Spring - Expression Language (SpEL)

Last Updated : 23 Jul, 2025

Spring Expression Language (SpEL) is a feature of the Spring framework that enables querying and manipulating object graphs at runtime. It can be used in both XML and annotation-based configurations, offering flexibility for developers.

- **Dynamic Expression Evaluation**: SpEL allows evaluation of expressions at runtime, enabling dynamic behavior
- **Method Invocation**: It supports calling methods on objects within expressions.
- **Property Access**: You can access and modify the properties of objects directly.
- **String Manipulation**: SpEL allows string templating and manipulation, making it versatile for text processing.
- **Support for Collections**: It provides powerful features for querying and manipulating collections like lists, maps, and sets.

SpEL API Components:

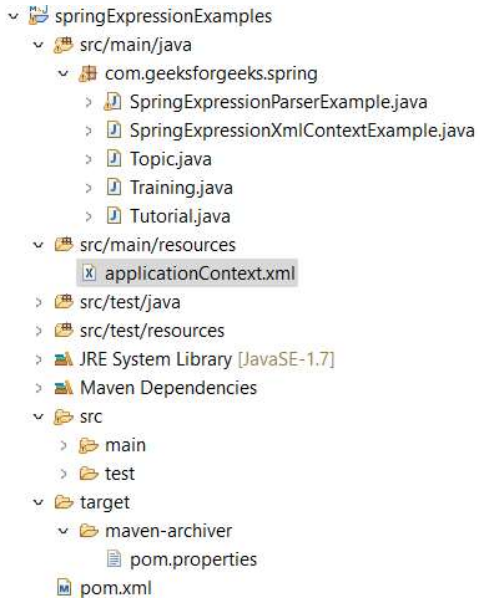
- Expression (Interface)
- ExpressionParser (Interface)
- SpelExpressionParser (Class, implementation of ExpressionParser)
- EvaluationContext (Interface)
- StandardEvaluationContext (Class, implementation of EvaluationContext)

Note: To use SpEL in the Sping project, we need to include the following dependencies (spring-core, spring-context,spring-beans) in our pom.xml file.

```
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-beans</artifactId>
        <version>${spring.version}</version>
    </dependency>
</dependencies>
```

Project Structure

The below diagram demonstrates the project structure



The project structure includes the following key files which are listed below:

- **Topic.java:** Represents a topic with a name.
- **Tutorial.java**: Represents a tutorial with a list of topics.
- **Training.java**: Represents a training program with a topic.

- **applicationContext.xml**: Configuration file that uses SpEL to define beans.

Spring Context XML Expression Support

Spring allows using expressions in XML context to simplify configuration. For example, in a training program, a lesson covers multiple technical subjects, with the course focusing on a specific subject. We will illustrate this using POJO classes.

Topic.java:

```
package com.geeksforgeeks.spring;

public class Topic {
    private String name;

    public String getName() {
        return name;

    }
    public void setName(String name) {
        this.name = name;

    }
    public String toString() {
        return name;

    }
}
```

Tutorial.java:

```
// Java Program to Illustrate Tutorial Class
package com.geeksforgeeks.spring;

// Importing required classes
import java.util.ArrayList;
import java.util.List;

// Class
public class Tutorial {

    // Class data members
    private String name;
```

```
    private List<?> topicsList = new ArrayList<>();

    // Getter
    public String getName() {
        return name;

    }

    // Setter
    public void setName(String name) {
        this.name = name;

    }

    // Method
    public List<?> getTopicsList() {
        return topicsList;

    }

    // Setter
    public void setTopicsList(List<?> topicsList)
    {
        this.topicsList = topicsList;
    }

    // Method
    // Overloading toString() method
    public String toString() {
        return name + topicsList;
    }
}
```

Training.java:

```
// Java Program to Illustrate Training Class

package com.geeksforgeeks.spring;

// Class
public class Training {

    // Class data member
    private Topic topic;

    // Getter
    public Topic getTopic() { return topic; }

    // Setter
    public void setTopic(Topic topic)
    {
        this.topic = topic;
```

```
    }
}
```

We shall define a couple of technical subjects, java core, and ScalaBasics, as well as a lesson and training, in the context of XML. When defining the beans, SpEL expressions can be combined with XML. #expression string> is the syntax.

