**EASY框架标签&组件使用文档**

**东方国信-电信事业部**

|  |  |  |
| --- | --- | --- |
| 名称 | 时间 | 说明 |
| 张凤达 | 2012年11月 | 编写 |
| 陈福权 | 2013年3月 | 更新 |

目录

[1. 框架介绍 2](#_Toc350764347)

[1.1. easy配置说明 2](#_Toc350764348)

[1.1.1. web.xml文件配置 2](#_Toc350764349)

[1.1.2. easyDataSourceType数据库连接池类型 2](#_Toc350764350)

[1.1.3. easyDataSource数据源名称 3](#_Toc350764351)

[1.1.4. easyDB数据库类型 3](#_Toc350764352)

[1.1.5. easyTableUsers数据库用户信息 3](#_Toc350764353)

[1.1.6. 配置例子 3](#_Toc350764354)

[1.1.7. JSP页面应用 4](#_Toc350764355)

[2. Expression Language 4](#_Toc350764356)

[2.1. EL简介 4](#_Toc350764359)

[2.2. EL语法 4](#_Toc350764360)

[2.2.1 .与 [ ] 运算符 5](#_Toc350764361)

[2.2.2 EL 变量 6](#_Toc350764362)

[2.2.3自动转变类型 7](#_Toc350764363)

[2.2.4 EL 保留字 9](#_Toc350764364)

[2.3. EL隐含对象 10](#_Toc350764365)

[2.3.1属性(Attribute)与范围(Scope) 11](#_Toc350764366)

[2.3.2与输入有关的隐含对象 12](#_Toc350764367)

[2.3.3其他隐含对象 12](#_Toc350764368)

[2.4. EL算术运算符 14](#_Toc350764369)

[2.5. EL关系运算符 15](#_Toc350764370)

[2.6. EL逻辑运算符 17](#_Toc350764371)

[2.7. EL其他运算符 17](#_Toc350764372)

[2.7.1 Empty 运算符 18](#_Toc350764373)

[2.7.2 条件运算符 18](#_Toc350764374)

[2.7.3 括号运算符 18](#_Toc350764375)

[3. EASY标签库 19](#_Toc350764376)

[3.1. SQL标签 19](#_Toc350764380)

[3.1.1. <e:q4l> 19](#_Toc350764385)

[3.1.2. <e:q4o> 20](#_Toc350764386)

[3.1.3. <e:update> 20](#_Toc350764387)

[3.2. HTML标签 21](#_Toc350764388)

[3.2.1. <e:table> 21](#_Toc350764390)

[3.2.2. <e:td> 21](#_Toc350764391)

[3.2.3. <e:pageController> 22](#_Toc350764392)

[3.2.4. <e:select> 22](#_Toc350764393)

[3.2.5. <e:checkbox> 23](#_Toc350764394)

[3.2.6. <e:radio> 23](#_Toc350764395)

[3.2.7. <e:url> 24](#_Toc350764396)

[3.2.8. <e:style> 24](#_Toc350764397)

[3.2.9. <e:script> 24](#_Toc350764398)

[3.3. Logic标签 25](#_Toc350764399)

[3.3.1. <e:if> 25](#_Toc350764406)

[3.3.2. <e:else> 25](#_Toc350764407)

[3.3.3. <e:forEach> 26](#_Toc350764408)

[3.3.4. <e:switch> 26](#_Toc350764409)

[3.3.5. <e:case> 26](#_Toc350764410)

[3.3.6. <e:set> 27](#_Toc350764411)

[3.3.7. <e:invoke> 28](#_Toc350764412)

[3.3.8. <e:param> 28](#_Toc350764413)

[3.3.9. <e:description> 29](#_Toc350764414)

[3.4. 文件系统操作标签 29](#_Toc350764415)

[3.4.1. <e:mkdir> 29](#_Toc350764417)

[3.4.2. <e:delete> 29](#_Toc350764418)

[3.4.3. <e:copy> 30](#_Toc350764419)

[3.4.4. <e:ls> 30](#_Toc350764420)

[3.5. 组件标签 31](#_Toc350764421)

[3.5.1. 标准组件标签 31](#_Toc350764430)

[3.5.2. 高级组件标签 31](#_Toc350764431)

[3.5.3. 应用组件标签 45](#_Toc350764432)

[3.6 Functions 47](#_Toc350764433)

[4. filter过滤器 48](#_Toc350764434)

[4.1. LoginFilter登录验证过滤器 48](#_Toc350764439)

[4.2. CharacterEncodingFilter转码过滤器 49](#_Toc350764440)

[5. EASY MVC 50](#_Toc350764441)

[5.1. ActionServlet中央控制器 50](#_Toc350764447)

# 框架介绍

EASY由标签库和MVC框架两部分功能组成。标签包含e标签和c标签，e标签有数据库SQL标签、HTML标签、逻辑标签、文件系统操作标签。C标签有图形标签、表格标签、样式标签。框架共有easy-1.x-yyyymmmdd.jar和easy-component-1.1yyyymmdd.jar2个文件。框架运行依赖于commons-logging.jar、commons-beanutils.jar、commons-dbutils.jar、paranamer.jar、commons-io-2.4.jar、ezmorph-1.0.6.jar、commons-collections-3.2.1.jar、commons-fileupload-1.2.1.jar，commons-jexl-2.1.1.jar类包。

# EASY标签库



## SQL标签



### <e:q4l>

此标签用于数据库查询操作，返回结果为List列表。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| var | 存储查询结果,具体参数有（epageNumberName页码名称,通常为变量名\_epageNumber，epageNumber页码，erecordCount 结果集大小，sqlParam sql参数，list 结果集） | N | String | 是 | 无 |
| pageSize | 数据容量，用于翻页 | N | int | 否 |  |

实例一：

<e:q4l var="userList" pageSize="3">

select \* from \_user where userId < #id#

</e:q4l>

实例二：

<e:q4l var="userList">

select \* from \_user where userId < #id#

</e:q4l>

### <e:q4o>

此标签用于数据库查询操作，返回结果为Map对象。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| var | 存储查询结果 | N | String | 是 | 无 |

实例一：

<e:q4o var=*"indicatorExt"*>

select \* from kpi\_indicator\_ext where indicator\_ext\_no = #extNo#

</e:q4o>

### <e:update>

此标签用于对数据库进行更新操作，这些更新主要是INSERT、UPDATE、DELETE语句。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| var | 存储改变的数据笔数 | N | String | 否 | 无 |

实例一：

执行单一操作。

<e:update>

update ${KPI\_ETU}.kpi\_indicator\_ext

set indicator\_ext\_name = #extName#,

unit\_no = #unit#,

indicator\_ext\_expression = upper(#indicatorExtExpression#)

where indicator\_ext\_no = #extNo#

</e:update>

实例二：

执行多个操作。

<e:update>

begin

<e:forEach items="${paramValues.selId}" var=*"item"*>

delete from kpi\_indicator where indicator\_no = '${item}';

delete from kpi\_indicator\_ext where indicator\_no = '${item}';

delete from kpi\_indicator\_dim\_rel where indicator\_no = '${item}';

delete from kpi\_indicator\_constraint where indicator\_no = '${item}';

delete from kpi\_indicator\_instance where indicator\_no = '${item}';

</e:forEach>

end;

</e:update>

## HTML标签



### <e:table>

此标签用于输出HTML中的表格Table。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| var | 用来存放现在指到的成员 | N | String | 是 | 无 |
| items | 被迭代的集合对象 | Y | Collection  Iterator | 是 | 无 |
| 动态属性 | HTML中Table的属性 | Y | String | 否 | 无 |

### <e:td>

此标签用于输出HTML中的表格的tr和td。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| title | 本列的表头 | Y | String | 否 | 无 |
| property | td中的内容 | Y | String | 否 | 无 |
| 动态属性 | HTML中td的属性 | Y | String | 否 | 无 |

### <e:pageController>

此标签用于输出分页控件，配合<e:q4l>使用，在配合<e:q4l>使用时，<e:q4l>的pageSize属性必需大于0。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| action | 当点击翻页时，数据提交的路径。 | Y | String | 否 | 本页地址 |
| queryObject | 查询对象 | Y | Object | 是 | 无 |
| 动态属性 | HTML中div的属性 | Y | String | 否 | 无 |

### <e:select>

此标签用于输出HTML中的select。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| items | 被迭代的集合对象 | Y | Collection  Iterator | 是 | 无 |
| defaultValue | select默认的选择值 | Y | String | 否 | 无 |
| headValue | Select中首<option>存储的值 | Y | String | 否 | 无 |
| headLabel | Select中首<option>显示的内容 | Y | String | 否 | 无 |
| value | <option>存储的值 | Y | String | 是 | 无 |
| label | <option>显示的内容 | Y | String | 是 | 无 |
| 动态属性 | HTML中select的属性 | Y | String | 否 | 无 |

