# Spring 表达式语言 (Spring Expression Language) SpEL

分类: Spring2012-03-07 21:46 605 人阅读 评论(0) 收藏 举报

Spring 3.0 创建了一种新的方式用以配置对象的注入(set 注入或者构造参数注入),它便是 SpEL (Spring Expression Language)下面我们一一做一介绍。

# ▲基础特性

- ——SpEL 使用#{...}作为定界符,所有在大框号中的字符都将被认为是 SpEL.
- ——1、字面量的表示

# 1>整数

[html] view plaincopy

```
1. count" value="#{5}"/>
```

# 2>小数

[html] view plaincopy

```
1. roperty name="frequency" value="#{89.7}"/>
```

# 3>科学计数法

[html] view plaincopy

```
1. capacity" value="#{1e4}"/>
```

4>String 可以使用单引号或者双引号作为字符串的定界符号。

[html] view plaincopy

# 5>Boolean

[html] view plaincopy

```
1. operty name="enabled" value="#{false}"/>
```

——2、 引用 Bean, 属性和方法

# 1>引用其他对象

[html] view plaincopy

通过 id: "saxophone"将对象注入到 instrument 属性中,这与下面的配置是一样的:

[html] view plaincopy

```
1. cproperty name="instrument" ref="saxophone"/>
```

# 2> 引用其他对象的属性

[html] view plaincopy

kenny 是 Bean Id 而 song 是属性的名字,这样配置就如同我们写了如下的代码

[java] view plaincopy

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

# 3>调用其他方法

[html] view plaincopy

```
    <property name="song" value="songSelector.selectSong()"/>
```

调用了 Beanld 为"songSelector"的对象的 selectSong()方法,并将返回值注入到 song 属性中。或者还可以链式操作。如下:

```
[html] view plaincopy
```

```
1. cproperty name="song" value="songSelector.selectSong().toUpperCase()"/>
```

如果 songSelector.selectSong()返回 null 的还会抛出异常,为了避免我们要使用?.表达式。 这样如果 songSelector.selectSong()为 null 就不会再调用后面的方法了。如下

[html] view plaincopy

```
1. roperty name="song" value="songSelector.selectSong()?.toUpperCase()"/>
```

#### 4>调用静态方法

我们已经知道如何通过一个对象调用它的方法了,但是如何调用一个静态方法呢?用 T()。它将返回一个 Class object

然后我们再调用相应的方法即可:

[html] view plaincopy

```
1. cproperty name="multiplier" value="T(java.lang.Math).PI"/>
```

# ▲SpEL 支持的运算符号

——1、算数运算符: +, -, \*, /, %, ^

[html] view plaincopy

```
1. cproperty name="adjustedAmount" value="#{counter.total + 42}"/>
2. cproperty name="adjustedAmount" value="#{counter.total - 20}"/>
3. cproperty name="circumference" value="#{2 * T(java.lang.Math).PI * circle.ra dius}"/>
4. cproperty name="average" value="#{counter.total / counter.count}"/>
5. cproperty name="remainder" value="#{counter.total % counter.count}"/>
6. cproperty name="area" value="#{T(java.lang.Math).PI * circle.radius ^ 2}"/>
```

# 加号还可以用作字符串连接

[html] view plaincopy

```
——2、比较运算符:    <, >, ==, <=, >=, lt, gt, eq, le, ge
```

[html] view plaincopy

```
1. roperty name="equal" value="#{counter.total == 100}"/>
```

不可以使用<和>号,应为在 xml 中它有特殊的含义,我们使用 lt 和 gt 代替

[html] view plaincopy

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

——3、 逻辑运算符号: and, or, not, |

[html] view plaincopy

- 2. cproperty name="outOfStock" value="#{!product.available}"/>
- 3. <property name="outOfStock" value="#{not product.available}"/>
- ——4、 If-else 运算符: ?: (ternary), ?: (Elvis)
- 〇最基本的 ?: (这如同我们在使用 EL 表达式语言):

[html] view plaincopy

```
1. roperty name="instrument" value="#{songSelector.selectSong() == 'Jingle Be
lls' ? piano : ' Jingle Bells '}"/>
```

〇变体的?:

