+2}。

在使用EL表达式时，要求page指令的isELIgnored属性为false。这样JSP在编译成.java时，才不会忽略EL表达式。

如果你希望某个EL表达式被JSP编译器忽略，那么可以在EL表达式之前添加“\”，例如：\${1+2}。

* 1. EL表达式运算(熟练)

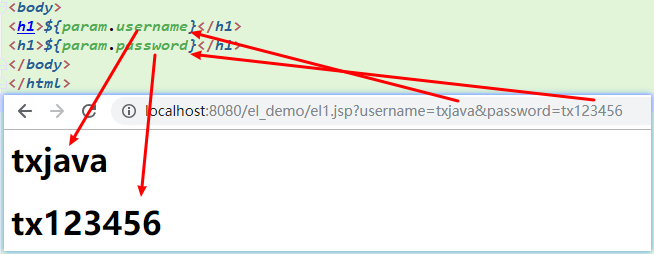
我们可以使用EL表达式来进行简单的逻辑运算

|  |  |  |  |
| --- | --- | --- | --- |
| 运算符 | 说明 | 范例 | 结果 |
| + | 加 | ${17+5} | 22 |
| - | 减 | ${17-5} | 12 |
| \* | 乘 | ${17\*5} | 85 |
| /或div | 除 | ${17/5}或${17 div 5} | 3 |
| %或mod | 取余 | ${17%5}或${17 mod 5} | 2 |
| ==或eq | 等于 | ${5==5}或${5 eq 5} | TRUE |
| !=或ne | 不等于 | ${5!=5}或${5 ne 5} | FALSE |
| <或lt | 小于 | ${3<5}或${3 lt 5} | TRUE |
| >或gt | 大于 | ${3>5}或${3 gt 5} | FALSE |
| <=或le | 小于等于 | ${3<=5}或${3 le 5} | TRUE |
| >=或ge | 大于等于 | ${3>=5}或${3 ge 5} | FALSE |
| &&或and | 并且 | ${true&&false}或${true and false} | FALSE |
| !或not | 非 | ${!true}或${not true} | FALSE |
| ||或or | 或者 | ${true||false}或${true or false} | TRUE |
| empty | 是否为空 | ${empty “”}，可以判断字符串、数组、集合的长度是否为0，为0返回true。empty还可以与not或!一起使用。${not empty “”} | TRUE |

* 1. EL表达式操作对象(熟练)
* 操作变量和常量：${name}、${8}；
* 操作List和数组：${list[0]}、${arr[0]}；
* 操作bean的属性：${person.name}、${person[‘name’]}，对应person.getName()方法；
* 操作Map的值：${map.key}、${map[‘key’]}，对应map.get(key)。
  1. EL表达式内置对象(了解)

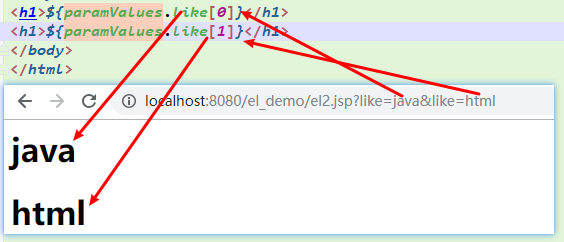
在EL表达式中，无需创建就可以使用的对象称之为EL隐藏（隐含、内置）对象。在EL中一共有11个隐藏对象，它们都与Map相似。其中10个是Map，一个是 PageContext;

* + 1. 参数隐藏对象
* param：param是Map<String,String>类型！param对象可以用来获取参数，与request.getParameter()方法相同。

