链滴

SpEL 语法

作者：kenan
原文链接：https://ld246.com/article/1490245370566
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

<h3 style="font-size:1.2em;font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:FFFFFF;">
5.3.1 基本表达式
</h3>

一、字面量表达式： SpEL支持的字面量包括：字符串、数字类型（in、long、float、double）、布尔类型、null类型。

<table cellspacing="0" cellpadding="0" border="1" style="color:#000000;font-family:Helvetic, Tahoma, Arial, sans-serif;background-color:#FFFFFF;font-size:12px;padding:0px;margin:0px;"

<tbody>
<tr>
<td width="72" style="font-size:1em;font-family:verdana, arial, helvetica, sans-serif;">

类型

</td>
<td width="468" style="font-size:1em;font-family:verdana, arial, helvetica, sans-serif;"

示例

</td>
</tr>
<tr>
<td width="72" style="font-size:1em;font-family:verdana, arial, helvetica, sans-serif;">

字符串

</td>
<td width="468" style="font-size:1em;font-family:verdana, arial, helvetica, sans-serif;"

String str1 = parser.parseExpression("'Hello World!'").getValue(String.class);

String str2 = parser.parseExpression("\"Hello World!\"").getValue(String.class);

</td>
</tr>
<tr>
<td width="72" style="font-size:1em;font-family:verdana, arial, helvetica, sans-serif;">

数字类型

</td>
<td width="468" style="font-size:1em;font-family:verdana, arial, helvetica, sans-serif;"