### <e:checkbox>

此标签用于输出HTML中的checkbox。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| defaultValue | checkbox的默认值 | Y | String | 否 | 无 |
| defaultValues | checkbox的默认值集合，每个值之间用逗号“，”来分隔。例如defaultValues=”1,2,3,4”,这说明checkbox的默认值可以是“1,2,3,4”其中任何一个，只要有一个符合，checkbox的状态就为选中状态。当defaultValue和defaultValues同时出现时以defaultValue为主，defaultValues失效。 | Y | String | 否 | 无 |
| value | checkbox存储的值 | Y | String | 否 | 无 |
| label | checkbox显示的值 | Y | String | 否 | 无 |
| 动态属性 | HTML中checkbox的属性 | Y | String | 否 | 无 |

### <e:radio>

此标签用于输出HTML中的radio单选框。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| defaultValue | radio的默认值 | Y | String | 否 | 无 |
| defaultValues | radio的默认值集合，每个值之间用逗号“，”来分隔。例如defaultValues=”1,2,3,4”,这说明checkbox的默认值可以是“1,2,3,4”其中任何一个，只要有一个符合，checkbox的状态就为选中状态。当defaultValue和defaultValues同时出现时以defaultValue为主，defaultValues失效。 | Y | String | 否 | 无 |
| value | radio存储的值 | Y | String | 否 | 无 |
| label | radio显示的值 | Y | String | 否 | 无 |
| 动态属性 | HTML中radio的属性 | Y | String | 否 | 无 |

### <e:url>

此标签用于补全指定文件在web应用中的绝对路径。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| value | 目标文件 | Y | String | 是 | 无 |
| system | 如果为true输入系统的全路径，如果为false输出相对web应用的全路径。 | N | boolean | 是 | false |

### <e:style>

此标签用于输出HTML中的CSS文件引用标签<link href="样式.css" rel="stylesheet" type="text/css" />。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| value | 在web应用中的绝对路径下的目标文件 | Y | String | 是 | 无 |

### <e:script>

此标签用于输出HTML中的javascript文件引用标签<script type="text/javascript" src="javascript文件.js"></script>。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| value | 在web应用中的绝对路径下的目标文件 | Y | String | 是 | 无 |

## Logic标签



### <e:if>

此标签用于进行条件判断，如果它的condition属性为true，那么就计算它的Body。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| var | 用来储存condition运算后的结果，即true 或false | N | String | 否 | 无 |
| condition | 表达式的条件，如果表达式的结果为true，则执行本体内容。 | Y | boolean | 是 | 无 |

实例一

<e:if var=*"test"* condition="${1==2}">

如果条件成立执行此段代码

</e:if>

### <e:else>

此标签配合if标签使用，当if标签condition属性为false，那么就计算它的Body。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| condition | if标签的返回值 | Y | boolean | 是 | 无 |

<e:if var=*"test"* condition="${1==2}">

如果条件成立执行此段代码

</e:if>

<e:else condition="${test}">

否则执行此段代码

</e:else>

### <e:forEach>

此标签用于迭代操作。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| var | 用来存放现在指到的成员 | N | String | 是 | 无 |
| items | 被迭代的集合对象,当集合对象为Map类型时，将迭代Map的每个键值对，键名为key，值名为value | Y | Collection  Iterator  Map | 是 | 无 |

实例一：

<e:forEach>标签有一个子标签<e:break/>，用来终止循环，注意终止是从下次循环开始，并不终止此次循环。

<e:forEach items="${user\_list}" var="item">

<e:if condition="${item == '2'}">

<e:break/>

</e:if>

</e:forEach>

实例二：

迭代对象dynattrs为Map类型，那么Map对象中的每个键值对将转换成一个名为dynattr的对象，键名将存放在dynattr的key中，键值将存放在dynattr的value中。

<e:forEach items="${dynattrs}" var=*"dynattr"*>

${dynattr.key}="${dynattr.value}"

</e:forEach>

### <e:switch>

此标签用于多分支选择，常用来根据表达式的值选择要执行的语句。和<e:case>配合使用。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| value | 表达式的值 | Y | String | 是 | 无 |

### <e:case>

此标签用于分支条件判断, 和<e:switch>配合使用。如果它的value属性值等于父标签<e:switch>的value属性值，那么就计算它的Body。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| value | 要进行比较的值 | Y | String | 是 | 无 |

多分支选择例子：

<e:switch value="{param.action}">

<e:case value="aaa">

分之一

</e:case>

<e:case value="bbb">

分支二

</e:case>

<e:case value="ccc">

分支三

</e:case>

</e:switch>

### <e:set>

此标签用于将变量存储至JSP的作用域中。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| var | 用来存放现在指到的成员（用于设置作用域属性） | N | String | 是 | 无 |
| value | 存储的对象 | Y | Object | 否 | 无 |
| clazz | 如果要实例化一个类的对象，那么就使用clazz属性，此属性存储要实例化类的全名。 | N | String | 否 | 无 |
| scope | 设置变量的作用域，取值为page、request、session、application | N | String | 否 | page |

变量存储有三种方式，第一种是存储value中的值，第二种是存储clazz指定类实例化的对象，第三种是存储标签体的内容，具体如下

1. 存储value属性中的值：

<e:set var="replaceStr" value="\n"/>

1. 存储clazz指定类实例化的对象

<e:set var="user\_list" clazz="java.util.ArrayList"/>

1. 存储标签体的内容：

<e:set var="sql">

Insert into table values()

</e:set>

### <e:invoke>

此标签用于对象的方法调用。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| var | 用来存放方法的返回值 | Y | Object | 否 | 无 |
| object | 操作的对象 | Y | Object | 是 | 无 |
| method | 要调用的方法名 | Y | String | 是 | 无 |

<e:set var="user\_list" clazz="java.util.ArrayList"/>

<e:invoke object="${user\_list}" method="add">

<e:param>abc</e:param>

</e:invoke>

<e:set var="user" value="zfd" />

<e:set var="user\_list" clazz="java.util.ArrayList" />

<e:invoke object="${user\_list}" method="add">

<e:param value="${zfd}" />

</e:invoke>

### <e:param>

此标签用于向父标签中添加参数。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| value | 参数值 | Y | Object | 否 | 无 |

参数设置有两种方式，一种是设置value中的值，一种是设置标签体的内容，具体如下

1. 设置value属性中的值：

<e:param value="${zfd}"/>

<e:param value="zfd"/>

1. 设置标签体的内容：

<e:param>

Insert into table values

</e:param>

### <e:description>

此标签用于程序中代码注释。

<e:description>单行注释</e:description>

<e:description>

多行注释

</e:description>

## 文件系统操作标签



### <e:mkdir>

此标签用于创建单级目录和多级目录，它非常类似于具有相同名称的 Windows 和 UNIX/Linux 命令。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| var | 存储创建结果。 | N | String | 否 | 无 |
| dir | 需要创建的目录名称，注：此路径是相对web应用的根目录。 | Y | String | 是 | 无 |

1. 创建单级目录：

<e:mkdir dir="com" />

1. 创建多级目录：

<e:mkdir dir="cn/com/bonc" />

### <e:delete>

此标签用于删除指定的目录和文件，当删除目录的时候，会连同目录包含的所有文件以及子目录一并删除。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| var | 存储删除结果。 | N | String | 否 | 无 |
| file | 需要删除的目录或文件名称。注：此路径是相对web应用的根目录。 | Y | String | 是 | 无 |

1. 删除目录：

<e:delete file="cn/com/bonc" />

1. 删除文件：

<e:delete file="cn/com/bonc/logo.gif" />

### <e:copy>

此标签用于文件拷贝。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| var | 存储拷贝结果。 | N | String | 否 | 无 |
| file | 需要拷贝的源文件。注：此路径是相对web应用的根目录。 | Y | String | 是 | 无 |
| tofile | 需要拷贝的目标文件。注：此路径是相对web应用的根目录 | Y | String | 是 | 无 |

1. 拷贝文件：

<e:copy file=" cn/com/bonc/logo.gif" tofile=" cn/com/bonc/logo2.gif" />

<e:copy file=" cn/com/bonc/logo.gif" tofile=" cn/logo.gif" />

### <e:ls>

此标签用于列出目录下的所有文件。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| var | 存储查询的结果。 | N | String | 否 | 无 |
| dir | 需要查询的目录。注：此路径是相对web应用的根目录。 | Y | String | 是 | 无 |
| suffix | 文件扩展名，如果需要按照文件类型来查找文件的话，那么就要使用该属性。（当属性值为dir时表示只显示目录） | N | String | 是 | 无 |