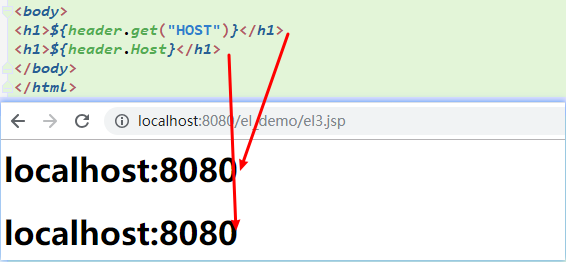


**注意，在使用EL获取参数时，如果参数不存在，返回的是空字符串，而不是null。这一点与使用request.getParameter()方法是不同的。**

* paramValues：paramValues是Map<String, String[]>类型，当一个参数名，对应多个参数值时可以使用它。



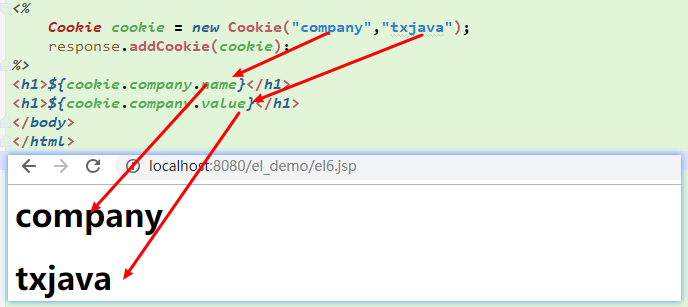
* header：header是Map<String,String>类型，用来获取请求头。



* headerValues：headerValues是Map<String,String[]>类型。当一个请求头名称，对应多个值时，使用该对象，这里就不在赘述。
* initParam：initParam是Map<String,String>类型。它对应web.xml文件中的<context-param>参数。



* cookie：cookie是Map<String,Cookie>类型，其中key是Cookie的名字，而值是Cookie对象本身。



* + 1. 域隐藏对象(精通)

使用EL表达式最为常用的就是获取域对象中保存的数据。例如：${pageScope.xxx}，表示获取在pageContext保存的数据。当然${pageScope[‘xxx’]}是相同的！

pageScope：pageScope是Map<String,Object>类型，${pageScope.xxx}的功能相等与pageContext.getAttribute(“xxx”)。两者的区别在于，前者在数据不存在时返回空字符串，而后者返回null。



* requestScope：requestScope是Map<String, Object>类型，装载了request对象中的所有数据；
* sessionScope：sessionScope是Map<String, Object>类型，装载了session对象中的所有数据；
* applicationScope：applicationScope是Map<String, Object>类型，装载了application对象中的所有数据；

当EL中给出的不是隐藏对象时，表示在四个域中查找数据。例如：${a}，表示

* 在${pageScope.a}中查找，如果找到就返回；
* 在${requestScope.a}中查找，如果找到就返回；
* 在${sessionScope.a}中查找，如果找到就返回；
* 在${applicationScope.a}中查找，如果找到就返回，找不到就返回空字符串。
  + 1. PageContext对象(了解)

pageContext：pageContext是PageContext类型！可以使用pageContext对象调用getXXX()方法，例如pageContext.getRequest，可以${pageContext.request}。

|  |  |
| --- | --- |
| Expression | 说明 |
| ${pageContext.request.queryString} | pageContext.getRequest().getQueryString(); |
| ${pageContext.request.requestURL} | pageContext.getRequest().getRequestURL(); |
| ${pageContext.request.contextPath} | pageContext.getRequest().getContextPath(); |
| ${pageContext.request.method} | pageContext.getRequest().getMethod(); |
| ${pageContext.request.protocol} | pageContext.getRequest().getProtocol(); |
| ${pageContext.request.remoteUser} | pageContext.getRequest().getRemoteUser(); |
| ${pageContext.request.remoteAddr} | pageContext.getRequest().getRemoteAddr(); |
| ${pageContext.session.new} | pageContext.getSession().isNew(); |
| ${pageContext.session.id} | pageContext.getSession().getId(); |
| ${pageContext.servletContext.serverInfo} | pageContext.getServletContext().getServerInfo(); |