[html] view plaincopy

```
1. roperty name="song" value="#{kenny.song ?: 'Greensleeves'}"/>
```

# ——5、 正则表达式: matches

[html] view plaincopy

表达式返回逻辑值,如果匹配返回 true,否则返回 false

▲SpEL 对集合的支持

# 有实体 City 定义如下:

[java] view plaincopy

```
    package com.habuma.spel.cities;
    public class City {
    private String name;
    private String state;
    private int population;
    }
```

#### Xml 中有如下定义

[html] view plaincopy

```
1. <util:list id="cities">
2. <bean class="com.habuma.spel.cities.City"</pre>
3. p:name="Chicago" p:state="IL" p:population="2853114"/>
4. <bean class="com.habuma.spel.cities.City"</pre>
5. p:name="Atlanta" p:state="GA" p:population="537958"/>
6. <bean class="com.habuma.spel.cities.City"</pre>
7. p:name="Dallas" p:state="TX" p:population="1279910"/>
8. <bean class="com.habuma.spel.cities.City"</pre>
9. p:name="Houston" p:state="TX" p:population="2242193"/>
10. <bean class="com.habuma.spel.cities.City"</pre>
11. p:name="Odessa" p:state="TX" p:population="90943"/>
12. <bean class="com.habuma.spel.cities.City"</pre>
13. p:name="El Paso" p:state="TX" p:population="613190"/>
14. <bean class="com.habuma.spel.cities.City"
15. p:name="Jal" p:state="NM" p:population="1996"/>
16. <bean class="com.habuma.spel.cities.City"</pre>
17. p:name="Las Cruces" p:state="NM" p:population="91865"/>
18. </util:list>
```

# ——1、 获取 Collection 中的某个对象

〇通过下标访问,如下:

[html] view plaincopy

```
1. roperty name="chosenCity" value="#{cities[2]}"/>
```

我们就会获得 population 为"1279910"的 city(记住下标从 0 开始) 〇下标可以通过变量指定,如下:

```
[html] view plaincopy
```

```
1. roperty name="chosenCity" value="#{cities[T(java.lang.Math).random() * cit
ies.size()]}"/>
```

〇如果是从 Map 中获得,可指定 key 值,如下

[html] view plaincopy

```
1. cproperty name="chosenCity" value="#{cities['Dallas']}"/>
```

〇也可以通过 key 访问 properties 的值,如下

[html] view plaincopy

```
    <util:properties id="settings" location="classpath:settings.properties"/>
    <property name="accessToken" value="#{settings['twitter.accessToken']}"/>
```

〇可以通过下标访问 systemEnvironment 和 SystemProperties 中的值

[html] view plaincopy

```
1. cproperty name="homePath" value="#{systemEnvironment['HOME']}"/>
```

〇如果在 jre 运行时配置了-Dapplication.home=/etc/myapp,我们可以通过如下方式访问 [html] view plaincopy

```
1. operty name="homePath" value="#{systemProperties['application.home']}"/>
```

〇通过下标获取 String 串中的某个字符

[html] view plaincopy

```
1. 'This is a test'[3]
```

——2、获取 Collection 中的子集-通过条件筛选(注意新对象是一个新的 Collection) 1>筛选子集(.?[])

[html] view plaincopy

```
1. roperty name="bigCities" value="#{cities.?[population gt 100000]}"/>
 2>获取第一个(.^[])
[html] view plaincopy
  1. roperty name="aBigCity" value="#{cities.^[population gt 100000]}"/>
 3>获取最后一个(.$[])
[html] view plaincopy
  1. roperty name="aBigCity" value="#{cities.$[population gt 100000]}"/>
——3、集合的投影(.![])
如果想获得所有城市的名称组成的列表, 可用如下操作
[html] view plaincopy
  1. cityNames" value="#{cities.![name]}"/>
将返回"Chicago", "Atlanta", "Dallas"
也可以组合两个列,如下:
[html] view plaincopy
  1. cityNames" value="#{cities.![name + ', ' + state]}"/>
将返回"Chicago, IL", "Atlanta, GA", and "Dallas, TX".
—— 4、将投影和筛选结合
[html] view plaincopy
  1. operty name="cityNames" value="#{cities.?[population gt 100000].![name +
      ', ' + state]}"/>
```