1. 列出子目录：

<e:ls var="flies" dir="/cn" suffix="dir" />

1. 列出gif图片：

<e:ls var="flies" dir="/cn" suffix=".gif" />

1. 列出所有文件：

<e:ls var="flies" dir="/cn" />

## 组件标签



### 标准组件标签

#### <c:pie>

此标签用于打印饼形图。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| id | 图形组建id | N | String | 是 | 无 |
| width | 图形宽度 | N | String | 是 | 无 |
| height | 图形高度 | N | String | 是 | 无 |
| items | 需要图形化的数据集 | Y | Collection  Iterator | 是 | 无 |
| dimension | 数据集的维度列 | N | String | 是 | 无 |
| value | 数据集的数据列 | N | String | 是 | 无 |
| title | 饼形图的标题 | N | String | 否 | 无 |
| unit | 数据单位 | N | String | 否 | 无 |
| backgroundColor | 图形背景颜色 | N | String | 否 | 无 |
| borderWidth | 图形边框宽度 | N | String | 否 | 0 |
| dataLabels | 是否显示数据信息 | N | String | 否 | false |
| legend | 是否显示图例 | N | String | 否 | false |
| percentage | 是否显示百分比 | N | String | 否 | false |
| defaultValue | 饼图中默认选中的值 | N | String | 否 | 无 |

实例一：

<e:q4l var="dev\_pie">

select t2.SVC\_NAME x,

t.tele\_type ,

sum(t.dev\_users) v

from ${dm }.dm\_cnl\_channel\_dev\_m t,

${dmcode}.code\_tele\_type t2

where t.tele\_type = t2.SVC\_ID

group by t2.SVC\_NAME,t.tele\_type

</e:q4l>

<c:pie legend="true" dimension="x" height="320px" id="zfd4" items="${dev\_pie.list}" value="v" width="400" unit="户"/>

#### <c:column>

此标签用于打印柱形图。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| id | 图形组建id | N | String | 是 | 无 |
| width | 图形宽度 | N | String | 是 | 无 |
| height | 图形高度 | N | String | 是 | 无 |
| items | 需要图形化的数据集 | Y | Collection  Iterator | 是 | 无 |
| indicator | 数据集的指标列 | N | String | 是 | 无 |
| dimension | 数据集的维度列 | N | String | 是 | 无 |
| value | 数据集的数据列 | N | String | 是 | 无 |
| title | 图形标题 | N | String | 否 | 无 |
| subtitle | 图形子标题 | N | String | 否 | 无 |
| ytitle | 图形y轴标题 | N | String | 否 | 无 |
| unit | 数据单位 | N | String | 否 | 无 |
| backgroundColor | 图形背景颜色 | N | String | 否 | 无 |
| borderWidth | 图形边框宽度 | N | String | 否 | 0 |
| legendBorderWidth | 图例边框宽度 | N | String | 否 | 0 |
| legend | 是否显示图例 | N | String | 否 | true |
| stacking | 是否堆叠显示 | N | String | 否 | false |

实例一：

<e:q4l var="dev\_col">

select t2.SVC\_NAME x,

t.tele\_type ,

sum(t.dev\_users) v

from ${dm }.dm\_cnl\_channel\_dev\_m t,

${dmcode}.code\_tele\_type t2

where t.tele\_type = t2.SVC\_ID

group by t2.SVC\_NAME,t.tele\_type

</e:q4l>

<c:column legend="true" dimension="x" height="320px" id="col1" items="${dev\_col.list}" value="v" width="400" unit="户"/>

#### <c:line>

此标签用于打印线形图。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| id | 图形组建id | N | String | 是 | 无 |
| width | 图形宽度 | N | String | 是 | 无 |
| height | 图形高度 | N | String | 是 | 无 |
| items | 需要图形化的数据集 | Y | Collection  Iterator | 是 | 无 |
| indicator | 数据集的指标列 | N | String | 是 | 无 |
| dimension | 数据集的维度列 | N | String | 是 | 无 |
| value | 数据集的数据列 | N | String | 是 | 无 |
| title | 图形标题 | N | String | 否 | 无 |
| subtitle | 图形子标题 | N | String | 否 | 无 |
| ytitle | 图形y轴标题 | N | String | 否 | 无 |
| unit | 数据单位 | N | String | 否 | 无 |
| backgroundColor | 图形背景颜色 | N | String | 否 | 无 |
| borderWidth | 图形边框宽度 | N | String | 否 | 0 |
| legendBorderWidth | 图例边框宽度 | N | String | 否 | 0 |
| legend | 是否显示图例 | N | String | 否 | true |
| crosshairs | 是否显示准线 | N | String | 否 | false |

实例1:

<e:q4l var="brokerage\_msg\_line">

SELECT '出账收入' ind,

ACCT\_MONTH acct\_month,

DECODE(SUM(ACCOUNT\_INC), NULL, '--', ROUND(SUM(ACCOUNT\_INC), 2)) va

FROM ${dm }.DM\_CNL\_BROKERAGE\_MSG\_M

GROUP BY ACCT\_MONTH

ORDER BY acct\_month

</e:q4l>

<c:line legendBorderWidth="1" dimension="acct\_month" height="250px" id="line1" indicator="ind" items="${brokerage\_msg\_line.list}" value="va" width="400" unit="元" />

#### <c:columnLine>

此标签用于打印柱形图和线形图的组合图，并且两种图是同一个坐标轴。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| id | 图形组建id | N | String | 是 | 无 |
| width | 图形宽度 | N | String | 是 | 无 |
| height | 图形高度 | N | String | 是 | 无 |
| lineItems | 线形图的数据集 | Y | Collection  Iterator | 是 | 无 |
| columnItems | 柱形图的数据集 | Y | Collection  Iterator | 是 | 无 |
| indicator | 数据集的指标列 | N | String | 是 | 无 |
| dimension | 数据集的维度列 | N | String | 是 | 无 |
| value | 数据集的数据列 | N | String | 是 | 无 |
| title | 图形标题 | N | String | 否 | 无 |
| subtitle | 图形子标题 | N | String | 否 | 无 |
| ytitle | 图形y轴标题 | N | String | 否 | 无 |
| unit | 数据单位 | N | String | 否 | 无 |
| backgroundColor | 图形背景颜色 | N | String | 否 | 无 |
| borderWidth | 图形边框宽度 | N | String | 否 | 0 |
| legendBorderWidth | 图例边框宽度 | N | String | 否 | 0 |
| legend | 是否显示图例 | N | String | 否 | true |

实例1：

<e:q4l var="line">

SELECT '出账收入' ind,

ACCT\_MONTH acct\_month,

DECODE(SUM(ACCOUNT\_INC), NULL, '--', ROUND(SUM(ACCOUNT\_INC), 2)) va

FROM ${dm }.DM\_CNL\_BROKERAGE\_MSG\_M

GROUP BY ACCT\_MONTH

ORDER BY acct\_month

</e:q4l>

<e:q4l var="col">

SELECT '出账收入' ind,

ACCT\_MONTH acct\_month,

DECODE(SUM(ACCOUNT\_INC), NULL, '--', ROUND(SUM(ACCOUNT\_INC), 2)) va

FROM ${dm }.DM\_CNL\_BROKERAGE\_MSG\_M

GROUP BY ACCT\_MONTH

ORDER BY acct\_month

</e:q4l>

<c:columnLine legendBorderWidth="1" dimension="acct\_month"

height="250px" id="line1" indicator="ind"

lineItems ="${line.list}" columnItems="${col.list}" value="va" width="400" unit="元" />

#### <c:columnLine2>

此标签用于打印柱形图和线形图的组合图，并且两种图是不同的坐标轴。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| id | 图形组建id | N | String | 是 | 无 |
| width | 图形宽度 | N | String | 是 | 无 |
| height | 图形高度 | N | String | 是 | 无 |
| lineItems | 线形图的数据集 | Y | Collection  Iterator | 是 | 无 |
| columnItems | 柱形图的数据集 | Y | Collection  Iterator | 是 | 无 |
| indicator | 数据集的指标列 | N | String | 是 | 无 |
| dimension | 数据集的维度列 | N | String | 是 | 无 |
| value | 数据集的数据列 | N | String | 是 | 无 |
| title | 图形标题 | N | String | 否 | 无 |
| subtitle | 图形子标题 | N | String | 否 | 无 |
| columnTitle | 柱形图y轴标题 | N | String | 否 | 无 |
| lineTitle | 线形图y轴标题 | N | String | 否 | 无 |
| columnUnit | 柱形图数据单位 | N | String | 否 | 无 |
| lineUnit | 线形图数据单位 | N | String | 否 | 无 |
| backgroundColor | 图形背景颜色 | N | String | 否 | 无 |
| borderWidth | 图形边框宽度 | N | String | 否 | 0 |
| legendBorderWidth | 图例边框宽度 | N | String | 否 | 0 |
| legend | 是否显示图例 | N | String | 否 | true |