1. JSTL标签(精通)
   1. JSTL概述
      1. 什么是JSTL

JSP 标准标记库（JSP Standard Tag Library，JSTL）是一个实现 Web 应用程序中常见的通用功能的定制标记库集，这些功能包括迭代和条件判断、数据管理格式化、XML 操作以及数据库访问。既然是第三方的东西，那么就需要导包。但是如果你使用的是MyEclipse，那么MyEclipse会帮你导包的。

* + 1. 导入标签库

导入标签库需要使用taglib指令！

|  |
| --- |
| <%@ ***taglib*** prefix="**c**" uri="**http://java.sun.com/jsp/jstl/core**" %> <%@ ***taglib*** prefix="**fmt**" uri="**http://java.sun.com/jsp/jstl/fmt**" %> <%@ ***taglib*** prefix="**fun**" uri="**http://java.sun.com/jsp/jstl/functions**" %> |

这东西不用去背，可以把光标放到uri属性位置上，使用Alt+/来查看uri列表，找到其中包含jstl，并且包含core的就是了。

* 1. JSTL核心库(熟练)
     1. c:out和c:set(精通)
* c:out

<c:out value=”aaa”/> 输出aaa字符串常量

<c:out value=”${aaa}”/> 与${aaa}相同

<c:out value=”${aaa}” default=”xxx”/> 当${aaa}不存在时，输出xxx字符串



<c:out value="${a }" default="xxx" escapeXml="false" /> 当escapeXml为false，不会转换“<”、“>”。这可能会受到JavaScript攻击。

* c:set

<c:set var=”a” value=”hello”/> 在pageContext中添加name为a，value为hello的数据。

<c:set var=”a” value=”hello” scope=”session”/> 在session中添加name为a，value为hello的数据。



* + 1. c:if和c:choose(熟练)
* c:if

if标签的test属性必须是一个boolean类型的值，如果test的值为true，那么执行if标签的内容，否则不执行。

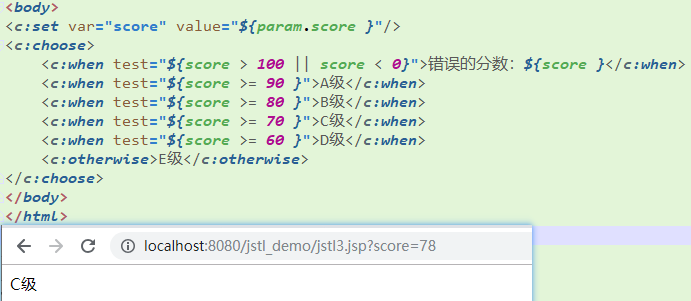


* c:choose

choose标签对应Java中的if/else if/else结构

when标签的test为true时，会执行这个when的内容。

当所有when标签的test都为false时，才会执行otherwise标签的内容



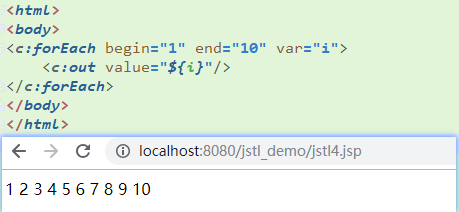
* + 1. c:forEach(精通)

forEach当前就是循环标签了,可以进行

**基本遍历:**

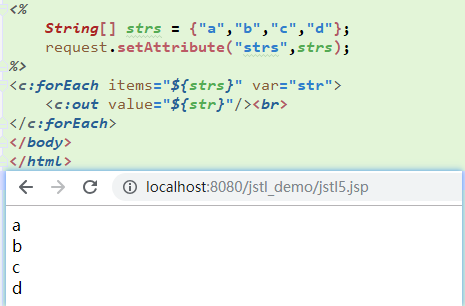
下面例子中设置域变量i，初始值为1，结束值为10。即i的值从1遍历到10，包含10。

|  |
| --- |
| <***c:forEach*** begin**="1"** end**="10"** var**="i"**>  <***c:out*** value**="*${i}*"**/> </***c:forEach***> |