#tutorial.topicsList[1] is used to set the topic of our training bean. It sets the subject property to the second topic from the instructional bean's list of topics.

applicationContext.xml:

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans/"
    xmlns:xsi="https://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-2.5.xsd">

     <context:component-scan base-package="com.geeksforgeeks.spring" />

    <bean id="tutorial" class="com.geeksforgeeks.spring.Tutorial">
        <property name="topicsList">
           <ref local="javaCore" />
           <ref local="scalaBasics" />
        </property>
    </bean>

    <bean id="javaCore" class="com.geeksforgeeks.spring.Topic">
        <property name="name" value="JavaCore" />
    </bean>

    <bean id="scalaBasics" class="com.geeksforgeeks.spring.Topic">
        <property name="name" value="ScalaBasics" />
    </bean>

    <bean id="training" class="com.geeksforgeeks.spring.Training">
        <property name="topic" value="#{tutorial.topicsList[1]}"/>
    </bean>

</beans>
```

SpringExpressionXmlContextExample.java:

```java
// Java Program to Illustrate Application Class

package com.geeksforgeeks.spring;

// Importing required classes
import org.springframework.context.support.ClassPathXmlApplicationContext;

// Main Class
public class SpringExpressionXmlContextExample {

    // Main driver method
    public static void main(String[] args)
    {
        ClassPathXmlApplicationContext context
            = new ClassPathXmlApplicationContext(
                "applicationContext.xml");

        // Try block to check for exceptions
        try {
            Training training
                = (Training)context.getBean("training");
            System.out.println(training.getTopic());

            System.out.println(training.getDefaultTopic());
        }

        // finally block that will execute for sure
        finally {

            // Closing the connections
            // using close() method
            context.close();
        }
    }
}
```

Output:

```
ScalaBasics
```

Annotation-Based Configuration in Spring Expressions

SpEL expressions may also be used to define beans using annotation-based configuration metadata. To define a default value, use the @Value annotation on fields, methods, and method/constructor arguments.

We've added a new member defaultTopic to our Training bean to show the example. To set the defaultTopic, we utilized a spring expression.

Training.java:

```
package com.geeksforgeeks.spring;

import org.springframework.beans.factory.annotation.Value;

public class Training
{
    private Topic topic;

    @Value("#{tutorial.topicsList[0]}")
    private Topic defaultTopic;

  //getters and setters
    public Topic getTopic()
    {
        return topic;
    }

    public void setTopic(Topic topic)
    {
        this.topic = topic;
    }

    public Topic getDefaultTopic()
    {
        return defaultTopic;
    }
}
```

The annotations are scanned and assessed by adding the context: the component-scan element to the context XML.

applicationContext.xml:

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans/"
    xmlns:xsi="https://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-2.5.xsd">

     <context:component-scan base-package="com.geeksforgeeks.spring" />
```

```
    <bean id="tutorial" class="com.geeksforgeeks.spring.Tutorial">
        <property name="topicsList">
          <ref local="javaCore" />
          <ref local="scalaBasics" />
        </property>
    </bean>

    <bean id="javaCore" class="com.geeksforgeeks.spring.Topic">
        <property name="name" value="JavaCore" />
    </bean>

    <bean id="scalaBasics" class="com.geeksforgeeks.spring.Topic">
        <property name="name" value="ScalaBasics" />
    </bean>

    <bean id="training" class="com.geeksforgeeks.spring.Training">
        <property name="topic" value="#{tutorial.topicsList[1]}"/>
    </bean>

</beans>
```

Let's see whether defaultTopic is enabled.