实例1：

<e:q4l var="line">

SELECT '出账收入' ind,

ACCT\_MONTH acct\_month,

DECODE(SUM(ACCOUNT\_INC), NULL, '--', ROUND(SUM(ACCOUNT\_INC), 2)) va

FROM ${dm }.DM\_CNL\_BROKERAGE\_MSG\_M

GROUP BY ACCT\_MONTH

ORDER BY acct\_month

</e:q4l>

<e:q4l var="col">

SELECT '出账收入' ind,

ACCT\_MONTH acct\_month,

DECODE(SUM(ACCOUNT\_INC), NULL, '--', ROUND(SUM(ACCOUNT\_INC), 2)) va

FROM ${dm }.DM\_CNL\_BROKERAGE\_MSG\_M

GROUP BY ACCT\_MONTH

ORDER BY acct\_month

</e:q4l>

<c:columnLine2 legendBorderWidth="1" dimension="acct\_month"

height="250px" id="line1" indicator="ind"

lineItems ="${line.list}" columnItems="${col.list}" value="va" width="400" columnUnit ="元" lineUnit ="元" />

#### <c:datebox>

此控件显示时间控件。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| id | 日历控件id | N | String | 是 | 无 |
| name | 日历控件name | N | String | 否 | 无 |
| required | 日历控件是否可以为空 | N | String | 是 | 无 |
| defaultValue | 日历控件默认日期 | Y | String | 否 | 无 |
| isData | 是否为数据日历控件[1为是，0为否]，默认为数据开放的日期 | Y | String | 否 | 无 |

实例1：

<c:datebox id="begin\_date" name="begin\_date" required="true" isData="1"/>

#### <c:month>

此标签显示月份账期

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| id | 月份控件id | N | String | 是 | 无 |
| name | 月份控件name | N | String | 否 | 无 |
| label | 月份控件前边的描述名称 | N | String | 是 | 无 |
| defaultValue | 月份控件默认日期 | Y | String | 否 | 无 |

<c:month id="acct\_month" name="acct\_month" label="帐期" defaultValue="201202"/>

#### <c: table>

生成表格标签

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| id | 表头id | N | String | 是 | 无 |
| url | 数据路径 | N | String | 是 | 无 |
| fitColumns | 列自动适合宽度 | N | String | 否 | 无 |
| pagination | 是否显示分页工具栏 | N | String | 否 | 无 |
| rownumbers | 是否显示行号 | N | String | 否 | 无 |
| singleSelect | 数据是否单选 | N | String | 否 | 无 |
| pageSize | 页面结果集大小 | N | String | 否 | 无 |
| onClickRow | 单击行事件 | N | String | 否 | 无 |
| onDblClickRow | 双击行事件 | N | String | 否 | 无 |
| onClickCell | 单击单元格事件 | N | String | 否 | 无 |
| onDblClickCell | 双击单元格事件 | N | String | 否 | 无 |
| download | 导出文件的名称，不为空时下载 | N | String | 否 | 无 |
| downArgs | 导出文件的获得sql的参数，格式为&A=33&b=44 | N | String | 否 | 无 |
| nowrap | 单元格中的内容是否不换行，true不换；false换行 | N | String | 否 | 无 |

<table id="table2" style="width:800px;height:auto;">

<thead>

<tr>

<th field="ACCT\_MONTH" width="100">账期</th>

<th field="AREA\_NO" width="100">地域编号</th>

<th field="CITY\_NO" width="100">区县编号</th>

<th field="TELE\_TYPE" width="100">业务类型</th>

<th field="CUST\_GROUP" width="200">战略分群</th>

<th field="CHANNEL\_TYPE" width="200">渠道类型</th>

</tr>

</thead>

</table>

<c:table url="/pages/example/table\_action.jsp" id="table2" />

#### <c:tablequery>

和<c:tabl>标签一起使用<c:tablequery>sql</c:tablequery>

#### <c:treetable>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| id | 月份控件id | N | String | 是 | 无 |
| idField | 主键对应列 | N | String | 是 | 无 |
| treeField | 树对应列 | N | String | 是 | 无 |
| treeFieldTitle | 树对应列的标题 | N | String | 是 | 无 |
| treeFieldWidth | 树对应列的宽度 | N | String | 是 | 无 |
| url | 数据路径 | N | String | 是 | 无 |
| menuWidth | 维度菜单宽度 | N | String | 是 | 无 |
| defaultDim | 默认选中维度 | N | String | 是 | 无 |

实例1：

1.demo1.jsp内容如下

<c:treetablesql dimField="default">

select

'-1' as ID,

'合计' as V2,

to\_char(sum(CHANNEL\_TOTAL)) as V3,

to\_char(sum(CHANNEL\_ADD)) as V4,

to\_char(sum(CHANNEL\_ACTIVE)) as V5,

to\_char(sum(CHANNEL\_NEWSILENCE)) as V6,

to\_char(sum(CHANNEL\_OLDSILENCE)) as V7,

to\_char(sum(CHANNEL\_ACTIVE)+sum(CHANNEL\_NEWSILENCE)+sum(CHANNEL\_OLDSILENCE)) as V8,

to\_char(sum(CHANNEL\_CANCEL)) as V9

from

${dm }.DM\_CNL\_CHANNEL\_BUILD\_M m

where

1=1

</c:treetablesql>

<c:treetablesql dimField="area">

select

c.city\_desc as V2,

to\_char(sum(CHANNEL\_TOTAL)) as V3,

to\_char(sum(CHANNEL\_ADD)) as V4,

to\_char(sum(CHANNEL\_ACTIVE)) as V5,

to\_char(sum(CHANNEL\_NEWSILENCE)) as V6,

to\_char(sum(CHANNEL\_OLDSILENCE)) as V7,

to\_char(SUM(channel\_active) + SUM(channel\_newsilence) + SUM(channel\_oldsilence)) AS v8,

to\_char(SUM(channel\_cancel)) AS v9

from

${dm }.DM\_CNL\_CHANNEL\_BUILD\_M m,${dss }.code\_city c

where m.area\_no=#q\_area\_no# and m.city\_no=c.city\_no

</c:treetablesql>

2.demo2.jsp

<c:treetable id="showtable" idField="ID" treeField="V2" treeFieldTitle="维度" treeFieldWidth="200" url="demo1.jsp" defaultDim="default,area" menuWidth="150">

<c:dimension label="地市" field="area" ></c:dimension>

</c:treetable>

#### <c:dimensionGroup>

此标签体现在表格下钻时的下钻维度分组，与标签<c:treetable>联合使用。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| label | 维度名称 | N | String | 是 | 无 |
| menuWidth | 维度菜单宽度 | N | String | 是 | 无 |



实例1：

1.demo1.jsp内容如下

<c:treetablesql dimField="**default**">

select

'-1' as ID,

'合计' as V2,

to\_char(sum(CHANNEL\_TOTAL)) as V3,

to\_char(sum(CHANNEL\_ADD)) as V4,

to\_char(sum(CHANNEL\_ACTIVE)) as V5,

to\_char(sum(CHANNEL\_NEWSILENCE)) as V6,

to\_char(sum(CHANNEL\_OLDSILENCE)) as V7,

to\_char(sum(CHANNEL\_ACTIVE)+sum(CHANNEL\_NEWSILENCE)+sum(CHANNEL\_OLDSILENCE)) as V8,

to\_char(sum(CHANNEL\_CANCEL)) as V9

from

${dm }.DM\_CNL\_CHANNEL\_BUILD\_M m

where

1=1

</c:treetablesql>

<c:treetablesql dimField="**area**">

select

c.city\_desc as V2,

to\_char(sum(CHANNEL\_TOTAL)) as V3,

to\_char(sum(CHANNEL\_ADD)) as V4,

to\_char(sum(CHANNEL\_ACTIVE)) as V5,

to\_char(sum(CHANNEL\_NEWSILENCE)) as V6,

to\_char(sum(CHANNEL\_OLDSILENCE)) as V7,

to\_char(SUM(channel\_active) + SUM(channel\_newsilence) + SUM(channel\_oldsilence)) AS v8,

to\_char(SUM(channel\_cancel)) AS v9

from

${dm }.DM\_CNL\_CHANNEL\_BUILD\_M m,${dss }.code\_city c

where m.area\_no=#q\_area\_no# and m.city\_no=c.city\_no

</c:treetablesql>

2.demo2.jsp

<c:treetable id="showtable" idField="ID" treeField="V2" treeFieldTitle="维度" treeFieldWidth="200" url="demo1.jsp" defaultDim="**default**,**area**" menuWidth="150">