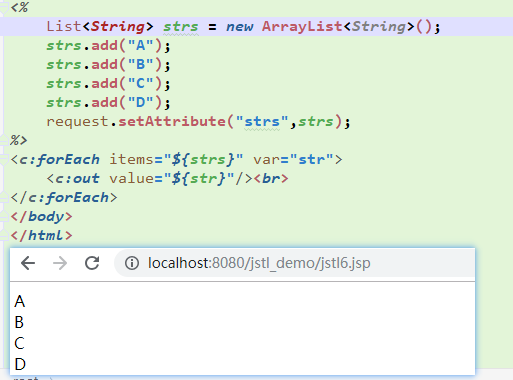
**遍历数组:**

|  |
| --- |
| <***c:forEach*** items**="*${strs}*"** var**="str"**>  <***c:out*** value**="*${str}*"**/>**<*br*>** </***c:forEach***> |



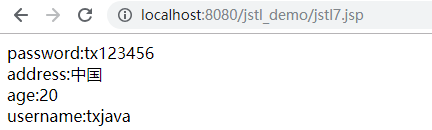
**遍历集合:**

|  |
| --- |
| ***<%*** List**<*String*> *strs* = *new* ArrayList<*String*>();  *strs*.add("A");  *strs*.add("B");  *strs*.add("C");  *strs*.add("D");** request**.setAttribute("strs",*strs*); *%>*** <***c:forEach*** items**="*${strs}*"** var**="str"**>  <***c:out*** value**="*${str}*"**/>**<*br*>** </***c:forEach***> |



**遍历Map:**

|  |
| --- |
| ***<%*** Map**<*String*,*String*> *map* = *new* HashMap<*String*,*String*>();  *map*.put("username","txjava");  *map*.put("password","tx123456");  *map*.put("age","20");  *map*.put("address","中国");** request**.setAttribute("users",*map*); *%>*** <***c:forEach*** items**="*${users}*"** var**="user"**>  <***c:out*** value**="*${user*.*key}*"**/>:<***c:out*** value**="*${user*.*value}*"**/>**<*br*>** </***c:forEach***> |



**varStatus:**

forEach标签还有一个属性：varStatus，这个属性用来指定接收“循环状态”的变量名，例如：<forEach varStatus=”vs” …/>，这时就可以使用vs这个变量来获取循环的状态了。

current当前这次迭代的（集合中的）项

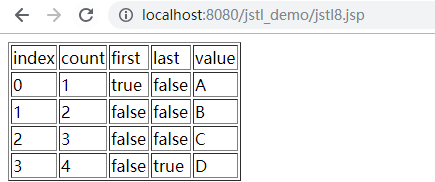
index当前这次迭代从 0 开始的迭代索引

count当前这次迭代从 1 开始的迭代计数

first用来表明当前这轮迭代是否为第一次迭代的标志

last用来表明当前这轮迭代是否为最后一次迭代的标志

|  |
| --- |
| ***<%*** List**<*String*> *users* = *new* ArrayList<*String*>();  *users*.add("A");  *users*.add("B");  *users*.add("C");  *users*.add("D");** request**.setAttribute("users",*users*); *%>* <*table* border="1">  <*tr*>  <*td*>**index**</*td*>  <*td*>**count**</*td*>  <*td*>**first**</*td*>  <*td*>**last**</*td*>  <*td*>**value**</*td*>  </*tr*>** <***c:forEach*** items**="*${users}*"** var**="user"** varStatus**="vs"**>  **<*tr*>  <*td*>** <***c:out*** value**="*${vs*.*index}*"**/>**</*td*>  <*td*>** <***c:out*** value**="*${vs*.*count}*"**/>**</*td*>  <*td*>** <***c:out*** value**="*${vs*.*first}*"**/>**</*td*>  <*td*>** <***c:out*** value**="*${vs*.*last}*"**/>**</*td*>  <*td*>** <***c:out*** value**="*${user}*"**/>**</*td*>  </*tr*>** </***c:forEach***> **</*table*>** |



