SpEL

本章目标

- 掌握SpEL基本概念
- 掌握SpEL基本语法
- 掌握ExpressionParser的应用
- 掌握基于XML的SpEL应用
- 掌握基于@Value注解的应用

SpEL基本概念

- SpEL是The Spring Expression Language的简称,它是表达式计算的基础
- SpEL是强大的表达式语言，支持运行时查询、操纵一个对象图功能。SpEL语言的语法类似于EL，但提供了更多的功能，最主要的是显式方法调用和基本字符串模板函数。
- 同很多可用的Java 表达式语言相比，例如OGNL，MVEL和JBoss EL，SpEL的诞生是为了给Spring社区提供一个可以给Spring目录中所有产品提供单一良好支持的表达式语言。SpEL是一个基于技术中立的API，允许需要时与其他表达式语言集成。
- SpEL与Spring不是直接绑定关系，它可以独立存在，并应用到其它平台。

SpEL基本概念

- SpEL支持如下功能：
  - 基本表达式：字面量表达式、关系，逻辑与算数运算表达式、字符串连接及截取表达式、三目运算及Elivis表达式、正则表达式、括号优先级表达式；
  - 类相关表达式：类类型表达式、类实例化、instanceof表达式、变量定义及引用、赋值表达式、自定义函数、对象属性存取及安全导航表达式、对象方法调用、Bean引用；
  - 集合相关表达式：内联List、内联数组、集合，字典访问、列表，字典，数组修改、集合投影、集合选择；不支持多维内联数组初始化；不支持内联字典定义；
  - 其他表达式：模板表达式。

SpEL基本语法

- SpEL基本语法：
  - XML中使用 ：#{ 表达式 }
  - Bean中使用 ：@Value("#{ 表达式 }")
- 引用其它对象的属性：

```
<property name="suffix" value="#{sequenceGenerator.suffix}"></property>
```

SpEL基本语法

- SpEL支持如下运算符：
  - 算数运算符：+, -, *, /, %, ^

```
<property name="adjustedAmount" value="#{counter.total + 42}"/>
<property name="adjustedAmount" value="#{counter.total - 20}"/>
<property name="circumference" value="#{2 * T(java.lang.Math).PI * circle.radius}"/>
<property name="average" value="#{counter.total / counter.count}"/>
<property name="remainder" value="#{counter.total % counter.count}"/>
<property name="area" value="#{T(java.lang.Math).PI * circle.radius ^ 2}"/>
```

  - 加号还可以用作字符串连接

```
<constructor-arg
    value="#{performer.firstName + ' ' + performer.lastName}"/>
```

SpEL基本语法

- 比较运算符： <, >, ==, <=, >=, lt, gt, eq, le, ge

```
<property name="equal" value="#{counter.total == 100}"/>
<property name="hasCapacity" value="#{counter.total le 100000}"/>
```

- 逻辑运算符号： and, or, not

```
<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}"/>
```

- if-else 运算符：

```
<constructor-arg
    value="#{songSelector.selectSong()=='Jingle Bells'?piano:' Jingle Bells '}"/>
```

SpEL基本语法

- 正则表达式：
  - 正则表示式示例：
    - 表示6位数字的邮政编码： **/^\d{6}$/**
    - 表示11位的电话号码，首位必须为1：**/^1\d{10}$/**
    - Email格式：**/^\w+@\w+.[a-zA-Z]{2,3}$/;**

```
<constructor-arg
    value="#{admin.email matches '[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,4}'}"/>
```

表达式	说明
.	除了换行符之外的任意字符
*	匹配前面的子表达式零次或多次。
+	匹配前面的子表达式一次或多次。
?	匹配前面的子表达式零次或一次。

SpEL基本语法

表达式	说明
/.../	代表一个模式的开启和结束
^	匹配输入字符串的开始位置。
$	匹配输入字符串的结束位置。
\s	任何空白字符
\S	任何非空白字符
\d	匹配一个数字字符。等价于[0-9]。
\D	匹配一个非数字字符。等价于[^0-9]。
\w	匹配包括下划线的任何单词字符。等价于“[A-Za-z0-9_]”。
\W	匹配任何非单词字符。等价于“[^A-Za-z0-9_]”。
{n}	n是一个非负整数。匹配确定的n次。
{n,}	n是一个非负整数。至少匹配n次。
{n,m}	m和n均为非负整数，其中n<=m。最少匹配n次且最多匹配m次。

ExpressionParser

- 方法动态调用示例

```
public static void main(String[] args) {
    ExpressionParser parser = new SpelExpressionParser();
    Expression exp = parser.parseExpression("'Hello World'.concat('!')");
    String message = (String) exp.getValue();
    System.out.println(message);
}
```

使用反射机制，动态调用了String的concat方法，结果为：Hello World!