**<c:dimensionGroup label="地域" menuWidth="200">**

<c:dimension label="地市" field="**area**"></c:dimension>

<c:dimension label="区县" field="**city**"></c:dimension>

**</c:dimensionGroup>**

</c:treetable>

#### <c:dimension>

此标签体现在表格下钻时的下钻维度，与标签<c:treetable>联合使用。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| label | 维度名称 | N | String | 是 | 无 |
| field | 维度类型 | N | String | 是 | 无 |
| group | 维度组名 | N | String | 是 | 无 |
| level | 维度级别 | N | String | 是 | 无 |



实例1：

1.demo1.jsp内容如下

<c:treetablesql dimField="**default**">

select

'-1' as ID,

'合计' as V2,

to\_char(sum(CHANNEL\_TOTAL)) as V3,

to\_char(sum(CHANNEL\_ADD)) as V4,

to\_char(sum(CHANNEL\_ACTIVE)) as V5,

to\_char(sum(CHANNEL\_NEWSILENCE)) as V6,

to\_char(sum(CHANNEL\_OLDSILENCE)) as V7,

to\_char(sum(CHANNEL\_ACTIVE)+sum(CHANNEL\_NEWSILENCE)+sum(CHANNEL\_OLDSILENCE)) as V8,

to\_char(sum(CHANNEL\_CANCEL)) as V9

from

${dm }.DM\_CNL\_CHANNEL\_BUILD\_M m

where

1=1

</c:treetablesql>

<c:treetablesql dimField="**area**">

select

c.city\_desc as V2,

to\_char(sum(CHANNEL\_TOTAL)) as V3,

to\_char(sum(CHANNEL\_ADD)) as V4,

to\_char(sum(CHANNEL\_ACTIVE)) as V5,

to\_char(sum(CHANNEL\_NEWSILENCE)) as V6,

to\_char(sum(CHANNEL\_OLDSILENCE)) as V7,

to\_char(SUM(channel\_active) + SUM(channel\_newsilence) + SUM(channel\_oldsilence)) AS v8,

to\_char(SUM(channel\_cancel)) AS v9

from

${dm }.DM\_CNL\_CHANNEL\_BUILD\_M m,${dss }.code\_city c

where m.area\_no=#q\_area\_no# and m.city\_no=c.city\_no

</c:treetablesql>

2.demo2.jsp

<c:treetable id="showtable" idField="ID" treeField="V2" treeFieldTitle="维度" treeFieldWidth="200" url="demo1.jsp" defaultDim="**default**,**area**" menuWidth="150">

**<c:dimension label="地市" field="area" ></c:dimension>**

</c:treetable>

#### <c:treetablesql>

主要实现封装下钻数据集（数据SQL），与标签<c:treetable>联合使用。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| dimField | 维度类型 | N | String | 是 | 无 |
| isiterate | 维度是否迭代 | N | String | 是 | 无 |

实例1：

1.demo1.jsp内容如下

**<c:treetablesql dimField="default">**

select

'-1' as ID,

'合计' as V2,

to\_char(sum(CHANNEL\_TOTAL)) as V3,

to\_char(sum(CHANNEL\_ADD)) as V4,

to\_char(sum(CHANNEL\_ACTIVE)) as V5,

to\_char(sum(CHANNEL\_NEWSILENCE)) as V6,

to\_char(sum(CHANNEL\_OLDSILENCE)) as V7,

to\_char(sum(CHANNEL\_ACTIVE)+sum(CHANNEL\_NEWSILENCE)+sum(CHANNEL\_OLDSILENCE)) as V8,

to\_char(sum(CHANNEL\_CANCEL)) as V9

from

${dm }.DM\_CNL\_CHANNEL\_BUILD\_M m

where

1=1

**</c:treetablesql>**

**<c:treetablesql dimField="area">**

select

c.city\_desc as V2,

to\_char(sum(CHANNEL\_TOTAL)) as V3,

to\_char(sum(CHANNEL\_ADD)) as V4,

to\_char(sum(CHANNEL\_ACTIVE)) as V5,

to\_char(sum(CHANNEL\_NEWSILENCE)) as V6,

to\_char(sum(CHANNEL\_OLDSILENCE)) as V7,

to\_char(SUM(channel\_active) + SUM(channel\_newsilence) + SUM(channel\_oldsilence)) AS v8,

to\_char(SUM(channel\_cancel)) AS v9

from

${dm }.DM\_CNL\_CHANNEL\_BUILD\_M m,${dss }.code\_city c

where m.area\_no=#q\_area\_no# and m.city\_no=c.city\_no

**</c:treetablesql>**

2.demo2.jsp

<c:treetable id="showtable" idField="ID" treeField="V2" treeFieldTitle="维度" treeFieldWidth="200" url="demo1.jsp" defaultDim="**default**,**area**" menuWidth="150">

**<c:dimension label="地市" field="area" ></c:dimension>**

</c:treetable>

### 应用组件标签

#### <a:area>

此标签展现地市控件。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| id | 地市控件ID | N | String | 否 | 无 |
| name | 地市控件NAME | N | String | 否 | 无 |
| areaNo | 登录用户的地市权限 | N | String | 否 | 无 |
| defaultValue | 地市默认值 | N | String | 否 | 无 |

实例1：

<a:area id="area\_no" name="area\_no " areaNo ="431" />

#### <a:city>

此标签展现区县控件。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| id | 地市控件ID | N | String | 否 | 无 |
| name | 地市控件NAME | N | String | 否 | 无 |
| areaNo | 区县所属地市 | N | String | 否 | 无 |
| cityNo | 登录用户的地市权限 | N | String | 否 | 无 |
| defaultValue | 地市默认值 | N | String | 否 | 无 |

实例1：

<a:city id="city\_no" name="city\_no" areaNo ="431" cityNo="3101" />

#### <a:areaAndCity>

此标签展现地市区县联动控件。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| areaid | 地市控件ID | N | String | 否 | 无 |
| cityid | 区县控件ID | N | String | 否 | 无 |
| areaNo | 区县所属地市 | N | String | 否 | 无 |
| cityNo | 登录用户的地市权限 | N | String | 否 | 无 |

实例1：

<a:areaAndCity areaid="area\_no" cityid="city\_no"/>

#### <a:title>

此标签展现页面上的标题。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| name | 标题名称 | N | String | 否 | 无 |

实例1：

<a:title name=”测试页面名称”/>

#### <a:resources>

此标签展现页面上的标题。

属性列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| type | 引用JS文件的组类型 | N | String | 否 | 无 |

实例1：

<a: resources type=”all”/>

## 3.6 Functions

|  |  |
| --- | --- |
| **函数** | **描述** |
| e:contains(string, substring) | 如果参数string中包含参数substring，返回true |
| e:containsIgnoreCase(string, substring) | 如果参数string中包含参数substring（忽略大小写），返回true |
| e:endsWith(string, suffix) | 如果参数 string 以参数suffix结尾，返回true |
| e: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 |
| e:indexOf(string, substring) | 返回参数substring在参数string中第一次出现的位置 |
| e: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。 |
| e:length(item) | 返回参数item中包含元素的数量。参数Item类型是数组、collection、map、string。如果是String类型,返回值是String中的字符数。 |
| e:replace(string, before, after) | 返回一个String对象。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 |
| e: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 |
| e:startsWith(string, prefix) | 如果参数string以参数prefix  开头，返回true |
| e: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 |
| e:substringAfter(string, substring) | 返回参数substring在参数string中后面的那一部分字符串 |
| e:substringBefore(string, substring) | 返回参数substring在参数string中前面的那一部分字符串 |
| e:toString(object) | 将参数object转换成String类型，通常用于将oracle的CLOB类型转换成java的String类型。并将其返回 |
| e:toLowerCase(string) | 将参数string所有的字符变为小写，并将其返回 |
| e:toUpperCase(string) | 将参数string所有的字符变为大写，并将其返回 |
| e:trim(string) | 去除参数string 首尾的空格，并将其返回 |
| e:getDate(string) | 返回参数format格式的时间 |
| e:java2json（java.lang.Object） | 返回json格式的字符串 |
| e:json2java(java.lang.String) | 返回java的List或Map类型的对象 |
| e:numberFormat(java.lang.String) | 数字格式化，比如1000000.89将被格式化成1,000,000.89,注意参数必须是数字格式的字符串 |