int int1 = parser.parseExpression("1").getValue(Integer.class);

```
                long long1 = parser.parseExpression("-1L").getValue(long.class);
            </p>
            <p style="font-size:14px;">
                float float1 = parser.parseExpression("1.1").getValue(Float.class);
            </p>
            <p style="font-size:14px;">
                double double1 = parser.parseExpression("1.1E+2").getValue(double.class);
            </p>
            <p style="font-size:14px;">
                int hex1 = parser.parseExpression("0xa").getValue(Integer.class);
            </p>
            <p style="font-size:14px;">
                long hex2 = parser.parseExpression("0xaL").getValue(long.class);
            </p>
        </td>
    </tr>
    <tr>
        <td width="72" style="font-size:1em;font-family:verdana, arial, helvetica, sans-serif;">
            <p style="font-size:14px;">
                布尔类型
            </p>
        </td>
        <td width="468" style="font-size:1em;font-family:verdana, arial, helvetica, sans-serif;"

            <p style="font-size:14px;">
                boolean true1 = parser.parseExpression("true").getValue(boolean.class);
            </p>
            <p style="font-size:14px;">
                boolean false1 = parser.parseExpression("false").getValue(boolean.class);
            </p>
        </td>
    </tr>
    <tr>
        <td width="72" style="font-size:1em;font-family:verdana, arial, helvetica, sans-serif;">
            <p style="font-size:14px;">
                null类型
            </p>
        </td>
        <td width="468" style="font-size:1em;font-family:verdana, arial, helvetica, sans-serif;"

            <p style="font-size:14px;">
                Object null1 = parser.parseExpression("null").getValue(Object.class);
            </p>
        </td>
    </tr>
  </tbody>
</table>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
  <strong> </strong>
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
  <strong>二、算数运算表达式：</strong> SpEL支持加(+)、减(-)、乘(*)、除(/)、求余（
```

```
）、幂（^）运算。
</p>
<table cellspacing="0" cellpadding="0" border="1" style="color:#000000;font-family:Helvetic
, Tahoma, Arial, sans-serif;background-color:#FFFFFF;font-size:12px;padding:0px;margin:0px;"

    <tbody>
        <tr>
            <td width="72" style="font-size:1em;font-family:verdana, arial, helvetica, sans-serif;">
                <p style="font-size:14px;">
                    类型
                </p>
            </td>
            <td width="468" style="font-size:1em;font-family:verdana, arial, helvetica, sans-serif;"

                <p style="font-size:14px;">
                    示例
                </p>
            </td>
        </tr>
        <tr>
            <td width="72" style="font-size:1em;font-family:verdana, arial, helvetica, sans-serif;">
                <p style="font-size:14px;">
                    加减乘除
                </p>
            </td>
            <td width="468" style="font-size:1em;font-family:verdana, arial, helvetica, sans-serif;"

                <p style="font-size:14px;">
                    int result1 = parser.parseExpression("1+2-3*4/2").getValue(Integer.class);//-3
                </p>
            </td>
        </tr>
        <tr>
            <td width="72" style="font-size:1em;font-family:verdana, arial, helvetica, sans-serif;">
                <p style="font-size:14px;">
                    求余
                </p>
            </td>
            <td width="468" style="font-size:1em;font-family:verdana, arial, helvetica, sans-serif;"

                <p style="font-size:14px;">
                    int result2 = parser.parseExpression("4%3").getValue(Integer.class);//1
                </p>
            </td>
        </tr>
        <tr>
            <td width="72" style="font-size:1em;font-family:verdana, arial, helvetica, sans-serif;">
                <p style="font-size:14px;">
                    幂运算
                </p>
            </td>
            <td width="468" style="font-size:1em;font-family:verdana, arial, helvetica, sans-serif;"

                <p style="font-size:14px;">
```

```
                int result3 = parser.parseExpression("2^3").getValue(Integer.class);//8
            </p>
        </td>
    </tr>
  </tbody>
</table>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
  SpEL还提供求余（MOD）和除（DIV）而外两个运算符，与“%”和“/”等价，不区分大小写。
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
   
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
  <strong>三、关系表达式：</strong>等于（==）、不等于(!=)、大于(&gt;)、大于等于(&gt;=)小于(&lt;)、小于等于(&lt;=)，区间（between）运算，如“parser.parseExpression("1&gt;2").getalue(boolean.class);”将返回false；而“parser.parseExpression("1 between {1, 2}").getValue(bolean.class);”将返回true。
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
         between运算符右边操作数必须是列表类型，且能包含2个元素。第一个元素为开始，第二个元素为结束，区间运算是包含边界值的，即 xxx&gt;=list get(0) &amp;amp;&amp; xxx&lt;=list.get(1)。
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
         SpEL同样提供了等价的“EQ”、“NE”、“GT、“GE”、“LT”、“LE”来表示等于、不等于、大于、大于等于、小于、小于等于，不区分大小。
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
   
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
  <strong>四、逻辑表达式：</strong>且（and）、或(or)、非(!或NOT)。
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
   
</p>
<div style="font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:#FFFFFF;margin:px auto;padding:0px;border:1px solid #AABBCC;">
  <span style="font-size:medium;font-weight:bold;font-family:&quot;">java代码：</span>
  <div style="font-family:Consolas, &quot;background-color:#E7E5DC;padding:1px 0px 0px margin:18px 0px !important;">
    <div style="padding:0px 0px 0px 45px;margin:0px auto;">
      <div style="padding:3px;margin:0px;font-weight:bold;">
<pre class="prettyprint lang-java">String expression1 = "2&gt;1 and (!true or !false)";
boolean result1 = parser.parseExpression(expression1).getValue(boolean.class);
```

```
Assert.assertEquals(true, result1);

String expression2 = "2&gt;1 and (NOT true or NOT false)";
boolean result2 = parser.parseExpression(expression2).getValue(boolean.class);
Assert.assertEquals(true, result2);  </pre>
<br />
</div>
</div>
</div>

</div>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
    
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
   注：<strong>逻辑运算符不支持</strong><strong> Java</strong><strong>中的 &amp;amp; </strong><strong>和 || </strong><strong>。</strong>
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
    
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
   <strong>五、字符串连接及截取表达式：</strong>使用“+”进行字符串连接，使用“'String'[0 [index]”来截取一个字符，目前只支持截取一个，如“'Hello ' + 'World!'”得到“Hello World!”而“'Hello World!'[0]”将返回“H”。
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
    
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
   <strong>六、三目运算及Elivis</strong><strong>运算表达式：</strong>
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
   三目运算符 <strong>“表达式1?</strong><strong>表达式2:</strong><strong>表达</strong><strong>3</strong><strong>”</strong>用于构造三目运算表达式，如“2&gt;1?tre:false”将返回true；
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
   Elivis运算符<strong>“表达式</strong><strong>1?:</strong><strong>表达式</strong><strong>2</strong><strong>”</strong>从Groovy语言引入用于简化三目运算符的，当表达式1为null时则返回表达式1，当表达式1为null时则返回表达式2，简化了三目运算符方式“表达式1? 表达式:表达式2”，如“null?:false”将返回false，而“true?:false”将返回true；
</p>
```


七、正则表达式：使用“str matches regex，如“'123' matches '\\d{3}'”返回true；

八、括号优先级表达式：使用“(表达式)”构造，括号里的具有高优先级。

<h3 style="font-size:1.2em;font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:FFFFFF;">
5.3.3 类相关表达式
</h3>

一、类类型表达式：使用“T(Type)”来表示java.lang.Class实例，“Type必须是类全限定名，“java.lang”包除外，即该包下的类可以不指定包名；使用类类型表达式还可以行访问类静态方法及类静态字段。