**如果对集合进行遍历,没有输出结果,那么请检查你的EL表达式内的变量名称是否正确**

* 1. JSTL格式化标签库(了解)

导入标签库需要使用taglib指令！

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/format" %>\

* + 1. fmt:formatDate

|  |  |  |  |
| --- | --- | --- | --- |
| value | 要显示的日期 | 是 | 无 |
| type | DATE, TIME, 或 BOTH | 否 | date |
| dateStyle | FULL, LONG, MEDIUM, SHORT, 或 DEFAULT | 否 | default |
| timeStyle | FULL, LONG, MEDIUM, SHORT, 或 DEFAULT | 否 | default |
| pattern | 自定义格式模式 | 否 | 无 |
| timeZone | 显示日期的时区 | 否 | 默认时区 |
| var | 存储格式化日期的变量名 | 否 | 显示在页面 |
| scope | 存储格式化日志变量的范围 | 否 | 页面 |

* + 1. fmt:formatNumber

|  |  |  |  |
| --- | --- | --- | --- |
| **属性** | **描述** | **是否必要** | **默认值** |
| value | 要显示的数字 | 是 | 无 |
| type | NUMBER，CURRENCY，或 PERCENT类型 | 否 | Number |
| pattern | 指定一个自定义的格式化模式用与输出 | 否 | 无 |
| currencyCode | 货币码（当type="currency"时） | 否 | 取决于默认区域 |
| currencySymbol | 货币符号 (当 type="currency"时) | 否 | 取决于默认区域 |
| groupingUsed | 是否对数字分组 (TRUE 或 FALSE) | 否 | TRUE |
| maxIntegerDigits | 整型数最大的位数 | 否 | 无 |
| minIntegerDigits | 整型数最小的位数 | 否 | 无 |
| maxFractionDigits | 小数点后最大的位数 | 否 | 无 |
| minFractionDigits | 小数点后最小的位数 | 否 | 无 |
| var | 存储格式化数字的变量 | 否 | Print to page |
| scope | var属性的作用域 | 否 | page |

* + 1. fmt:parseNumber

|  |  |  |  |
| --- | --- | --- | --- |
| **属性** | **描述** | **是否必要** | **默认值** |
| value | 要解析的数字 | 否 | Body |
| type | NUMBER,，CURRENCY，或 PERCENT | 否 | number |
| parseLocale | 解析数字时所用的区域 | 否 | 默认区域 |
| integerOnly | 是否只解析整型数（true）或浮点数（false） | 否 | FALSE |
| pattern | 自定义解析模式 | 否 | 无 |
| timeZone | 要显示的日期的时区 | 否 | 默认时区 |
| var | 存储待解析数字的变量 | 否 | Print to page |
| scope | var属性的作用域 | 否 | page |

使用方式如下:

