# 表达式语言（EL）简介

1. 使用表达式语言（EL）可以方便的访问属性范围中的属性，可以避免出现许多的

Scriptlet代码。访问语法如下：

**$ { 属性名称 }**

1. 使用表达式语言可以方便地访问对象中的属性、提交的参数或者是进行各种数学运 算，而且使用表达式语言的最大好处是如果输出的内容为null，则会自动使用空字 符串””表示。
2. 表达式语言（Expression Language， EL）可以使JSP页面更加整洁。

总之一点，表达式语言的功能是在JSP页面进行内容的显示。

# EL内置对象与常用操作

表达式语言的主要功能就是进行内容的显示。为了显示方便，表达式语言中提供了许 多的内置对象，通过对不同内置对象的设置，表达式语言可以输出不同的内容。这些内 置对象如下所示：

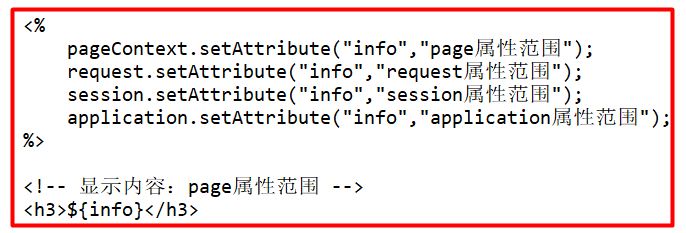
|  |  |  |
| --- | --- | --- |
| **No** | **表达式内置对象** | **说明** |
| **1** | **pageContext** | **表示javax.servlet.jsp.PageContext内置对象** |
| **2** | **pageScope** | **表示从page属性范围查找属性** |
| **3** | **requestScope** | **表示从request属性范围查找属性** |
| **4** | **sessionScope** | **表示从session属性范围查找属性** |
| **5** | **applicationScope** | **表示从application属性范围查找属性** |
| **6** | **param** | **接收传递到本页面的参数** |
| **7** | **paramValues** | **接收传递到本页面的一组参数** |
| 8 | header | 取得一个头信息数据 |
| 9 | headerValues | 取得一组头信息数据 |
| 10 | cookie | 取得cookie中的数据 |
| 11 | initParam | 取得配置的初始化参数 |

利用这写内置对象完成的常用操作有：访问4种属性范围的内容、调用内置对象的操作、 接收请求参数等。

## 访问指定范围中的属性

1. 如果在不同的属性范围中设置了相同的属性名称，则将按照如下顺序进行查找： **page ---> request ---> session --> application**

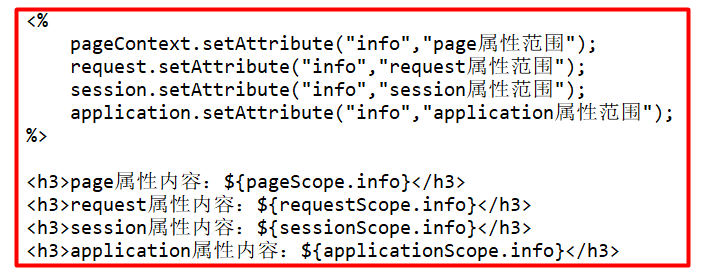
例如：



1. 这时如果想要取出指定范围内的属性，则可以使用表达式的内置对象完成，如下：

|  |  |
| --- | --- |
| **范例** | **说明** |
| ${ pageScope.属性名} | 取出page范围的属性内容 |
| ${requestScope.属性名} | 取出request范围的属性内容 |
| ${sessionScope.属性名} | 取出session范围的属性内容 |
| ${applicationScope.属性名} | 取出application范围的属性内容 |

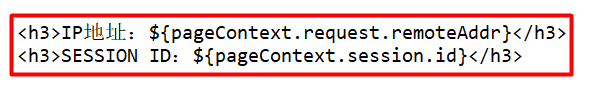
例如：



## **调用内置对象的操作**

在讲解JSP内置对象时就介绍过pageContext对象的作用，使用pageContext对象可以取 得request、session、application等内置对象，而在表达式语言中刚好内置了pageContext 对象，同时在表达式语言中允许方法的调用，进而可以通过pageContext对象获取其他 的内置对象，调用它们的方法。在通过表达式的内置对象调用方法时，都是以调用 getXxx( )或isXxx( )形式的方法居多。（其调用方法的内部原理是通过反射机制实现的）。