        具体使用方法如下：

java代码：

<pre class="prettyprint lang-java">@Test
public void testClassTypeExpression() {
ExpressionParser parser = new SpelExpressionParser();
//java.lang包类访问

```
    Class&lt;String&gt; result1 = parser.parseExpression("T(String)").getValue(Class.class);
    Assert.assertEquals(String.class, result1);
    //其他包类访问
    String expression2 = "T(cn.javass.spring.chapter5.SpELTest)";
    Class&lt;String&gt; result2 = parser.parseExpression(expression2).getValue(Class.class);    A
sert.assertEquals(SpELTest.class, result2);
    //类静态字段访问
    int result3=parser.parseExpression("T(Integer).MAX_VALUE").getValue(int.class);
    Assert.assertEquals(Integer.MAX_VALUE, result3);
    //类静态方法调用
    int result4 = parser.parseExpression("T(Integer).parseInt('1')").getValue(int.class);
    Assert.assertEquals(1, result4);
}  </pre>
<pre class="prettyprint lang-java"></pre>
        <p>
            <br />
        </p>
        <p>
            <br />
        </p>
    </div>
</div>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
           对于java.lang包里的可以直接使用“T(String)”
问；其他包必须是类全限定名；可以进行静态字段访问如“T(Integer).MAX_VALUE”；也可以进行
态方法访问如“T(Integer).parseInt('1')”。
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
    <strong>二、类实例化：</strong>类实例化同样使用java关键字“new”，类名必须是全限定
，但java.lang包内的类型除外，如String、Integer。
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
     
</p>
<div style="font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:#FFFFFF;margin:
px auto;padding:0px;border:1px solid #AABBCC;">
    <span style="font-size:medium;font-weight:bold;font-family:&quot;">java代码：</span>
    <div style="font-family:Consolas, &quot;background-color:#E7E5DC;padding:1px 0px 0px
margin:18px 0px !important;">
        <div style="padding:0px 0px 0px 45px;margin:0px auto;">
            <div style="padding:3px;margin:0px;font-weight:bold;">
                <span style="color:#5C5C5C;font-weight:normal;">
<pre class="prettyprint lang-java">@Test
```

```
public void testConstructorExpression() {
    ExpressionParser parser = new SpelExpressionParser();
    String result1 = parser.parseExpression("new String('haha')").getValue(String.class);
    Assert.assertEquals("haha", result1);
    Date result2 = parser.parseExpression("new java.util.Date()").getValue(Date.class);
    Assert.assertNotNull(result2);
} </pre>
 </span>
            </div>
        </div>
    </div>
</div>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
           实例化完全跟Java内方式一样。
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
    <strong>三、instanceof</strong><strong>表达式：</strong>SpEL支持instanceof运算符
跟Java内使用同义；如“'haha' instanceof T(String)”将返回true。
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
    <strong>四、变量定义及引用：</strong>变量定义通过EvaluationContext接口的setVariable(v
riableName, value)方法定义；在表达式中使用“#variableName”引用；除了引用自定义变量，Sp
还允许引用根对象及当前上下文对象，使用“#root”引用根对象，使用“#this”引用当前上下文对
;
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
     
</p>
<div style="font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:#FFFFFF;margin:
px auto;padding:0px;border:1px solid #AABBCC;">
    <span style="font-size:medium;font-weight:bold;font-family:&quot;">java代码：</span>
    <div style="font-family:Consolas, &quot;background-color:#E7E5DC;padding:1px 0px 0px
margin:18px 0px !important;">
        <div style="padding:0px 0px 0px 45px;margin:0px auto;">
            <div style="padding:3px;margin:0px;font-weight:bold;">
                <span style="color:#5C5C5C;font-weight:normal;">
<pre class="prettyprint lang-java">@Test
public void testVariableExpression() {
```

```
    ExpressionParser parser = new SpelExpressionParser();
    EvaluationContext context = new StandardEvaluationContext();
    context.setVariable("variable", "haha");
    context.setVariable("variable", "haha");
    String result1 = parser.parseExpression("#variable").getValue(context, String.class);
    Assert.assertEquals("haha", result1);


context = new StandardEvaluationContext("haha");
String result2 = parser.parseExpression("#root").getValue(context, String.class);
Assert.assertEquals("haha", result2);
String result3 = parser.parseExpression("#this").getValue(context, String.class);
Assert.assertEquals("haha", result3);

} </pre>
 </span>
</div>
</div>
</div>

</div>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
           使用“#variable”来引用在EvaluationContext定的变量；除了可以引用自定义变量，还可以使用“#root”引用根对象，“#this”引用当前上下文对，此处“#this”即根对象。
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
    <strong>五、自定义函数：</strong>目前只支持类静态方法注册为自定义函数；SpEL使用StandardEvaluationContext的registerFunction方法进行注册自定义函数，其实完全可以使用setVariable替，两者其实本质是一样的；
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
     
</p>
<div style="font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:#FFFFFF;margin:px auto;padding:0px;border:1px solid #AABBCC;">
    <span style="font-size:medium;font-weight:bold;font-family:&quot;">java代码：</span>
    <div style="font-family:Consolas, &quot;background-color:#E7E5DC;padding:1px 0px 0px margin:18px 0px !important;">
        <div style="padding:0px 0px 0px 45px;margin:0px auto;">
            <div style="padding:3px;margin:0px;font-weight:bold;">
```