|  |
| --- |
| <%@ ***page*** import="**java.util.\***" %> <%@ ***page*** contentType="**text/html;charset=UTF-8**" language="**java**" %> <%@ ***taglib*** prefix="**c**" uri="**http://java.sun.com/jsp/jstl/core**" %> <%@ ***taglib*** prefix="**fmt**" uri="**http://java.sun.com/jsp/jstl/fmt**" %> <%@ ***taglib*** prefix="**fn**" uri="**http://java.sun.com/jsp/jstl/functions**" %> **<*html*> <*body*> *<%  Date date1* = *new* Date();  *Double num1* = 12345*.4567*;  *pageContext*.setAttribute("date1",*date1*);  *pageContext*.setAttribute("num1",*num1*);  *%>* <*p*>**日期格式化 (1): <***fmt:formatDate*** type**="time"** value**="*${date1}*"** />**</*p*> <*p*>**日期格式化 (2): <***fmt:formatDate*** type**="date"** value**="*${date1}*"** />**</*p*> <*p*>**日期格式化 (3): <***fmt:formatDate*** type**="both"** value**="*${date1}*"** />**</*p*> <*p*>**日期格式化 (4): <***fmt:formatDate*** type**="both"** dateStyle**="short"** timeStyle**="short"** value**="*${date1}*"** />**</*p*> <*p*>**日期格式化 (5): <***fmt:formatDate*** type**="both"** dateStyle**="medium"** timeStyle**="medium"** value**="*${date1}*"** />**</*p*> <*p*>**日期格式化 (6): <***fmt:formatDate*** type**="both"** dateStyle**="long"** timeStyle**="long"** value**="*${date1}*"** />**</*p*> <*p*>**日期格式化 (7): <***fmt:formatDate*** pattern**="yyyy-MM-dd"** value**="*${date1}*"** />**</*p*> <*hr*> <*p*>**格式化数字 (1): <***fmt:formatNumber*** value**="*${num1}*"** type**="currency"**/>**</*p*> <*p*>**格式化数字 (2): <***fmt:formatNumber*** type**="number"** maxIntegerDigits**="3"** value**="*${num1}*"** />**</*p*> <*p*>**格式化数字 (3): <***fmt:formatNumber*** type**="number"** maxFractionDigits**="3"** value**="*${num1}*"** />**</*p*> <*p*>**格式化数字 (4): <***fmt:formatNumber*** type**="number"** groupingUsed**="false"** value**="*${num1}*"** />**</*p*> <*p*>**格式化数字 (5): <***fmt:formatNumber*** type**="percent"** maxIntegerDigits**="3"** value**="*${num1}*"** />**</*p*> <*p*>**格式化数字 (6): <***fmt:formatNumber*** type**="percent"** minFractionDigits**="10"** value**="*${num1}*"** />**</*p*> <*p*>**格式化数字 (7): <***fmt:formatNumber*** type**="percent"** maxIntegerDigits**="3"** value**="*${num1}*"** />**</*p*> <*p*>**格式化数字 (8): <***fmt:formatNumber*** type**="number"** value**="*${num1}*"** pattern**="0.00‰"**/>**</*p*> <*p*>**格式化数字 (9): <***fmt:formatNumber*** type**="number"** value**="*${num1}*"** pattern**="0.00%"**/>**</*p*> <*p*>**美元 :  <***fmt:setLocale*** value**="en\_US"**/>  <***fmt:formatNumber*** value**="*${num1}*"** type**="currency"**/>**</*p*> </*body*>  <*hr*>** <***fmt:parseNumber*** var**="i"** type**="number"** value**="*${num1}*"** /> **<*p*>**数字解析 (1) : <***c:out*** value**="*${i}*"** />**</*p*>** <***fmt:parseNumber*** var**="i"** integerOnly**="true"** type**="number"** value**="*${num1}*"** /> **<*p*>**数字解析 (2) : <***c:out*** value**="*${i}*"** />**</*p*>**  **</*html*>** |

使用效果如下:



* 1. JSTL函数库(了解)
     1. 什么EL函数库

EL函数库是由第三方对EL的扩展，我们现在学习的EL函数库是由JSTL添加的。下面我们会学习JSTL标签库。

EL函数库就是定义一些有返回值的静态方法。然后通过EL语言来调用它们！当然，不只是JSTL可以定义EL函数库，我们也可以自定义EL函数库。

EL函数库中包含了很多对字符串的操作方法，以及对集合对象的操作。例如：${fn:length(“abc”)}会输出3，即字符串的长度。

* + 1. 导入函数库

因为是第三方的东西，所以需要导入。导入需要使用taglib指令！

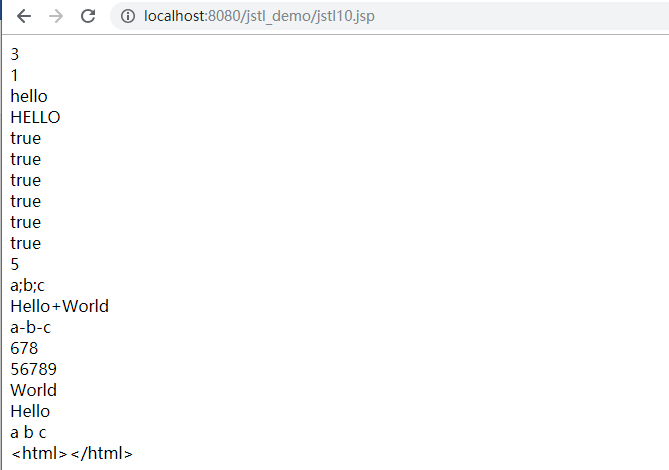
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