# filter过滤器



## LoginFilter登录验证过滤器

此过滤器用于对整个系统的页面进行session登录验证时使用，验证登录的session名、验证失败返回的主页和不进行登录验证的页面都是可配置的。

filter路径：cn.com.easy.filter.LoginFilter

参数列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| homePage | 验证失败后返回的页面，通常为登录页面。 | N | String | 是 | 无 |
| session | 需要验证的session名称。 | N | String | 是 | 无 |
| excludes | 不进行登录验证的页面，可以是多个页面，中间要用逗号进行分割。 | N | String | 是 | 无 |

实例：

<filter>

<filter-name>LoginFilter</filter-name>

<filter-class>cn.com.easy.filter.LoginFilter</filter-class>

<init-param>

<param-name>homePage</param-name>

<param-value>index.jsp</param-value>

</init-param>

<init-param>

<param-name>session</param-name>

<param-value>userInfo</param-value>

</init-param>

<init-param>

<param-name>excludes</param-name>

<param-value>index.jsp,home.jsp,login.do</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>LoginFilter</filter-name>

<url-pattern>\*.do</url-pattern>

</filter-mapping>

<filter-mapping>

<filter-name>LoginFilter</filter-name>

<url-pattern>\*.jsp</url-pattern>

</filter-mapping>

## CharacterEncodingFilter转码过滤器

CharacterEncodingFilter类可以通过简单配置来帮我们实现字符集转换的功能。

filter路径：cn.com.easy.filter.CharacterEncodingFilter

参数列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| encoding | 字符集编码格式 | N | String | 是 | 无 |

实例：

<filter>

<filter-name>CharacterEncoding</filter-name>

<filter-class>

cn.com.easy.filter.CharacterEncodingFilter

</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>UTF-8</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>CharacterEncoding</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

# EASY MVC



## ActionServlet中央控制器

ActionServlet 类是 easy 的控制中心类，所有的控制器组件和业务组件的初始化、Action参数匹配和Action 的执行都需要这个类来进行导航。如果想要使用easy框架的MVC，那么就需要添加ActionServlet。

ActionServlet路径：cn.com.easy.action.ActionServlet

参数列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名称** | **说明** | **EL** | **类型** | **必选** | **默认值** |
| componentScan | 系统中MVC控制器、业务逻辑组件、数据组件等类所在包的地址。例如：我们系统中有一个控制器类cn.com.controller.SystemC，那么我们就可以写成cn.com.controller或cn.com或cn。 | N | String | 是 | 无 |

实例：

<servlet>

<servlet-name>ActionServlet</servlet-name>

<servlet-class>cn.com.easy.action.ActionServlet</servlet-class>

<init-param>

<param-name>componentScan</param-name>

<param-value>cn.com.controller</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>ActionServlet</servlet-name>

<url-pattern>\*.e</url-pattern>

</servlet-mapping>

# Expression Language



## EL简介

EL全名为Expression Language，它原本是JSTL 1.0为方便存取数据所自定义的语言。当时EL只能在JSTL 标签中使用，如下：

<c:out value="${ 3 + 7}">

程序执行结果为10。但是你却不能直接在JSP 网页中使用：

<p>Hi ! ${ username }</p>

到了JSP 2.0之后，EL已经正式纳入成为标准规范之一。因此，只要是支持Servlet 2.4/JSP2.0的Container，就都可以在JSP网页中直接使用EL了。

除了JSP 2.0建议使用EL 之外，JavaServer Faces( JSR-127 )也考虑将EL 纳入规范，由此可知，EL 如今已经是一项成熟、标准的技术。

注意,假若您所用的Container只支持Servlet 2.3/JSP 1.2，如：Tomcat 4.1.29，您就不能在JSP网页中直接使用EL，必须安装支持Servlet 2.4/JSP 2.0的Container。

## EL语法

EL语法很简单，它最大的特点就是使用上很方便。接下来介绍EL主要的语法结构：${sessionScope.user.sex}

所有EL都是以${为起始、以}为结尾的。上述EL范例的意思是：从Session的范围中，取得用户的性别。假若依照之前JSP Scriptlet的写法如下：

User user = (User)session.getAttribute("user");

String sex = user.getSex( );

两者相比较之下，可以发现EL的语法比传统JSP Scriptlet更为方便、简洁。

### 2.2.1 .与 [ ] 运算符

EL 提供 . 和 [ ] 两种运算符来存取数据。下列两者所代表的意思是一样的：

${sessionScope.user.sex} 等于 ${sessionScope.user["sex"]}

. 和 [ ] 也可以同时混合使用，如下：

${sessionScope.shoppingCart[0].price}

回传结果为shoppingCart中第一项物品的价格。

不过，以下两种情况，两者会有差异：

(1) 当要存取的属性名称中包含一些特殊字符，如. 或 – 等并非字母或数字的符号，就一定要使用 [ ]，例如：

${user.My-Name }

上述是不正确的方式，应当改为：

${user["My-Name"]}

(2) 我们来考虑下列情况：

${sessionScope.user[data]}

此时，data 是一个变量，假若data的值为"sex"时，那上述的例子等于${sessionScope.user.sex}；假若data 的值为"name"时，它就等于${sessionScope.user.name}。因此，如果要动态取值时，就可以用上述的方法来做，但. 无法做到动态取值。

接下来，我们更详细地来讨论一些情况，首先假设有一个EL：

${expr-a[expr-b]}

1. 当expr-a 的值为null时，它会回传null。
2. 当expr-b的值为null时，它会回传null。
3. 当 expr-a 的值为一Map 类型时：

* 假若 !value-a.containsKey(value-b)为真，则回传null。
* 否则回传value-a.get(value-b)。

1. 当expr-a 的值为List 或array 类型时：

* 将 value-b 的值强制转型为int，假若不能转型为int 时，会产生error。
* 然后， 假若value-a.get(value-b) 或Array.get(value-a, value-b) 产生ArrayIndexOutOfBoundsException 或IndexOutOfBoundsException 时，则回传null。
* 假若 value-a.get(value-b)或Array.get(value-a, value-b)产生其他的异常时，则会产生error。
* 最后都没有任何异常产生时，回传value-a.get(value-b)或Array.get(value-a, value-b)。

1. 当expr-a 的值为JavaBean 对象时：

* 将 value-b 的值强制转型为String。
* 假若 getter 产生异常时，则会产生error。若没有异常产生时，则回传getter 的结果。

### 2.2.2 EL 变量

EL 存取变量数据的方法很简单，例如：${username}。它的意思是取出某一范围中名称为username的变量。因为我们并没有指定哪一个范围的username，所以它的默认值会先从Page 范围找，假如找不到，再依序到Request、Session、Application范围。假如途中找到username，就直接回传，不再继续找下去，但是假如全部的范围都没有找到时，就回传null（见表6-1）：

表 6-1

|  |  |
| --- | --- |
| **属性范围** | **在 EL 中的名称** |
| Page | PageScope |
| Request | RequestScope |
| Session | SessionScope |
| Application | ApplicationScope |

自动搜索顺序

我们也可以指定要取出哪一个范围的变量（见表6-2）：

表 6-2

|  |  |
| --- | --- |
| **范围** | **说明** |
| ${pageScope.username} | 取出Page 范围的username 变量 |
| ${requestScope.username} | 取出Request 范围的username 变量 |
| ${sessionScope.username} | 取出Session 范围的username 变量 |
| ${applicationScope.username} | 取出Application 范围的username 变量 |

其中，pageScope、requestScope、sessionScope和applicationScope都是EL 的隐含对象，由它们的名称可以很容易猜出它们所代表的意思，例如：${sessionScope.username}是取出Session范围的username 变量。这种写法是不是比之前JSP 的写法：

String username = (String) session.getAttribute("username");

容易、简洁许多。有关EL 隐含对象在6-3 节中有更详细的介绍。

### 2.2.3自动转变类型

EL 除了提供方便存取变量的语法之外，它另外一个方便的功能就是：自动转变类型，我们来看下面这个范例：

${param.count + 20}

假若窗体传来count的值为10时，那么上面的结果为30。之前没接触过JSP 的读者可能会认为上面的例子是理所当然的，但是在JSP 1.2 之中不能这样做，原因是从窗体所传来的值，它们的类型一律是String，所以当你接收之后，必须再将它转为其他类型，如：int、float 等等，然后才能执行一些数学运算，下面是之前的做法：

String str\_count = request.getParameter("count");

int count = Integer.parseInt(str\_count);

count = count + 20;