例如：



## 接收请求参数

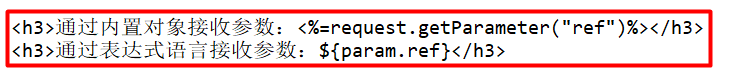
表达式语言可以完成request对象的getParameter( )和getParameterValues( )方法的功 能，但无法使用变量接收参数值，只能在页面上显示。

1. **接收参数**

使用表达式语言完成request对象的getParameter( )方法的功能。其语法为：

**${ param.参数名称 }**

示例如下：



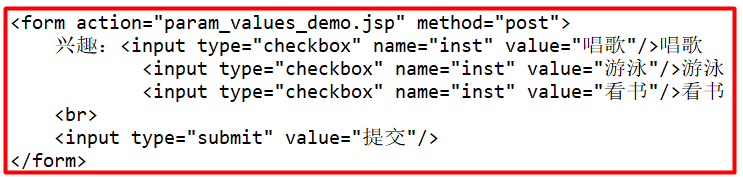
为了说明功能，同时使用了request和表达式语言两种方式显示参数；在运行时， 通过地址进行传参以上代码。

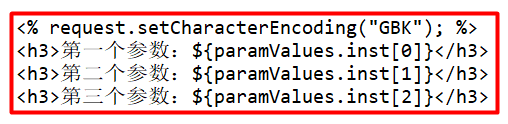
**2、接收一组参数**

使用表达式语言完成request对象的getParameterValues( )方法的功能。其语法为：

**${ paramValues.参数名称 }**

但需要注意，由于接收的是一组参数，所以如果想要取出，则需要分别指定下标。 示例如下：



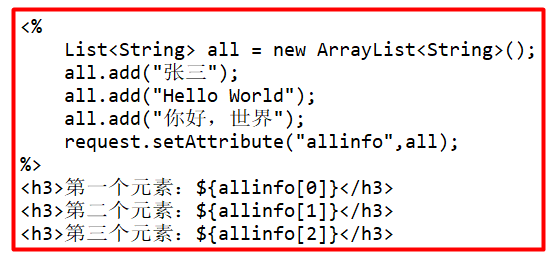


# 集合操作

集合操作在开发中被广泛地采用，在表达式语言中也已经很好地支持了集合的操作，可 以方便地使用表达式语言显示Collection（子接口：List、Set）、Map集合中的内容。

## 输出Collection接口集合

Collection接口是List、Set接口的父接口，属于单值型集合。所有的List、Set等类型的 集合都可以使用表达式语言进行显示。示例如下：



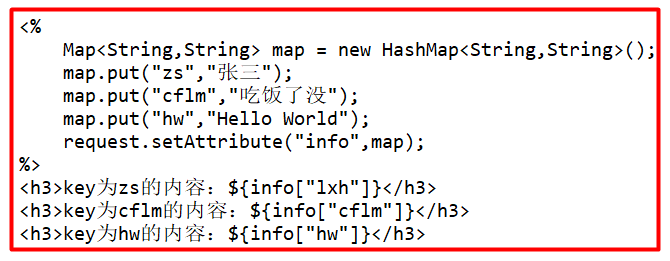
本程序首先定义一个List集合，并向其中添加了3个元素；

由于表达式语言只能访问保存在属性范围中的内容。通常在Servlet程序中将结果设置为request范围的属性，然后使用服务端跳转到JSP页面使用表达式语言进行显示。

在使用表达式语言输出时，直接通过集合的下标即可访问。

## 输出Map接口集合

Map接口是HashMap，HashTable等类的父接口，属于键值型集合。所有的HashMap、 HashTable等类型的集合都可以使用表达式语言进行输出。示例如下：



在访问Map数据时，就需要通过key找到对应的value；

在表达式语言中，除了可以采 用“[ ]”的形式访问，也可以采用“.”的形式访问；

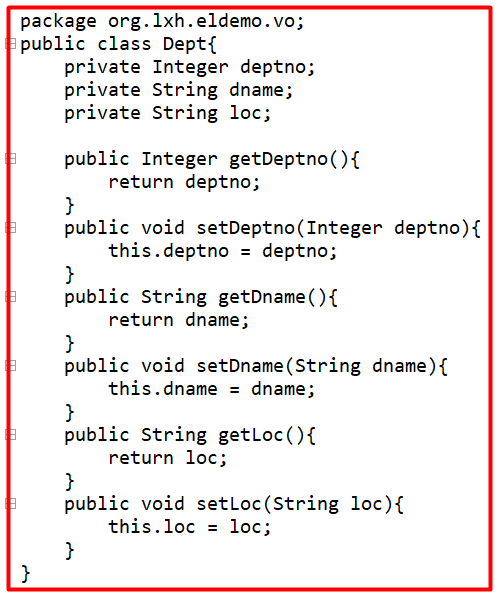
由于表达式语言只能访问保存在属性范围中的内容。通常在Servlet程序中将结果设置为request范围的属性，然后使用服务端跳转到JSP页面使用表达式语言进行显示。