<pre class="prettyprint lang-java">@Test
public void testFunctionExpression() throws SecurityException, NoSuchMethodException {
    ExpressionParser parser = new SpelExpressionParser();
    StandardEvaluationContext context = new StandardEvaluationContext();
    Method parseInt = Integer.class.getDeclaredMethod("parseInt", String.class);
    context.registerFunction("parseInt", parseInt);
    context.setVariable("parseInt2", parseInt);
    String expression1 = "#parseInt('3') == #parseInt2('3')";
    boolean result1 = parser.parseExpression(expression1).getValue(context, boolean.class);
    Assert.assertEquals(true, result1);
} </pre>

       此处可以看出“registerFunction”和“setVariabl”都可以注册自定义函数，但是两个方法的含义不一样，推荐使用“registerFunction”方法注册自义函数。

六、赋值表达式：SpEL即允许给自定义变量赋值，也允许给跟对象赋值，直使用“#variableName=value”即可赋值：

java代码：

<pre class="prettyprint lang-java">@Test
public void testAssignExpression() {
    ExpressionParser parser = new SpelExpressionParser();
    //1.给root对象赋值
    EvaluationContext context = new StandardEvaluationContext("aaaa");
    String result1 = parser.parseExpression("#root='aaaaa'").getValue(context, String.class);
    Assert.assertEquals("aaaaa", result1);
    String result2 = parser.parseExpression("#this='aaaa'").getValue(context, String.class);
    Assert.assertEquals("aaaa", result2);

```
//2.给自定义变量赋值
context.setVariable("#variable", "variable");
String result3 = parser.parseExpression("#variable=#root").getValue(context, String.class);
Assert.assertEquals("aaaa", result3);

} </pre>
 </span>
</div>
</div>
</div>

</div>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
           使用“#root='aaaaa'”给根对象赋值，使用“"#ths='aaaa'”给当前上下文对象赋值，使用“#variable=#root”给自定义变量赋值，很简单。
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
    <strong>七、对象属性存取及安全导航表达式：</strong>对象属性获取非常简单，即使用如“a.roperty.property”这种点缀式获取，SpEL对于属性名首字母是不区分大小写的；SpEL还引入了Groovy语言中的安全导航运算符“(对象|属性)?.属性”，用来避免但“?.”前边的表达式为null时抛出空指异常，而是返回null；修改对象属性值则可以通过赋值表达式或Expression接口的setValue方法修改。
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
     
</p>
<div style="font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:#FFFFFF;margin:px auto;padding:0px;border:1px solid #AABBCC;">
    <span style="font-size:medium;font-weight:bold;font-family:&quot;">java代码：</span>
    <div style="font-family:Consolas, &quot;background-color:#E7E5DC;padding:1px 0px 0px margin:18px 0px !important;">
        <div style="padding:0px 0px 0px 45px;margin:0px auto;">
            <div style="padding:3px;margin:0px;font-weight:bold;">
                <span style="color:#5C5C5C;font-weight:normal;">
<pre class="prettyprint lang-java">ExpressionParser parser = new SpelExpressionParser();
//1.访问root对象属性
Date date = new Date();
StandardEvaluationContext context = new StandardEvaluationContext(date);
int result1 = parser.parseExpression("Year").getValue(context, int.class);
Assert.assertEquals(date.getYear(), result1);
int result2 = parser.parseExpression("year").getValue(context, int.class);
Assert.assertEquals(date.getYear(), result2);  </pre>
    </span>
            </div>
        </div>
    </div>
```


       对于当前上下文对象属性及方法访问，以直接使用属性或方法名访问，比如此处根对象date属性“year”，注意此处属性名首字不区分大小写。

java代码：

<pre class="prettyprint lang-java">//2.安全访问
context.setRootObject(null);
Object result3 = parser.parseExpression("#root?.year").getValue(context, Object.class);
Assert.assertEquals(null, result3); </pre>

       SpEL引入了Groovy的安全导航运算符，比如此处对象为null，所以如果访问其属性时肯定抛出空指针异常，而采用“?.”安全访问导航运算符将不抛指针异常，而是简单的返回null。

java代码：
<div style="font-family:Consolas, "background-color:#E7E5DC;padding:1px 0px 0px

margin:18px 0px !important;">

<pre class="prettyprint lang-java">//3.给root对象属性赋值
context.setRootObject(date);
int result4 = parser.parseExpression("Year = 4").getValue(context, int.class);
Assert.assertEquals(4, result4);
parser.parseExpression("Year").setValue(context, 5);
int result5 = parser.parseExpression("Year").getValue(context, int.class);
Assert.assertEquals(5, result5); </pre>