接下来再详细说明EL类型转换的规则：

* 1. 将A 转为String 类型
* 假若 A 为String 时：回传A
* 否则，当A 为null 时：回传 ""
* 否则，当A.toString( )产生异常时：错误!
* 否则，回传 A.toString( )
  1. 将A 转为Number 类型的N
* 假若 A 为null 或 "" 时：回传0
* 假若 A 为Character 时：将A 转为 new Short((short)a.charValue( ))
* 假若 A 为Boolean 时：错误!
* 假若 A 为Number 类型和N 一样时：回传A
* 假若 A 为Number 时：
* 假若N 是BigInteger 时：
* 假若A 为BigDecimal 时：回传 A.toBigInteger( )
* 否则，回传 BigInteger.valueOf(A.longValue( ))
* 假若N 是BigDecimal 时：
* 假若A 为BigInteger 时：回传 A.toBigDecimal( )
* 否则，回传 BigDecimal.valueOf(A.doubleValue( ))
* 假若N 为Byte 时：回传 new Byte(A.byteValue( ))
* 假若N 为Short 时：回传 new Short(A.shortValue( ))
* 假若N 为Integer 时：回传 new Integer(A.intValue( ))
* 假若N 为Long 时：回传 new Long(A.longValue( ))
* 假若N 为Float 时：回传 new Float(A.floatValue( ))
* 假若N 为Double 时：回传 new Double(A.doubleValue( ))
* 否则，错误!
* 假若 A 为String 时：
* 假若N 是BigDecimal 时：
* 假若 new BigDecimal(A)产生异常时：错误!
* 否则，回传 new BigDecimal(A)
* 假若N 是BigInteger 时：
* 假若 new BigInteger(A)产生异常时：错误!
* 否则，回传 new BigInteger(A)
* 假若N.valueOf(A)产生异常时：错误!
* 否则，回传 N.valueOf(A)
* 否则，错误!
  1. 将A 转为Character 类型
* 假若 A 为null 或 "" 时：回传 (char)0
* 假若 A 为Character 时：回传 A
* 假若 A 为Boolean 时：错误!
* 假若 A 为Number 时：转换为Short 后，然后回传Character
* 假若 A 为String 时：回传 A.charAt(0)
* 否则，错误!
  1. 将A 转为Boolean 类型
* 假若 A 为null 或 "" 时：回传 false
* 否则，假若A 为Boolean 时：回传 A
* 否则，假若A为String，且Boolean.valueOf(A)没有产生异常时：回传Boolean.valueOf(A)
* 否则，错误!

### 2.2.4 EL 保留字

EL 的保留字如表6-3：

表6-3

|  |  |  |  |
| --- | --- | --- | --- |
| And | eq | gt | true |
| Or | ne | le | false |
| No | lt | ge | null |
| instanceof | empty | div | mod |

所谓保留字的意思是指变量在命名时，应该避开上述的名字，以免程序编译时发生错误。

## EL隐含对象

EL本身也有自己的隐含对象。EL 隐含对象总共有11 个（见表6-4）

表6-4

|  |  |  |
| --- | --- | --- |
| **隐含对象** | **类型** | **说明** |
| PageContext | javax.servlet.ServletContext | 表示此JSP 的PageContext |
| PageScope | java.util.Map | 取得Page范围的属性名称所对应的值 |
| RequestScope | java.util.Map | 取得Request 范围的属性名称所对应的值 |
| sessionScope | java.util.Map | 取得Session 范围的属性名称所对应的值 |
| applicationScope | java.util.Map | 取得Application 范围的属性名称所对应的值 |
| param | java.util.Map | 如同  ServletRequest.getParameter(Stringname)。回传String 类型的值 |
| paramValues | java.util.Map | 如同  ServletRequest.getParameterValues(Stringname)。回传String []类型的值 |
| header | java.util.Map | 如同ServletRequest.getHeader(String  name)。回传String 类型的值 |
| headerValues | java.util.Map | 如同ServletRequest.getHeaders(String  name)。回传String []类型的值 |
| cookie | java.util.Map | 如同HttpServletRequest.getCookies( ) |
| initParam | java.util.Map | 如同  ServletContext.getInitParameter(String name)。回传String 类型的值 |

这 11 个隐含对象(Implicit Object)分成三类：

* + 1. 与范围有关的隐含对象

applicationScope

sessionScope

requestScope

pageScope

* + 1. 与输入有关的隐含对象

param

paramValues

* + 1. 其他隐含对象

cookie

header

headerValues

initParam

pageContext

依照上面的分类顺序，介绍这些隐含对象。

### 2.3.1属性(Attribute)与范围(Scope)

与范围有关的EL 隐含对象包含以下四个：pageScope、requestScope、sessionScope 和applicationScope，它们基本上就和JSP的pageContext、request、session和application一样，所以笔者在这里只稍略说明。不过必须注意的是，这四个隐含对象只能用来取得范围属性值，即JSP中的getAttribute(String name)，却不能取得其他相关信息，例如：JSP中的request对象除可以存取属性之外，还可以取得用户的请求参数或表头信息等等。但是在EL中，它就只能单纯用来取得对应范围的属性值，例如：我们要在session 中储存一个属性，它的名称为username，在JSP 中使用session.getAttribute("username") 来取得username 的值， 但是在EL 中， 则是使用${sessionScope.username}来取得其值的。接下来分别对这四个隐含对象做简短的说明：

* pageScope

范围和JSP 的Page 相同，也就是单单一页JSP Page 的范围(Scope)。

* requestScope

范围和JSP 的Request 相同，requestScope 的范围是指从一个JSP 网页请求到另一个JSP 网页请求之间，随后此属性就会失效。

* sessionScope

范围和JSP Scope 中的session 相同，它的属性范围就是用户持续在服务器连接的时间。

* applicationScope

范围和JSP Scope 中的application 相同，它的属性范围是从服务器一开始执行服务，到服务器关闭为止。

### 2.3.2与输入有关的隐含对象

与输入有关的隐含对象有两个：param和paramValues，它们是EL中比较特别的隐含对象。一般而言，我们在取得用户的请求参数时，可以利用下列方法：

request.getParameter(String name)

request.getParameterValues(String name)

在 EL中则可以使用param和paramValues两者来取得数据。

${param.name}

${paramValues.name}

这里param 的功能和request.getParameter(String name) 相同， 而paramValues 和request.getParameterValues(String name)相同。如果用户填了一个表格，表格名称为username，则我们就可以使用${param.username}来取得用户填入的值。

### 2.3.3其他隐含对象

介绍完上面六个隐含对象后，接下来将介绍最后五个隐含对象。

* cookie

所谓的cookie是一个小小的文本文件，它是以key、value的方式将Session Tracking的内容记录在这个文本文件内，这个文本文件通常存在于浏览器的暂存区内。JSTL并没有提供设定cookie的动作，因为这个动作通常都是后端开发者必须去做的事情，而不是交给前端的开发者。假若我们在cookie 中设定一个名称为userCountry的值，那么可以使用${cookie.userCountry}来取得它。

* header 和headerValues

header 储存用户浏览器和服务端用来沟通的数据，当用户要求服务端的网页时，会送出一个记载要求信息的标头文件，例如：用户浏览器的版本、用户计算机所设定的区域等其他相关数据。假若要取得用户浏览器的版本，即${header["User-Agent"]}。另外在鲜少机会下，有可能同一标头名称拥有不同的值，此时必须改为使用headerValues 来取得这些值。

注意：因为User-Agent 中包含“-”这个特殊字符，所以必须使用“[]”，而不能写成$(header.User-Agent)

* initParam

就像其他属性一样，我们可以自行设定web 站台的环境参数(Context)，当我们想取得这些参数时，可以使用initParam 隐含对象去取得它，例如：当我们在web.xml 中设定如下：

<context-param>

<param-name>userid</param-name>

<param-value>mike</param-value>

</context-param>

那么我们就可以直接使用 ${initParam.userid}来取得名称为userid，其值为mike 的参数。下面是之前的做法：

String userid = (String)application.getInitParameter("userid");

* pageContext

我们可以使用 ${pageContext}来取得其他有关用户要求或页面的详细信息。表6-5 列出了几个

比较常用的部分。

|  |  |
| --- | --- |
| **Expression** | **说明** |
| ${pageContext.request.queryString} | 取得请求的参数字符串 |
| ${pageContext.request.requestURL} | 取得请求的URL，但不包括请求之参数字符串 |
| ${pageContext.request.contextPath} | 服务的web |
| ${pageContext.request.method} | 取得HTTP 的方法(GET、POST) |
| ${pageContext.request.protocol} | 取得使用的协议(HTTP/1.1、HTTP/1.0) |
| ${pageContext.request.remoteUser} | 取得用户名称 |
| ${pageContext.request.remoteAddr } | 取得用户的IP 地址 |
| ${pageContext.session.new} | 判断session 是否为新的，所谓新的session，表示刚由server 产生而client 尚未使用 |
| ${pageContext.session.id} | 取得session 的ID |
| ${pageContext.servletContext.serverInfo} | 取得主机端的服务信息 |