运行结果：Sat Nov 09 15:55:10 CST 2019

```
public class Test2 {

    public static void main(String[] args) {
        ExpressionParser parser = new SpelExpressionParser();
        Expression exp = parser.parseExpression("new java.util.Date()");
        Date date = (Date)exp.getValue();
        System.out.println(date.toString());
    }
```

ExpressionParser

- 动态调用bean的属性

```xml
<bean id="user" class="com.icss.entity.User">
    <property name="uname" value="tom"></property>
    <property name="pwd" value="123456"></property>
    <property name="role" value="2"></property>
</bean>
```

使用Spel的ExpressionParser调用bean比较麻烦

```java
public static void main(String[] args) {

    ApplicationContext app = new ClassPathXmlApplicationContext("beans.xml");
    User user = app.getBean(User.class);

    //解析表达式需要的上下文，解析时有一个默认的上下文
    EvaluationContext ctx = new StandardEvaluationContext();
    //在上下文中设置变量，变量名为user，内容为user对象
    ctx.setVariable("user", user);

    ExpressionParser parser = new SpelExpressionParser();
    Expression exp = parser.parseExpression("#user.getUname()");
    String uname = (String)exp.getValue(ctx);
    System.out.println(uname);
}
```

基于XML的SpEL应用案例

```
<bean id="numberGuess" class="org.spring.samples.NumberGuess">
    <property name="randomNumber"
        value="#{ T(java.lang.Math).random() * 100.0 }" />
    <!-- other properties -->
</bean>

<bean id="shapeGuess" class="org.spring.samples.ShapeGuess">
    <property name="initialShapeSeed"
        value="#{ numberGuess.randomNumber }" />
    <!-- other properties -->
</bean>

<bean id="taxCalculator" class="org.spring.samples.TaxCalculator">
    <property name="defaultLocale"
        value="#{ systemProperties['user.country'] }" />
    <!-- other properties -->
</bean>
```

配置Bean，给属性赋值

基于XML的SpEL应用案例

```java
public class NumberGuess {
    private int randomNumber;

    public int getRandomNumber() {
        return randomNumber;
    }

    public void setRandomNumber(int randomNumber) {
        this.randomNumber = randomNumber;
    }

    public void doSomething() {
        System.out.println("randomNumber=" + randomNumber);
    }
```

```java
public class ShapeGuess {
    private int initialShapeSeed;

    public int getInitialShapeSeed() {
        return initialShapeSeed;
    }

    public void setInitialShapeSeed(int initialShapeSeed) {
        this.initialShapeSeed = initialShapeSeed;
    }

    public void doSomething() {
        System.out.println("initialShapeSeed=" + initialShapeSeed);
    }
```

```java
public class TaxCalculator {
    private String defaultLocale;

    public String getDefaultLocale() {
        return defaultLocale;
    }

    public void setDefaultLocale(String defaultLocale) {
        this.defaultLocale = defaultLocale;
    }

    public void doSomething() {
        System.out.println("defaultLocale=" + defaultLocale);
    }
```

编写bean的class

基于XML的SpEL应用案例

```
public static void main(String[] args) {
    ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");

    NumberGuess  ng = (NumberGuess)context.getBean("numberGuess");
    ng.doSomething();

    ShapeGuess  sg = (ShapeGuess)context.getBean("shapeGuess");
    sg.doSomething();

    TaxCalculator  tc = (TaxCalculator)context.getBean("taxCalculator");
    tc.doSomething();

    //与SpEl中调用信息一致
    System.out.println(System.getProperty("user.country"));
}
```

测试

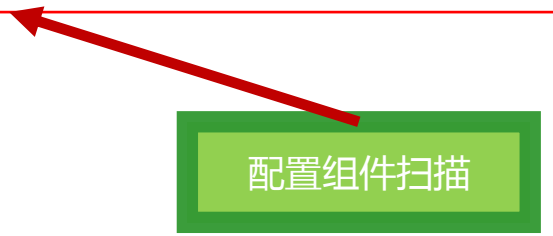
```
信息: Loading XML bean definitions from class path resource [beans.xml]
randomNumber=92
initialShapeSeed=92
defaultLocale=CN
CN
```

基于@Value注解的应用

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">

    <context:component-scan base-package="org.spring.samples"/>

</beans>
```



基于@Value注解的应用

```java
@Component
public class NumberGuess {

    @Value("#{ T(java.lang.Math).random() * 100.0 }")
    private int randomNumber;


    public int getRandomNumber() {
        return randomNumber;
    }

    public void setRandomNumber(int randomNumber) {
        this.randomNumber = randomNumber;
    }

    public void doSomething() {
        System.out.println("randomNumber=" + randomNumber);
    }
```

使用@Value配合SpEL赋值

```java
@Component
public class ShapeGuess {
    @Value("#{ numberGuess.randomNumber }")
    private int initialShapeSeed;

    public int getInitialShapeSeed() {
        return initialShapeSeed;
    }

    public void setInitialShapeSeed(int initialShapeSeed) {
        this.initialShapeSeed = initialShapeSeed;
    }

    public void doSomething() {
        System.out.println("initialShapeSeed=" + initialShapeSeed);
    }
```

```java
@Component
public class TaxCalculator {

    @Value("#{ systemProperties['user.country'] }")
    private String defaultLocale;

    public String getDefaultLocale() {
        return defaultLocale;
    }

    public void setDefaultLocale(String defaultLocale) {
        this.defaultLocale = defaultLocale;
    }

    public void doSomething() {
        System.out.println("defaultLocale=" + defaultLocale);
    }
```

基于@Value注解的应用

信息: Loading XML bean definitions from class path resource [beans.xml]

randomNumber=2

initialShapeSeed=2

defaultLocale=CN

CN