       给对象属性赋值可以采用赋值表达式或Expression接口的setValu方法赋值，而且也可以采用点缀方式赋值。

八、对象方法调用：对象方法调用更简单，跟Java语法一样；如“'haha'.substring(2,4)”将返回“ha”；而对于根对象可以直接调用方法；

java代码：

<pre class="prettyprint lang-java">Date date = new Date();
StandardEvaluationContext context = new StandardEvaluationContext(date);
int result2 = parser.parseExpression("getYear()").getValue(context, int.class);
Assert.assertEquals(date.getYear(), result2);</pre>

<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-

erif;">
       比如根对象date方法“getYear”可以直接调用。

九、Bean引用：SpEL支持使用“@”符号来引用Bea
，在引用Bean时需要使用BeanResolver接口实现来查找Bean，Spring提供BeanFactoryResolver实
;

<div style="font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:#FFFFFF;margin:
px auto;padding:0px;border:1px solid #AABBCC;">
java代码：
<div style="font-family:Consolas, "background-color:#E7E5DC;padding:1px 0px 0px
margin:18px 0px !important;">

<pre class="prettyprint lang-java">@Test
public void testBeanExpression() {
ClassPathXmlApplicationContext ctx = new ClassPathXmlApplicationContext();
ctx.refresh();
ExpressionParser parser = new SpelExpressionParser();
StandardEvaluationContext context = new StandardEvaluationContext();
context.setBeanResolver(new BeanFactoryResolver(ctx));
Properties result1 = parser.parseExpression("@systemProperties").getValue(context, Proper
ies.class);
Assert.assertEquals(System.getProperties(), result1);
} </pre>

       在示例中我们首先初始化了一个IoC容器，ClassPa
hXmlApplicationContext 实现默认会把“System.getProperties()”注册为“systemProperties”B
an，因此我们使用 “@systemProperties”来引用该Bean。

<h3 style="font-size:1.2em;font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:
FFFFFF;">

</h3>

<h3 style="font-size:1.2em;font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:FFFFFF;">
5.3.3 集合相关表达式
</h3>

一、内联List： 从Spring3.0.4开始支持内联List，使用{达式，……}定义内联List，如“{1,2,3}”将返回一个整型的ArrayList，而“{}”将返回空的List，对于面量表达式列表，SpEL会使用java.util.Collections.unmodifiableList方法将列表设置为不可修改。

java代码：

<pre class="prettyprint lang-java">//将返回不可修改的空List
List<Integer> result2 = parser.parseExpression("{}").getValue(List.class); </pre>

java代码：

<pre class="prettyprint lang-java">//对于字面量列表也将返回不可修改的List
List<Integer> result1 = parser.parseExpression("{1,2,3}").getValue(List.class);
Assert.assertEquals(new Integer(1), result1.get(0));
try {
result1.set(0, 2);
//不可能执行到这，对于字面量列表不可修改
Assert.fail();

```
} catch (Exception e) {
}  </pre>
<br />
            </div>
        </div>
    </div>
</div>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
     
</p>
<div style="font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:#FFFFFF;margin:
px auto;padding:0px;border:1px solid #AABBCC;">
    <span style="font-size:medium;font-weight:bold;font-family:&quot;">java代码：  </span>
    <div style="font-family:Consolas, &quot;background-color:#E7E5DC;padding:1px 0px 0px
margin:18px 0px !important;">
        <div style="padding:0px 0px 0px 45px;margin:0px auto;">
            <div style="padding:3px;margin:0px;font-weight:bold;">
<pre class="prettyprint lang-java">//对于列表中只要有一个不是字面量表达式，将只返回原始Lis
,
//不会进行不可修改处理
String expression3 = "{{1+2,2+4},{3,4+4}}";
List&lt;List&lt;Integer&gt;&gt; result3 = parser.parseExpression(expression3).getValue(List.cla
s);
result3.get(0).set(0, 1);
Assert.assertEquals(2, result3.size());  </pre>
<br />
            </div>
        </div>
    </div>
</div>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
     
</p>
<div style="font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:#FFFFFF;margin:
px auto;padding:0px;border:1px solid #AABBCC;">
    <span style="font-size:medium;font-weight:bold;font-family:&quot;">java代码：  </span>
    <div style="font-family:Consolas, &quot;background-color:#E7E5DC;padding:1px 0px 0px
margin:18px 0px !important;">
        <div style="padding:0px 0px 0px 45px;margin:0px auto;">
            <div style="padding:3px;margin:0px;font-weight:bold;">
<pre class="prettyprint lang-java">//声明一个大小为2的一维数组并初始化
int[] result2 = parser.parseExpression("new int[2]{1,2}").getValue(int[].class);  </pre>
<br />
            </div>
```

```
        </div>
    </div>
</div>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
     
</p>
<div style="font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:#FFFFFF;margin:px auto;padding:0px;border:1px solid #AABBCC;">
    <span style="font-size:medium;font-weight:bold;font-family:&quot;">java代码：</span>
    <div style="font-family:Consolas, &quot;background-color:#E7E5DC;padding:1px 0px 0px margin:18px 0px !important;">
        <div style="padding:0px 0px 0px 45px;margin:0px auto;">
            <div style="padding:3px;margin:0px;font-weight:bold;">
<pre class="prettyprint lang-java">//定义一维数组但不初始化
int[] result1 = parser.parseExpression("new int[1]").getValue(int[].class);</pre>
<br />
            </div>
        </div>
    </div>
</div>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
    二、内联数组：和Java 数组定义类似，只是在定义时进行多维数组初始化。
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
     
</p>
<div style="font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:#FFFFFF;margin:px auto;padding:0px;border:1px solid #AABBCC;">
    <span style="font-size:medium;font-weight:bold;font-family:&quot;">java代码：</span>
    <div style="font-family:Consolas, &quot;background-color:#E7E5DC;padding:1px 0px 0px margin:18px 0px !important;">
        <div style="padding:0px 0px 0px 45px;margin:0px auto;">
            <div style="padding:3px;margin:0px;font-weight:bold;">
<pre class="prettyprint lang-java">//定义多维数组但不初始化
int[][][] result3 = parser.parseExpression("new int[1][2][3]").getValue(int[][][].class);</pre>
```