* + 1. EL函数库介绍
* String toUpperCase(String input)：
* String toLowerCase(String input)：
* int indexOf(String input, String substring)：
* boolean contains(String input, String substring)：
* boolean containsIgnoreCase(String input, String substring)：
* boolean startsWith(String input, String substring)：
* boolean endsWith(String input, String substring)：
* String substring(String input, int beginIndex, int endIndex)：
* String substringAfter(String input, String substring)：
* substringBefore(String input, String substring)：
* String escapeXml(String input)：
* String trim(String input)：
* String replace(String input, String substringBefore, String substringAfter)：
* String[] split(String input, String delimiters)：
* int length(Object obj)：
* String join(String array[], String separator)：

使用方式:

|  |
| --- |
| ***<%  String*[] *strs* = {"a", "b","c"};** List ***list* = *new* ArrayList();  *list*.add("a");  *pageContext*.setAttribute("arr", *strs*);  *pageContext*.setAttribute("list", *list*);  *%>  ${fn***:***length*(*arr*) *}*<*br*/>**<!--3--> ***${fn***:***length*(*list*) *}*<*br*/>**<!--1--> ***${fn***:***toLowerCase*("Hello") *}*<*br*/>** <!-- hello --> ***${fn***:***toUpperCase*("Hello") *}*<*br*/>** <!-- HELLO --> ***${fn***:***contains*("abc", "a")*}*<*br*/>**<!-- true --> ***${fn***:***containsIgnoreCase*("abc", "Ab")*}*<*br*/>**<!-- true --> ***${fn***:***contains*(*arr*, "a")*}*<*br*/>**<!-- true --> ***${fn***:***containsIgnoreCase*(*list*, "A")*}*<*br*/>**<!-- true --> ***${fn***:***endsWith*("Hello.java", ".java")*}*<*br*/>**<!-- true --> ***${fn***:***startsWith*("Hello.java", "Hell")*}*<*br*/>**<!-- true --> ***${fn***:***indexOf*("Hello-World", "-")*}*<*br*/>**<!-- 5 --> ***${fn***:***join*(*arr*, ";")*}*<*br*/>**<!-- a;b;c --> ***${fn***:***replace*("Hello-World", "-", "+")*}*<*br*/>**<!-- Hello+World --> ***${fn***:***join*(*fn***:***split*("a;b;c;", ";"), "-")*}*<*br*/>**<!-- a-b-c -->  ***${fn***:***substring*("0123456789", *6*, *9*)*}*<*br*/>**<!-- 678 --> ***${fn***:***substring*("0123456789", *5*,** -***1*)*}*<*br*/>**<!-- 56789 --> ***${fn***:***substringAfter*("Hello-World", "-")*}*<*br*/>**<!-- World --> ***${fn***:***substringBefore*("Hello-World", "-")*}*<*br*/>**<!-- Hello --> ***${fn***:***trim*(" a b c ")*}*<*br*/>**<!-- a b c --> ***${fn***:***escapeXml*("<html></html>")*}*<*br*/>** <!-- <html></html> --> |

展示效果:



1. 学习目标

掌握EL表达式的运算,操作对象和内置对象的使用;

掌握JSTL标签的核心标签库常用标签,并能使用JSTL格式化标签库处理相关格式化操作;

了解JSTL函数库相关操作;

能够使用EL表达式和JSTL标签库完成一张用户表的增删改查;