SpringExpressionXmlContextExample.java:

```
package com.geeksforgeeks.spring;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SpringExpressionXmlContextExample {
    public static void main(String[] args) {
        ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext(
                "applicationContext.xml");
        try {
            Training training = (Training) context.getBean("training");
            System.out.println(training.getTopic());

            System.out.println(training.getDefaultTopic());
        } finally {
            context.close();
        }
    }
}
```

Output:

```
ScalaBasics
JavaCore
```

Spring Expression Language Parser Example

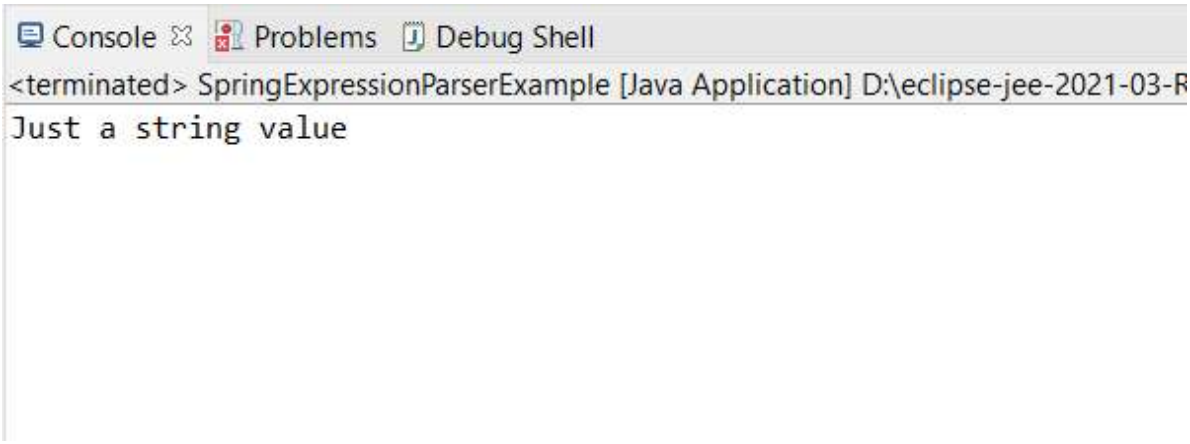
Using the expression parser, the SpEL may also be utilized as a standalone component.

> *org.springframework.expression.spel.standard.SpelExpressionParser*
> .

SpringExpressionParserExample.java:

```
package com.geeksforgeeks.spring;

import org.springframework.expression.Expression;
import org.springframework.expression.spel.standard.SpelExpressionParser;

public class SpringExpressionParserExample {
    public static void main(String[] args) {
        SpelExpressionParser parser = new SpelExpressionParser();
        Expression exp = parser.parseExpression("'Just a string value'");
        String message = (String) exp.getValue();
        System.out.println(message);

        System.out.println(parser.parseExpression("'Just a string
value'.substring(5)").getValue());

        System.out.println(parser.parseExpression("'Just a string
value'.length()").getValue());

        System.out.println(parser.parseExpression("'Just a string
value'.substring('Just '.length())").getValue());

        System.out.println(parser.parseExpression("'Just a string
value'.class").getValue());

        System.out.println(parser.parseExpression("'Just a string
value'.bytes").getValue());

        System.out.println(parser.parseExpression("new
com.geeksforgeeks.spring.Topic('Java')").getValue(Topic.class).getClass());
    }
}
```

Output:

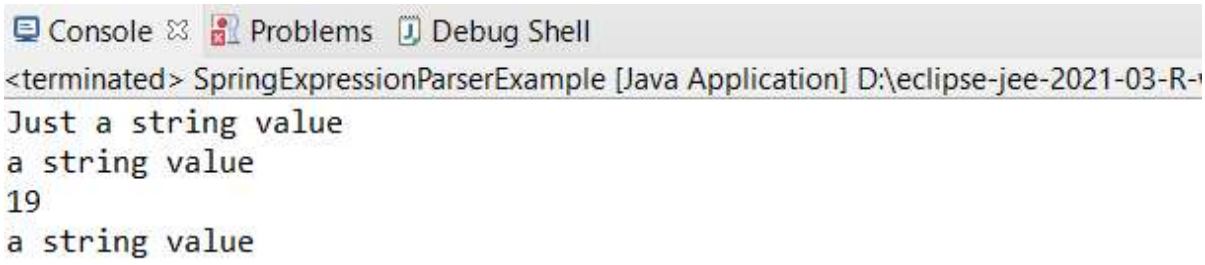


Let's have a look at a couple more instance: Using Spring ExpressionParser to call a method where in this example, we call String's substring(), length().