java代码：

<pre class="prettyprint lang-java">//错误的定义多维数组，多维数组不能初始化
String expression4 = "new int[1][2][3]{{1}{2}{3}}";
try {
int[][][] result4 = parser.parseExpression(expression4).getValue(int[][][].class);
Assert.fail();
} catch (Exception e) {
} </pre>

三、集合，字典元素访问：SpEL目前支持所有集合类型和字典类型的元素访问，使用“集合[索引]访问集合元素，使用“map[key]”访问字典元素；

java代码：

```
        <div style="padding:0px 0px 0px 45px;margin:0px auto;">
            <div style="padding:3px;margin:0px;font-weight:bold;">
                <span style="background-color:inherit;font-weight:normal;">
<pre class="prettyprint lang-java">//SpEL内联List访问
int result1 = parser.parseExpression("{1,2,3}[0]").getValue(int.class);
//即list.get(0)
Assert.assertEquals(1, result1);  </pre>
  </span>
            </div>
        </div>
    </div>
</div>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
     
</p>
<div style="font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:#FFFFFF;margin:
px auto;padding:0px;border:1px solid #AABBCC;">
    <span style="font-size:medium;font-weight:bold;font-family:&quot;">java代码：  </span>
    <div style="font-family:Consolas, &quot;background-color:#E7E5DC;padding:1px 0px 0px
margin:18px 0px !important;">
        <div style="padding:0px 0px 0px 45px;margin:0px auto;">
            <div style="padding:3px;margin:0px;font-weight:bold;">
<pre class="prettyprint lang-java">//SpEL目前支持所有集合类型的访问
Collection&lt;Integer&gt; collection = new HashSet&lt;Integer&gt;();
collection.add(1);
collection.add(2);
EvaluationContext context2 = new StandardEvaluationContext();
context2.setVariable("collection", collection);
int result2 = parser.parseExpression("#collection[1]").getValue(context2, int.class);
//对于任何集合类型通过Iterator来定位元素
Assert.assertEquals(2, result2);  </pre>
<br />
            </div>
        </div>
    </div>
</div>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
     
</p>
<div style="font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:#FFFFFF;margin:
px auto;padding:0px;border:1px solid #AABBCC;">
    <span style="font-size:medium;font-weight:bold;font-family:&quot;">java代码：  </span>
    <div style="font-family:Consolas, &quot;background-color:#E7E5DC;padding:1px 0px 0px
margin:18px 0px !important;">
```

```
        <div style="padding:0px 0px 0px 45px;margin:0px auto;">
            <div style="padding:3px;margin:0px;font-weight:bold;">
<pre class="prettyprint lang-java">//SpEL对Map字典元素访问的支持
Map&lt;String, Integer&gt; map = new HashMap&lt;String, Integer&gt;();
map.put("a", 1);
EvaluationContext context3 = new StandardEvaluationContext();
context3.setVariable("map", map);
int result3 = parser.parseExpression("#map['a']").getValue(context3, int.class);
Assert.assertEquals(1, result3);  </pre>
<br />
            </div>
        </div>
    </div>
</div>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
           注：<strong>集合元素访问是通过</strong><st
ong>Iterator</strong><strong>遍历来定位元素位置的</strong>。
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
    <strong>四、列表，字典，数组元素修改：</strong>可以使用赋值表达式或Expression接口的se
Value方法修改；
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
     
</p>
<div style="font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:#FFFFFF;margin:
px auto;padding:0px;border:1px solid #AABBCC;">
    <span style="font-size:medium;font-weight:bold;font-family:&quot;">java代码：</span>
    <div style="font-family:Consolas, &quot;background-color:#E7E5DC;padding:1px 0px 0px
margin:18px 0px !important;">
        <div style="padding:0px 0px 0px 45px;margin:0px auto;">
            <div style="padding:3px;margin:0px;font-weight:bold;">
<pre class="prettyprint lang-java">//1.修改数组元素值
int[] array = new int[] {1, 2};
EvaluationContext context1 = new StandardEvaluationContext();
context1.setVariable("array", array);
int result1 = parser.parseExpression("#array[1] = 3").getValue(context1, int.class);
Assert.assertEquals(3, result1);  </pre>
<br />
```

```
            </div>
        </div>
    </div>
