Atitit 重要的动态语言技术 表达式技术

目录

[1.1. 6.2 功能特性 1](#_Toc16434)

[1.2. 表达式求值 2](#_Toc26217)

[1.3. 6.5.19 表达式模板 表达式模板运行在一段文本中混合包含一个或多个求值表达式模块。各个求值块都通过可被自定义的前后缀字符分隔，一个通用的选择是使用#{ }作为分隔符。例如： 2](#_Toc3538)

[1.4. 6.5.10 构造器 调用java语言 构造器可以使用new操作符来调用。除了元数据类型和String（比如int,float等可以直接使用）都需要限定类的全名。 2](#_Toc22428)

[1.5. 集合操作 3](#_Toc20380)

## ****6.2 功能特性****

SpEL支持以下的一些特性：

* 字符表达式
* 布尔和关系操作符
* 正则表达式
* 类表达式
* 访问properties，arrays，lists，maps等集合
* 方法调用
* 关系操作符
* 赋值
* 调用构造器
* Bean对象引用
* 创建数组
* 内联lists
* 内联maps
* 三元操作符
* 变量
* 用户自定义函数
* 集合投影
* 集合选择
* 模板表达式

## ****表达式求值****

## 6.5.19 表达式模板 表达式模板运行在一段文本中混合包含一个或多个求值表达式模块。各个求值块都通过可被自定义的前后缀字符分隔，一个通用的选择是使用#{ }作为分隔符。例如：

|  |  |
| --- | --- |
| 1 | String randomPhrase = parser.parseExpression( |

|  |  |
| --- | --- |
| 2 | "random number is #{T(java.lang.Math).random()}", |

|  |  |
| --- | --- |
| 3 | **new** TemplateParserContext()).getValue(String.**class**); |

|  |  |
| --- | --- |
| 4 |  |

|  |  |
| --- | --- |
| 5 | // evaluates to "random number is 0.7038186818312008" |

## ****6.5.10 构造器 调用java语言**** 构造器可以使用new操作符来调用。除了元数据类型和String（比如int,float等可以直接使用）都需要限定类的全名。

|  |  |
| --- | --- |
| 1 | nventor einstein = p.parseExpression( |

|  |  |
| --- | --- |
| 2 | "new org.spring.samples.spel.inventor.Inventor('Albert Einstein', 'German')") |

|  |  |
| --- | --- |
| 3 | .getValue(Inventor.**class**); |

|  |  |
| --- | --- |
| 4 |  |

|  |  |
| --- | --- |
| 5 | //create new inventor instance within add method of List |

|  |  |
| --- | --- |
| 6 | p.parseExpression( |

|  |  |
| --- | --- |
| 7 | "Members.add(**new** org.spring.samples.spel.inventor.Inventor( |

|  |  |
| --- | --- |
| 8 | 'Albert Einstein', 'German'))").getValue(societyContext); |

## 集合操作