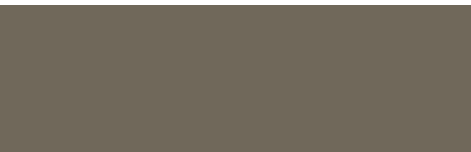


Spring Expression Language



What is SpEL?

- ❑ is a powerful expression language
- ❑ much like OGNL, Jboss EL, etc.
- ❑ supports querying and manipulating an object graph at runtime
- ❑ can be used across all the products in the Spring portfolio
- ❑ can be used outside of Spring

Features



- ❑ expressions
- ❑ accessing properties, arrays, etc.
- ❑ assignment
- ❑ method invocation
- ❑ collection selection & projection
- ❑ etc.

Fundamentals

- ExpressionParser
- Expression
 - getValue
 - setValue
- EvaluationContext
 - root
 - setVariable
 - propertyAccessor

Expression access

- configuration XML/ @Value
 - `#{expression}`
- programming
 - `parser.parseExpression("expression for root")`
 - `parser.parseExpression("#expression for variable")`
- custom template
 - `parser.parseExpression("it is #{expression}")`

Using SpEL

XML

```
<bean id="systemConfig" class="org.training.spel.SystemConfig">  
  <property name="operatingSystem"  
    value="#{systemProperties['os.name']}" />  
  
  <property name="javaVersion"  
    value="#{systemProperties['java.vm.version']}" />  
</bean>
```

@Value

```
public class SystemConfig {  
  
    @Value("#{systemProperties['java.vm.version']}")  
    private String operatingSystem;  
  
    @Value("#{systemProperties['java.vm.version']}")  
    private String javaVersion;  
  
}
```

Note: `<context:annotation-config/>`

Expressions

Literal expressions

```
ExpressionParser parser = new SpelExpressionParser();

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

parser.parseExpression("6.0221415E+23").getValue(Double.class);

parser.parseExpression("0x7FFFFFFF").getValue(Integer.class);

parser.parseExpression("'2011/01/17'").getValue(Date.class);

parser.parseExpression("true").getValue();

parser.parseExpression("null").getValue();
```

Type conversion

□ Converter

```
public interface Converter<S, T> {  
    T convert(S source);  
}
```

□ ConversionService

- <http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/validation.html#core-convert>

Object properties

- `#{person.name}`
- `#{person.Name}`
- `#{person.getName()}`

Collections

- `#{list[0]}`
- `#{list[0].name}`
- `#{map['key']}`

Methods

- `#{"Some Text".substring(0, 2)}`
- `#{"Some Text".startsWith('text')}`
- `#{"variable.toString()"}`

Relational operators

- `#{5 == 5}` or `#{5 eq 5}`
- `#{'black' > 'block'}` or `#{'black' gt 'block'}`
- `#{'text' instanceof T(int)}`
- `#{'5.00' matches '^-\?\d+(\.\d{2})?$'}`

Arithmetic operators

- $\#{5 + 5}$
- $\#{(5 + 5) * 2}$
- $\#{17 / 5 \% 3}$

- $\#{\text{'Hello'} + \text{' ' + 'world'}}$

Logical operators

- `#{true or false}`
- `#{!true}`
- `#{not isUserInGroup('admin')}`

Assignment

```
SimpleBean dima = new SimpleBean("Dima", 26);  
EvaluationContext context = new StandardEvaluationContext(dima);  
  
parser.parseExpression("name").setValue(context, "Dmitry");  
  
parser.parseExpression("age=27").getValue(context);
```

Type operator

- `#{T(java.util.Date)}`
- `#{T(String)}`
- `#{T(int)}`

- accessing static class members
 - `#{T(Math).PI}`
 - `#{T(Math).random()}`

instanceof

- `#{'text' instanceof T(String)}`
- `#{27 instanceof T(Integer)}`
- `#{false instanceof T(Boolean)}`

Constructor

- `#{new org.training.spel.Person('Misha', 28)}`
- `#{list.add(new org.training.spel.Person())}`

Variable registration

```
Map<String, Person> map = new HashMap<String, Person>();  
map.put("Dima", new Person("Dima", 27));  
map.put("Anya", new Person("Anya", 23));
```

```
ExpressionParser parser = new SpelExpressionParser();  
StandardEvaluationContext ctx = new StandardEvaluationContext();  
ctx.setVariable("map", map);  
ctx.setVariable("anya", "Anya");
```

```
parser.parseExpression("#map['Dima']").getValue(ctx);  
parser.parseExpression("#map[#anya]").getValue(ctx);
```

If-then-else

- `#{person.age>50 ? 'Old' : 'Young'}`
- `#{person.name ? : 'N/A'}`

Safe navigation

- `#{address.city?.name}`
`}`
- `#{person.name?.length()}`
`}`

Collection selection

- select all
 - `#{list.?[age>20]}`
 - `#{list.?[name.startsWith('D')]}`
- select first
 - `#{list.^[age>20]}`
- select last
 - `#{list.$[getAge()>20]}`

Collection projection

- select the names of all elements
 - `#{list.![name]}`
- select the names length of all elements
 - `#{list.![name.length()]}`

Functions

```
ExpressionParser parser = new SpelExpressionParser();
EvaluationContext context = new StandardEvaluationContext();

context.registerFunction("max", Collections.class.
    getDeclaredMethod("max", new Class[]{Collection.class}));

parser.parseExpression("#max(#list.![age])").getValue(context);
```

Templating

```
ExpressionParser parser = new SpelExpressionParser();  
  
String value = parser.parseExpression(  
    "Random number is #{T(java.lang.Math).random()}",  
    new TemplateParserContext()  
) .getValue(String.class);
```

But:

```
parser.parseExpression("#{#primes.[#this>10]}", ...)
```

#root and #this

□ array of integer

```
list.addAll(Arrays.asList(2,3,5,7,11,13,17));  
p.parseExpression("#list.?[#this>10]").getValue(context);
```

□ list of age

```
List<Person> list = new ArrayList<Person>();  
p.parseExpression("#list.![age].?[#this>20]").getValue(context);
```

Using root object

□ unchanging

```
StandardEvaluationContext context = new  
    StandardEvaluationContext(new Person("Dima", 25));  
parser.parseExpression("name").getValue(context);
```

□ changing

```
parser.parseExpression("name").getValue(new Person("Dima", 27));
```

□ cached context

```
StandardEvaluationContext context = new  
    StandardEvaluationContext(new Person("Dima", 25));  
parser.parseExpression("name").getValue(context, person1);  
parser.parseExpression("name").getValue(context, person2);
```

Access to Spring context

```
<bean id="simpleBean" class="org.training.spel.Person"  
      p:name="Misha" p:age="#{25+23}"/>
```

```
ApplicationContext context =  
    new ClassPathXmlApplicationContext("context.xml");  
Person bean = context.getBean(Person.class);  
  
ExpressionParser parser = new SpelExpressionParser();  
StandardEvaluationContext evaluation =  
    new StandardEvaluationContext(context);  
evaluation.addPropertyAccessor(new BeanFactoryAccessor());  
  
parser.parseExpression("simpleBean").getValue(evaluation);
```

Wiring properties

□ simple

```
@Value("#{systemProperties['locale']}")  
private Locale locale;
```

□ default

```
@Value("#{systemProperties['locale']?:'RU'}")  
private Locale locale;
```

□ selective

```
@Value("#{systemProperties['level']>2 ? gold : default}")  
private AccountRepository repository;
```