</div>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
     
</p>
<div style="font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:#FFFFFF;margin:px auto;padding:0px;border:1px solid #AABBCC;">
    <span style="font-size:medium;font-weight:bold;font-family:&quot;">java代码：  </span>
    <div style="font-family:Consolas, &quot;background-color:#E7E5DC;padding:1px 0px 0px margin:18px 0px !important;">
        <div style="padding:0px 0px 0px 45px;margin:0px auto;">
            <div style="padding:3px;margin:0px;font-weight:bold;">
<pre class="prettyprint lang-java">//2.修改集合值
Collection&lt;Integer&gt; collection = new ArrayList&lt;Integer&gt;();
collection.add(1);
collection.add(2);
EvaluationContext context2 = new StandardEvaluationContext();
context2.setVariable("collection", collection);
int result2 = parser.parseExpression("#collection[1] = 3").getValue(context2, int.class);
Assert.assertEquals(3, result2);
parser.parseExpression("#collection[1]").setValue(context2, 4);
result2 = parser.parseExpression("#collection[1]").getValue(context2, int.class);
Assert.assertEquals(4, result2);  </pre>
<br />
            </div>
        </div>
    </div>
</div>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
     
</p>
<div style="font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:#FFFFFF;margin:px auto;padding:0px;border:1px solid #AABBCC;">
    <span style="font-size:medium;font-weight:bold;font-family:&quot;">java代码：  </span>
    <div style="font-family:Consolas, &quot;background-color:#E7E5DC;padding:1px 0px 0px margin:18px 0px !important;">
        <div style="padding:0px 0px 0px 45px;margin:0px auto;">
            <div style="padding:3px;margin:0px;font-weight:bold;">
<pre class="prettyprint lang-java">//3.修改map元素值
Map&lt;String, Integer&gt; map = new HashMap&lt;String, Integer&gt;();
map.put("a", 1);
EvaluationContext context3 = new StandardEvaluationContext();
```

```
context3.setVariable("map", map);
int result3 = parser.parseExpression("#map['a'] = 2").getValue(context3, int.class);
Assert.assertEquals(2, result3);  </pre>
<br />
            </div>
        </div>
    </div>
</div>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
           对数组修改直接对“#array[index]”赋值即可修
元素值，同理适用于集合和字典类型。
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
    <strong>五、集合投影：</strong>在SQL中投影指从表中选择出列，而在SpEL指根据集合中的
素中通过选择来构造另一个集合，该集合和原集合具有相同数量的元素；SpEL使用“（list|map）.![
影表达式]”来进行投影运算：
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
     
</p>
<div style="font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:#FFFFFF;margin:
px auto;padding:0px;border:1px solid #AABBCC;">
    <span style="font-size:medium;font-weight:bold;font-family:&quot;">java代码：</span>
    <div style="font-family:Consolas, &quot;background-color:#E7E5DC;padding:1px 0px 0px
margin:18px 0px !important;">
        <div style="padding:0px 0px 0px 45px;margin:0px auto;">
            <div style="padding:3px;margin:0px;font-weight:bold;">
<pre class="prettyprint lang-java">//1.首先准备测试数据
Collection&lt;Integer&gt; collection = new ArrayList&lt;Integer&gt;();
collection.add(4);   collection.add(5);
Map&lt;String, Integer&gt; map = new HashMap&lt;String, Integer&gt;();
map.put("a", 1);    map.put("b", 2);  </pre>
<br />
            </div>
        </div>
    </div>
</div>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
```


</p>
<div style="font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:#FFFFFF;margin:
px auto;padding:0px;border:1px solid #AABBCC;">
java代码：
<div style="font-family:Consolas, "background-color:#E7E5DC;padding:1px 0px 0px
margin:18px 0px !important;">

<pre class="prettyprint lang-java">//2.测试集合或数组
EvaluationContext context1 = new StandardEvaluationContext();
context1.setVariable("collection", collection);
Collection<Integer> result1 =
parser.parseExpression("#collection.![#this+1]").getValue(context1, Collection.class);
Assert.assertEquals(2, result1.size());
Assert.assertEquals(new Integer(5), result1.iterator().next()); </pre>