SpringExpressionParserExample.java:

```
package com.geeksforgeeks.spring;

import java.util.Arrays;

import org.springframework.expression.EvaluationContext;
import org.springframework.expression.Expression;
import org.springframework.expression.spel.standard.SpelExpressionParser;
import org.springframework.expression.spel.support.StandardEvaluationContext;

public class SpringExpressionParserExample {
    public static void main(String[] args) {
        SpelExpressionParser parser = new SpelExpressionParser();
        Expression exp = parser.parseExpression("'Just a string value'");
        String message = (String) exp.getValue();
        System.out.println(message);

        System.out.println(parser.parseExpression("'Just a string
value'.substring(5)").getValue());

        System.out.println(parser.parseExpression("'Just a string
value'.length()").getValue());

        System.out.println(parser.parseExpression("'Just a string
value'.substring('Just '.length())").getValue());
    }
}
```

Output:



Access JavaBean Properties using Spring ExpressionParser

Let's access a couple of JavaBean properties classes and bytes of the String objects.

SpringExpressionParserExample.java:

```
// Java Program to Illustrate Spring Expression Parser

package com.geeksforgeeks.spring;

// Importing required classes
import java.util.Arrays;
import org.springframework.expression.EvaluationContext;
import org.springframework.expression.Expression;
import org.springframework.expression.spel.standard.SpelExpressionParser;
import org.springframework.expression.spel.support.StandardEvaluationContext;

// Class
public class SpringExpressionParserExample {

    // Main driver method
    public static void main(String[] args)
    {

        SpelExpressionParser parser
            = new SpelExpressionParser();
        Expression exp = parser.parseExpression(
            "'Just a string value'");
        String message = (String)exp.getValue();

        // Print commands
        System.out.println(message);

        System.out.println(
            parser
                .parseExpression(
                    "'Just a string value'.substring(5)")
```

```
                .getValue());

        System.out.println(
            parser
                .parseExpression(
                    "'Just a string value'.length()")
                .getValue());

        System.out.println(
            parser
                .parseExpression(
                    "'Just a string value'.substring('Just '.length())")
                .getValue());

        System.out.println(
            parser
                .parseExpression(
                    "'Just a string value'.class")
                .getValue());

        System.out.println(
            parser
                .parseExpression(
                    "'Just a string value'.bytes")
                .getValue());
    }
}
```

Output:

Calling Constructor using Spring ExpressionParser

We can create an object by calling the constructor of the class in the spring expression. For example:

'new com.geeksforgeeks.spring.Topic('Java')'

SpringExpressionParserExample.java:

```
// Java Program to Illustrate Spring Expression Parser

package com.geeksforgeeks.spring;

// Importing required classes
import java.util.Arrays;
import org.springframework.expression.EvaluationContext;
import org.springframework.expression.Expression;
import org.springframework.expression.spel.standard.SpelExpressionParser;
import org.springframework.expression.spel.support.StandardEvaluationContext;

// Class
public class SpringExpressionParserExample {

    // Main driver method
    public static void main(String[] args)
    {

        SpelExpressionParser parser
            = new SpelExpressionParser();
        Expression exp = parser.parseExpression(
            "'Just a string value'");
        String message = (String)exp.getValue();
        System.out.println(message);

        // Print commands
        System.out.println(
            parser
                .parseExpression(
                    "'Just a string value'.substring(5)")
                .getValue());

        System.out.println(
            parser
                .parseExpression(
                    "'Just a string value'.length()")
                .getValue());

        System.out.println(
            parser
                .parseExpression(
                    "'Just a string value'.substring('Just '.length())")
                .getValue());

        System.out.println(
            parser
                .parseExpression(
                    "'Just a string value'.class")
                .getValue());

        System.out.println(
            parser
                .parseExpression(
```

```
                    "'Just a string value'.bytes")
                .getValue());

        System.out.println(
            parser
                .parseExpression(
                    "new com.geeksforgeeks.spring.Topic('Java')")
                .getValue(Topic.class)
                .getClass());
    }
}
```

Output: