极客学院
jikexueyuan.com

Spring 表达式语言

Spring 表达式语言 — **课程概要**

- Spring 表达式语言的入门介绍
- Spring 表达式语言的操作范围
- Spring 表达式语言的运算符
- Spring 表达式语言的集合操作

Spring 表达式语言 – **学习目标**

知识目标：

- 熟悉Spring 表达式语言的基本概念
- 了解Spring 表达式语言的主要用途
- 清楚Spring 表达式语言的应用场景

技能目标：

- 创建基于Java的表达式语言
- 创建基于XML的表达式语言
- 创建基于注释的表达式语言

Spring 表达式语言的入门介绍

Spring表达式语言的入门介绍

- 基本概述
- 示例分析
- 工作原理
- 配置风格

Spring 表达式语言的入门介绍 – **基本概述**

Spring表达式语言全称为“Spring Expression Language”，缩写为“SpEL”，能在运行时构建复杂表达式、存取对象属性、对象方法调用等等，并且能与Spring功能完美整合。主要支持如下表达式：

- 基本表达式
- 类相关表达式
- 集合相关表达式
- 其他表达式

Spring 表达式语言的入门介绍 – **示例分析**

示例代码：

```
ExpressionParser parser = new SpelExpressionParser();
Expression expression =  parser.parseExpression("('Hello' + ' World').concat(#end)");
EvaluationContext context = new StandardEvaluationContext();
context.setVariable("end", "!");
System.out.println(expression.getValue(context));
```

步骤解析：

- 创建解析器
- 解析表达式
- 构造上下文
- 求值

Spring 表达式语言的入门介绍 – **工作原理**

基本概念：

- 创建解析器
- 解析表达式
- 构造上下文
- 求值

工作原理：

- 定义表达式
- 定义解析器ExpressionParser
  - ✓ 生成记号流
  - ✓ 生成抽象语法树
  - ✓ 生成Expression接口
- 定义上下文对象
- 根据表达式求值

主要接口：

- ExpressionParser接口
- EvaluationContext接口
- Expression接口

Spring 表达式语言的入门介绍 – 配置风格

XML风格的配置：

SpEL支持在Bean定义时注入，默认使用“#{SpEL表达式}”表示，其中“#root”根对象默认可以认为是ApplicationContext，只有ApplicationContext实现默认支持SpEL，获取根对象属性其实是获取容器中的Bean。

注解风格的配置：

基于注解风格的SpEL配置也非常简单，使用@Value注解来指定SpEL表达式，该注解可以放到字段、方法及方法参数上。

Spring 表达式语言的操作范围

Spring 表达式语言的操作范围

SpEL表达式的首要目标是通过计算获得某个值，在计算这个值得过程中，会使用到其他的值并会对这些值进行操作，值的操作范围如下：

- 字面值
- Bean及Bean的属性或方法
- 类的方法和常量

Spring 表达式语言的操作范围 – **字面值**

最简单的SpEL表达式仅包含一个简单的字面值：

```
<property name="count" value="#{5}"/>

<property name="message" value="The value is #{5}"/>

<property name="frequency" value="#{89.7}"/>

<property name="capacity" value="#{1e4}"/>

<property name="name" value="#{'Chuck'}"/>

<property name='name' value='#{"Chuck"}'/>

<property name="enabled" value="#{false}"/>
```

Spring 表达式语言的操作范围 – **Bean及Bean的属性或方法**

SpEL表达式可以引用Bean本身、Bean的属性以及Bean的方法：

引用Bean本身：

```
<property name="instrument" value="#{saxophone}"/>
<property name="instrument" ref="saxophone"/>
```

引用Bean的属性：

```
<bean id="carl"  class="com.jike.***.Instrumentalist">
        <property name="song" value="#{kenny.song}" />
</bean>

Instrumentalist carl = new Instrumentalist();
carl.setSong(kenny.getSong());
```

引用Bean的方法：

```
<property name="song" value="#{songSelector.selectSong()}"/>
<property name="song" value="#{songSelector.selectSong().toUpperCase()}"/>
<property name="song" value="#{songSelector.selectSong()?.toUpperCase()}"/>
```

Spring 表达式语言的操作范围 – **操作类的方法和常量**

在SpEL 中，使用T() 运算符会调用类作用域的方法和常量：

```
T(java.lang.Math)

<property name="multiplier" value="#{T(java.lang.Math).PI}"/>

<property name="randomNumber" value="#{T(java.lang.Math).random()}"/>
```

Spring 表达式语言的运算符

Spring 表达式语言的运算符

运算符类型：

运算符类型	运算符示例
数值运算	+、 -、 *、 %、 ^
比较运算	<、 >、 ==、 >=、 <=、lt、gt、eg、le、ge
逻辑运算	and、or、not、|
条件运算	?: (ternary)、 ?:(Elvis)
正则表达式	matches

Spring 表达式语言的运算符 – **数值运算**

数值运算符可以对SpEL 表达式中的值执行基础数学运算：

加运算： <property name="adjustedAmount" value="#{counter.total + 42}"/>

减运算： <property name="adjustedAmount" value="#{counter.total - 20}"/>

乘运算： <property name="circumference" value="#{2 * T(java.lang.Math).PI * circle.radius}"/>

除运算： <property name="average" value="#{counter.total / counter.count}"/>

求余运算： <property name="remainder" value="#{counter.total % counter.count}"/>

乘方运算： <property name="area" value="#{T(java.lang.Math).PI * circle.radius ^ 2}"/>

字符串连接： property name="fullName" value="#{performer.firstName + ' ' + performer.lastName}"/>

Spring 表达式语言的运算符 – **比较运算**

SpEL 同样提供了Java 所支持的比较运算符：

```
<property name="equal"   value="#{counter.total == 100}"/>

counter.total <= 100000

<property name="hasCapacity" value="#{counter.total le 100000}"/>
```

比较运算符：

运算符	符号	文本类型
等于	==	eq
小于	<	lt
小于等于	<=	le
大于	>	gt
大于等于	>=	ge

Spring 表达式语言的运算符 – **逻辑运算**

逻辑运算符用于对两个比较表达式进行求值，或者对某些布尔类型的值进行非运算：

运算符	操作
and	逻辑AND运算操作，只有运算符两边都是true，表达式才能使true
or	逻辑OR运算操作，只要运算符的任意一边是true，表达式就会使ture
not 或 ！	逻辑NOT运算操作，对运算结果求反

```
<property name="largeCircle"  value="#{shape.kind == 'circle' and shape.perimeter gt 10000}"/>

<property name="outOfStock" value="#{!product.available}"/>

<property name="outOfStock" value="#{not product.available}"/>
```

Spring 表达式语言的运算符 – **条件运算**

当某个条件为true 时，SpEL 表达式的求值结果是某个值；如果该条件为false 时，它的求值结果是另一个值时，可以使用SpEL 的三元运算符（?:）：

```
<property name="instrument"  value="#{songSelector.selectSong()=='Jingle Bells'?piano:saxophone}”  />

<property name="song"   value="#{kenny.song != null ? kenny.song : 'Greensleeves'}”  />

<property name="song" value="#{kenny.song ?: 'Greensleeves'}”  />
```

Spring 表达式语言的运算符 – **正则表达式**

当处理文本时，检查文本是否匹配某种模式有时是非常有用的。SpEL 通过 matches 运算符支持表达式中的模式匹配：

```
<property name="validEmail"
  value= "#{admin.email  matches '[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.com'}”     />
```

Spring 表达式语言的集合操作

Spring 表达式语言的集合操作

SpEL可以引用集合中的某个成员，就像在Java 里操作一样,同样具有基于属性值来过滤集合成员的能力：

- 访问集合成员
- 查询集合成员
- 投影集合

Spring 表达式语言的集合操作 – **访问集合成员**

为了展示SpEL访问集合成员的用途，需要定义一个City 类，然后使用<util:list> 元素在Spring 里配置了一个包含City 对象的List 集合，示例如下：

```
package com.jike.***;
public class City {
    private String name;
    private String state;
    private int population;
}
```

```
<util:list id="cities">
    <bean class="com.jike.***. City"
        p:name="Chicago" p:state="IL" p:population="2853114"/>
    <bean class="com.habuma.spel.cities.City"
        p:name="Las Cruces" p:state="NM “ p:population="91865"/>
</util:list>
```

Spring 表达式语言的集合操作 – **访问集合成员**

SpEL提供了几种运算法，这些运算符可以用在SpEL表达式中的值上：

```
<property name="chosenCity" value="#{cities[2]}"/>

<property name="chosenCity"  value="#{cities[T(java.lang.Math).random() * cities.size()]}"/>

<property name="chosenCity" value="#{cities['Dallas']}"/>

<util:properties id="settings"   location="classpath:settings.properties"/>

<property name="accessToken" value="#{settings['twitter.accessToken']}"/>

<property name="homePath" value="#{systemEnvironment['HOME']}"/>
```

Spring 表达式语言的集合操作 – **查询集合成员**

利用SpEL来查询集合成员：

```
<property name="bigCities" value="#{cities.?[population gt  100000]}"/>

<property name="aBigCity" value="#{cities.^[population gt  100000]}"/>

<property name="aBigCity" value="#{cities.$[population gt  100000]}"/>
```

Spring 表达式语言的集合操作 – **投影集合**

集合投影是从集合的每一个成员中选择特定的属性放入一个新的集合中。SpEL的投影运算符（.![]）完全可以做到这点：

```
<property name="cityNames" value="#{cities.![name]}"/>

<property name="cityNames" value="#{cities.![name + ', ' + state]}"/>

<property name="cityNames"   value="#{cities.?[population gt 100000].![name + ', ' + state]}"/>
```

Spring 表达式语言 – **课后练习**

请编写一个利用表达式语言来对集合进行操作的小程序，要求如下：

- 准备一个list集合，里面包含中国所有的省市自治区；
- 利用表达式语言的筛选功能，筛选出名字中包含“北”的省市自治区；
- 利用表达式语言的投影功能，将名字中包含“北”的省市自治区放入一个新的集合当中；

Spring 表达式语言

在本套课程中我们深入和详细的学习了SpEL，它是一种在运行期将值装配到Bean的属性或构造器参数中的表达式语言。通过本课程的学习，你应当可以利用SpEL处理以下事情：

- 通过Bean的id对Bean进行引用
- 调用方法以及引用对象中的属性
- 计算表达式的值
- 正则表达式的匹配
- 集合的操作

通过本课程的学习，你应该对Spring的表达式语言有了一个全面的认识。如果你想继续提高，可以继续在极客学院学习Spring的其他相关课程。

极客学院

jikexueyuan.com

中国最大的IT职业在线教育平台