<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">

</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">
       对于集合或数组使用如上表达式进行投影运算，其中投影表达式
“#this”代表每个集合或数组元素，可以使用比如“#this.property”来获取集合元素的属性，其中
#this”可以省略。
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">

</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-
erif;">

</p>
<div style="font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:#FFFFFF;margin:
px auto;padding:0px;border:1px solid #AABBCC;">
java代码：
<div style="font-family:Consolas, "background-color:#E7E5DC;padding:1px 0px 0px
margin:18px 0px !important;">

<pre class="prettyprint lang-java">//3.测试字典
EvaluationContext context2 = new StandardEvaluationContext();
context2.setVariable("map", map);
List<Integer> result2 =
parser.parseExpression("#map.![value+1]").getValue(context2, List.class);
Assert.assertEquals(2, result2.size()); </pre>


```
            </div>
        </div>
    </div>
</div>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
           SpEL投影运算还支持Map投影，但Map投影最终能得到List结果，如上所示，对于投影表达式中的“#this”将是Map.Entry，所以可以使用“value”获取值，使用“key”来获取键。
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
     
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
    <strong>六、集合选择：</strong>在SQL中指使用select进行选择行数据，而在SpEL指根据原合通过条件表达式选择出满足条件的元素并构造为新的集合，SpEL使用“(list|map).?[选择表达式]”其中选择表达式结果必须是boolean类型，如果true则选择的元素将添加到新集合中，false将不添加新集合中。
</p>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
     
</p>
<div style="font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:#FFFFFF;margin:px auto;padding:0px;border:1px solid #AABBCC;">
    <span style="font-size:medium;font-weight:bold;font-family:&quot;">java代码：</span>
    <div style="font-family:Consolas, &quot;background-color:#E7E5DC;padding:1px 0px 0px margin:18px 0px !important;">
        <div style="padding:0px 0px 0px 45px;margin:0px auto;">
            <div style="padding:3px;margin:0px;font-weight:bold;">
<pre class="prettyprint lang-java">//1.首先准备测试数据
Collection&lt;Integer&gt; collection = new ArrayList&lt;Integer&gt;();
collection.add(4);   collection.add(5);
Map&lt;String, Integer&gt; map = new HashMap&lt;String, Integer&gt;();
map.put("a", 1);    map.put("b", 2);  </pre>
<br />
            </div>
        </div>
    </div>
</div>
<p style="font-size:14px;background-color:#FFFFFF;font-family:verdana, arial, helvetica, sans-erif;">
     
</p>
```


java代码：

<pre class="prettyprint lang-java">//2.集合或数组测试
EvaluationContext context1 = new StandardEvaluationContext();
context1.setVariable("collection", collection);
Collection<Integer> result1 =
parser.parseExpression("#collection.?[#this>4]").getValue(context1, Collection.class);
Assert.assertEquals(1, result1.size());
Assert.assertEquals(new Integer(5), result1.iterator().next());</pre>

       对于集合或数组选择，如“#collection.?[#this>4]”将选择出集合元素值大于4的所有元素。选择表达式必须返回布尔类型，使用“#this”表示当前素。

java代码：

<pre class="prettyprint lang-java">//3.字典测试
EvaluationContext context2 = new StandardEvaluationContext();
context2.setVariable("map", map);
Map<String, Integer> result2 =
parser.parseExpression("#map.?[#this.key != 'a']").getValue(context2, Map.class);
Assert.assertEquals(1, result2.size());

List<Integer> result3 =

parser.parseExpression("#map.?[key != 'a'].![value+1]").getValue(context2, List.class);

Assert.assertEquals(new Integer(3), result3.iterator().next()); </pre>

       对于字典选择，如“#map.?[#this.key != 'a']”将择键值不等于”a”的，其中选择表达式中“#this”是Map.Entry类型，而最终结果还是Map，这点投影不同；集合选择和投影可以一起使用，如“#map.?[key != 'a'].![value+1]”将首先选择键值不于”a”的，然后在选出的Map中再进行“value+1”的投影。

<h3 style="font-size:1.2em;font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:FFFFFF;">

</h3>
<h3 style="font-size:1.2em;font-family:Helvetica, Tahoma, Arial, sans-serif;background-color:FFFFFF;">
5.3.4 表达式模板
</h3>

       模板表达式就是由字面量与一个或多个表达式块组。每个表达式块由“前缀+表达式+后缀”形式组成，如“${1+2}”即表达式块。在前边我们已经介了使用ParserContext接口实现来定义表达式是否是模板及前缀和后缀定义。在此就不多介绍了，如“rror ${#v0} ${#v1}”表达式表示由字面量“Error ”、模板表达式“#v0”、模板表达式“#v1”组，其中v0和v1表示自定义变量，需要在上下文定义。