注意：因为 ${} 在JSP 2.0 中是特殊字符，JSP容器会自动将它当做EL来执行，因此，假若要显示 ${}时，必须在 $ 前加上 \ ，如：\${ XXXXX }

## EL算术运算符

EL 算术运算符主要有以下五个

|  |  |  |  |
| --- | --- | --- | --- |
| **算术运算符** | **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 | 余数 | ${ 17 % 5 } 或 ${ 17 mod 5 } | 2 |

接下来，我们依照下列几种情况，详细说明EL 算术运算符的规则：

1. A {+ ,－ , \*} B

* 假若 A 和B 为null：回传 (Long)0
* 假若 A 或B 为BigDecimal 时，将另一个也转为BigDecimal，则：
* 假若运算符为 + 时：回传 A.add(B)
* 假若运算符为－ 时：回传 A.subtract(B)
* 假若运算符为 \* 时：回传 A.multiply(B)
* 假若 A 或B 为Float、Double 或包含 e / E 的字符串时：
* 假若A 或B 为BigInteger 时，将另一个转为BigDecimal，然后依照运算符执行运算
* 否则，将两者皆转为Double，然后依照运算符执行运算
* 假若 A 或B 为BigInteger 时，将另一个也转为BigInteger，则：
* 假若运算符为 + 时：回传 A.add(B)
* 假若运算符为－ 时：回传 A.subtract(B)
* 假若运算符为 \* 时：回传 A.multiply(B)
* 否则，将A 和B 皆转为Long，然后依照运算符执行运算
* 假若运算结果产生异常时，则错误!

1. (2) A {/ , div} B

* 假若 A 和B 为null：回传 (Long)0
* 假若 A 或B 为BigDecimal 或BigInteger 时，皆转为BigDecimal，然后回传 A.divide(B,BigDecimal.ROUND\_HALF\_UP)
* 否则，将A 和B 皆转为Double，然后依照运算符执行运算
* 假若运算结果产生异常时，则错误!

1. A {% , mod} B

* 假若 A 和B 为null：回传 (Long)0
* 假若A 或B为BigDecimal、Float、Double或包含 e / E 的字符串时，皆转为Double，然
* 后依照运算符执行运算
* 假若 A 或B 为BigInteger 时，将另一个转为BigInteger，则回传 A.remainder(B)
* 否则，将A 和B 皆转为Long，然后依照运算符执行运算
* 假若运算结果产生异常时，则错误!

1. -A

* 假若 A 为null：回传 (Long)0
* 假若 A 为BigDecimal 或BigInteger 时，回传 A.negate( )
* 假若 A 为String 时：
* 假若A 包含 e / E 时，将转为Double，然后依照运算符执行运算
* 否则，转为Long，然后依照运算符执行运算
* 假若运算结果产生异常时，则错误!
* 假若 A 为Byte、Short、Integer、Long、Float 或Double
* 直接依原本类型执行运算
* 假若运算结果产生异常时，则错误!
* 否则，错误!

## EL关系运算符

EL 关系运算符有以下六个运算符

|  |  |  |  |
| --- | --- | --- | --- |
| **关系运算符** | **说明** | **范例** | **结果** |
| = = 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 |

注意：在使用EL 关系运算符时，不能够写成：

${param.password1} = = ${param.password2}

或者

${ ${param.password1 } = = ${ param.password2 } }

而应写成

${ param.password1 = = param.password2 }

接下来，我们依照下列几种情况，详细说明EL 关系运算符的规则：

1. A {<, >, <=, >=, lt, gt, le, ge} B

* 假若 A= = B，运算符为<=, le, >=, ge 时，回传true，否则回传false
* 假若 A 为null 或B 为null 时，回传false
* 假若A或B为BigDecimal时，将另一个转为BigDecimal，然后回传A.compareTo(B)的值
* 假若 A 或B 为Float、Double 时，皆转为Double 类型，然后依其运算符运算
* 假若A或B为BigInteger时，将另一个转为BigInteger，然后回传A.compareTo(B)的值
* 假若A 或B为Byte、Short、Character、Integer或Long时，皆转为Long 类型，然后依其运算符运算
* 假若 A 或B 为String 时，将另一个也转为String，然后做词汇上的比较
* 假若 A 为Comparable 时，则：
* 假若A.compareTo(B)产生异常时，则错误!
* 否则，采用 A.compareTo(B) 的比较结果
* 假若 B 为Comparable 时，则：
* 假若 B.compareTo(A)产生异常时，则错误!
* 否则，采用 A.compareTo(B) 的比较结果
* 否则，错误!

1. A {= =, !=, eq, ne} B

* 假若 A= = B，依其运算符运算
* 假若 A 为null 或B 为null 时：= = /eq 则回传false，!= / ne 则回传true
* 假若 A 或B 为BigDecimal 时，将另一个转为BigDecimal，则：
* 假若运算符为 = = / eq，则回传A.equals(B)
* 假若运算符为 != / ne，则回传 !A.equals(B)
* 假若 A 或B 为Float、Double 时，皆转为Double 类型，然后依其运算符运算
* 假若 A 或B 为BigInteger 时，将另一个转为BigInteger，则：
* 假若运算符为 = = / eq，则回传A.equals(B)
* 假若运算符为 != / ne，则回传 !A.equals(B)
* 假若A 或B为Byte、Short、Character、Integer 或Long时，皆转为Long 类型，然后依其运算符运算
* 假若 A 或B 为Boolean 时，将另一个也转为Boolean，然后依其运算符运算
* 假若 A 或B 为String 时，将另一个也转为String，然后做词汇上的比较
* 否则，假若A.equals(B)产生异常时，则错误!
* 否则，然后依其运算符运算，回传A.equals(B)

## EL逻辑运算符

EL 逻辑运算符只有三个

|  |  |  |  |
| --- | --- | --- | --- |
| **逻辑运算符** | **说明** | **范例** | **结果** |
| && 或 and | 交集 | ${ A && B } 或 ${ A and B } | true / false |
| || 或 or | 并集 | ${ A || B } 或 ${ A or B } | true / false |
| ! 或 not | 非 | ${ !A } 或 ${ not A } | true / false |

下面举几个例子：

${ param.month = = 7 and param.day = = 14 }

${ param.month = = 7 || param.day = = 14 }

${ not param.choice }

EL 逻辑运算符的规则很简单：

1. A {&&, and, || 或 or } B
   * 将A 和B 转为Boolean，然后依其运算符运算
2. {!, not}A
   * 将A 转为Boolean，然后依其运算符运算

## EL其他运算符

EL 除了上述三大类的运算符之外，还有下列几个重要的运算符：

1. Empty 运算符
2. 条件运算符
3. ( ) 括号运算符

### 2.7.1 Empty 运算符

Empty 运算符主要用来判断值是否为null 或空的，例如：

${ empty param.name }

接下来说明Empty 运算符的规则：

(1) {empty} A

* 假若 A 为null 时，回传true
* 否则，假若A 为空String 时，回传true
* 否则，假若A 为空Array 时，回传true
* 否则，假若A 为空Map 时，回传true
* 否则，假若A 为空Collection 时，回传true
* 否则，回传false

### 2.7.2 条件运算符

所谓条件运算符如下：

${ A ? B : C}

意思是说，当A 为true 时，执行B；而A 为false 时，则执行C。

### 2.7.3 括号运算符

括号运算符主要用来改变执行优先权，例如：${ A \* (B+C) }

至于运算符的优先权，如下所示(由高至低，由左至右)：

* []、.
* ( )
* - (负)、not、!、empty
* \*、/、div、%、mod
* +、- (减)
* <、>、<=、>=、lt、gt、le、ge
* = =、!=、eq、ne
* &&、and
* ||、or
* ${ A ? B : C}

EL 的数学运算符、相等运算符、关系运算符和逻辑运算符就跟其他程序语言一样，并没有特别的地方。但是它的empty 运算符就比较特别，为了测试它，写了这样一行程序代码：

<c:set value="mike" var="username" scope="request" />

这样Request 属性范围里就存在一个名称为username、值为mike 的属性。执行此程序时，将会发现${empty username}为false； ${empty password}为true，其代表的意义就是：它可以在四种属性范围中找到username这个属性，但是找不到password这个属性。