# 在MVC中应用表达式语言

## 调用对象的方法

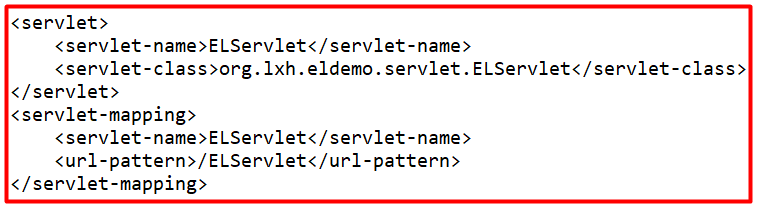
表达式语言的强大功能还在于，可以通过反射的方式调用保存在属性范围中的Java 对 象的方法。

1. 例如存在VO类—Dept.java：

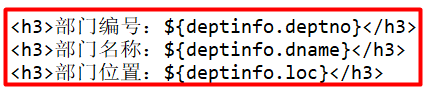


1. 在Servlet中将Dept类型对象保存在request属性范围中，进行服务器端跳转，通过表达式语言输出：





1. 编写print\_vo.jsp页面通过保存在属性范围中的对象调用方法显示内容：

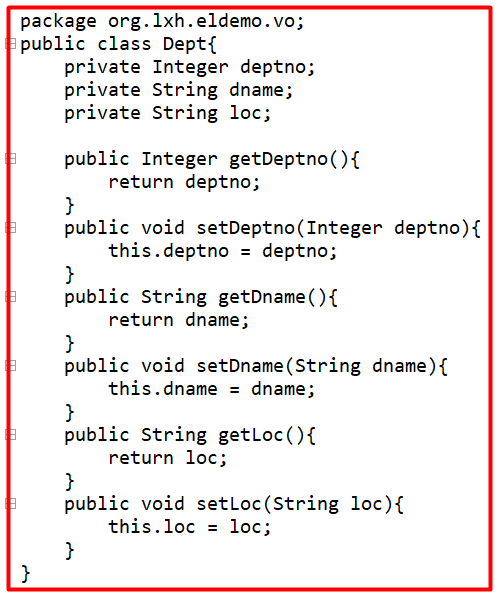


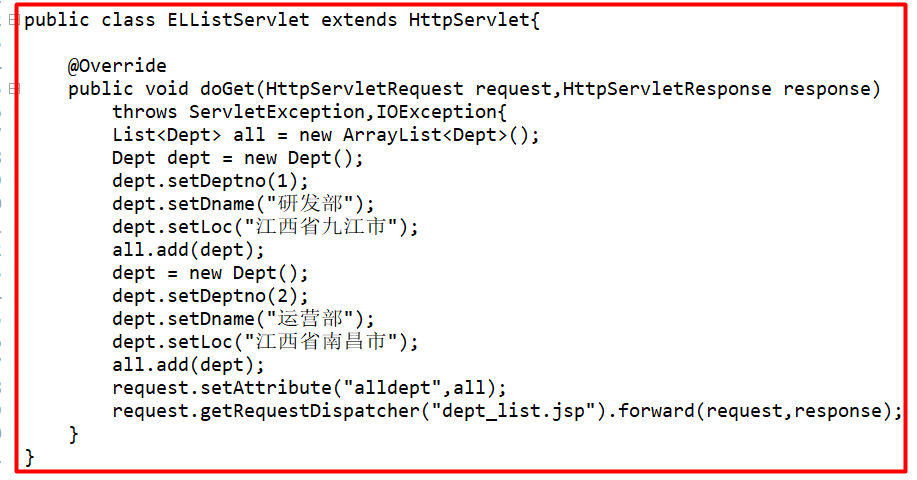
并将对象保存属性范围中，以后可以使用表达式语言输出时就会方便许多，可以直 接访问。但实际上还是依靠Java的反射机制完成的，为便于理解可以在getter方 法中加上系统输出，观察表达式语言中是否是通过反射调用的getter方法。

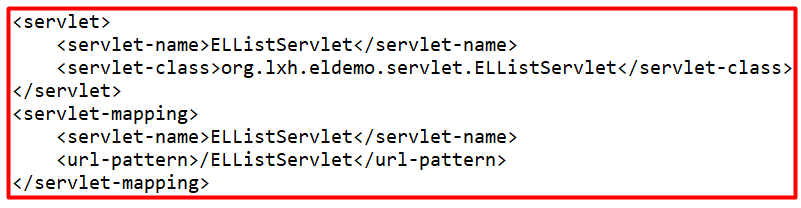
## 遍历对象集合

如果从数据库中查询的返回的是一组对象，在 Servlet中将其设置为属性范围中的数属性后，在JSP页面需要依靠Iterator接口进行迭代，但表达式语言只能操作4种范围中的内容，所以需要将每一个取出的对象存放在page范围中，再通过表达式语言进行输出显示。

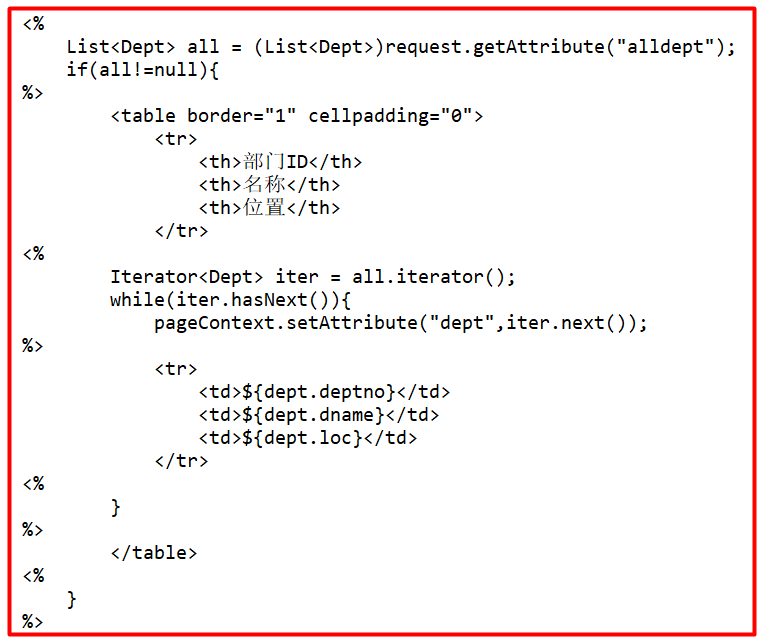
1. 例如存在VO类—Dept.java：



1. 编写Servlet，将对象集合保存在属性范围中：



1. 编写dept\_list.jsp页面，进行迭代输出显示：



在迭代时需要将每一个对象取出放在page范围中。

# 表达式语言中的运算符

在表达式语言中为了方便显示，定义了许多运算符，如算术运算符、关系运算符、逻辑 运算符等。使用这些运算符将使得JSP页面更加简洁；但是对于太复杂的操作还是应 该在Servlet或JavaBean中完成。

在使用这些运算符时，所有的操作内容也可以直接使用设置的属性，而不需要考虑转型 的问题。

## 算术运算符

表达式语言中提供了5种算术运算符，如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **算术运算符** | **描述** | **范例** | **结果** |
| + | 加法操作 | $ { 20+30 } | 50 |
| - | 减法操作 | $ { 20-30 } | -10 |
| \* | 乘法操作 | $ { 20\*30 } | 600 |
| / | 除法操作 | $ { 20/30 } | 0.666 |
| % | 取余（求模）操作 | $ { 20/30 } | 20 |

## 关系运算符

表达式语言中提供了6种关系运算符，如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **关系运算符** | **描述** | **范例** | **结果** |
| == 或 eq | 等于 | $ { 20==30 } | false |
| != 或 ne | 不等于 | $ { 20!=30 } | true |
| < 或 lt | 小于 | $ { 20<30 } | true |
| >或gt | 大于 | $ { 20>30 } | false |
| <=或le | 小于等于 | $ { 20<=30 } | true |
| >=或ge | 大于等于 | ${ 20>=30 } | false |

## 逻辑运算符

表达式语言中提供了3种逻辑运算符，如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **逻辑运算符** | **描述** | **范例** | **结果** |
| &&或and | 与操作 | $ { true&&false } | false |
| ||或or | 或操作 | $ { true&&false } | true |
| ! 或not | 非操作 | $ { ! false } | true |

## 其他运算符

除了上述运算符外，表达式语言中还有一些其他运算符，如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **其他运算符** | **描述** | **范例** | **结果** |
| empty | 判断属性是否为null | $ { empty info } | true |
| ? : | 三目运算符 | $ { 10> 20 : true ? false } | false |
| ( ) | 括号运算符 | $ { 10\*(10+20) } | 300 |