**Spring Expression Languages (SpEL) - Introduction**

Spring Expression Language, usually referred as SpEL is one of the modules available under Spring’s Core Module. As the name suggests, it is a powerful expression language mainly used in Spring Framework for evaluating expressions such as Mathematical, Relational & Logical Operations, Conditional & Regular Expressions etc., at runtime.

It can be used by means of XML Configurations as well as Annotation based Configurations.

It is to be noted that there are many Expression Languages available for Java such as JBoss EL (Expression Language), OGNL (Object-Graph Navigation Language) & MVEL (MVFLEX Expression Language) etc., Spring Expression Language was created to satisfy the developers in Spring Community that can be used & easily coupled with other Spring Products such as Thymeleaf (Spring’s Template Engine)

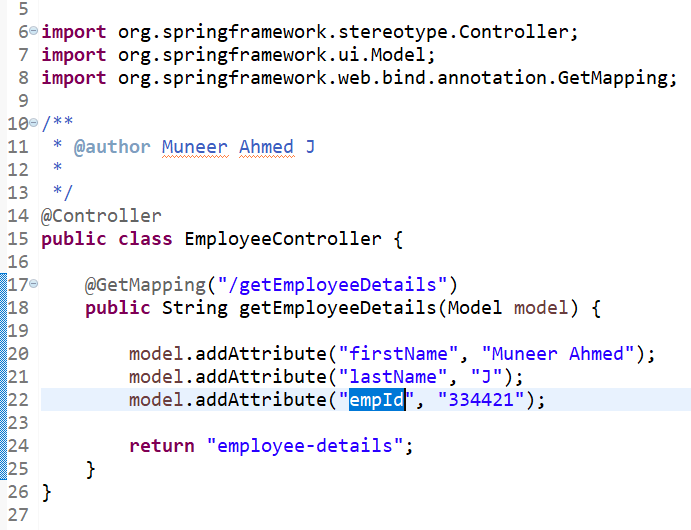
SpEL couples well along with Spring Products, but it is not tightly coupled to Spring i.e., it can also be used independently. This can be achieved by creating bootstrapping few infrastructure classes by making use of Expression Parser Libraries.

**Need of Spring Expression Language**

As we came through, Spring Expression Language (SpEL) is very powerful in providing support for modifying the object value at run time.

Though SpEL provides support for evaluating expression, it provides immense support in rendering the view layers using JSP (Java Server Pages) or Thymeleaf Template Engine as well as in querying the values from various properties file (such as application.properties) in Spring Boot framework.

Let us see a practical example of using SpEL in view layer using Thymeleaf.





**employee-details.html**

From the above code snippets, when a request is made for /**getEmployeeDetails**, we are assigning the values of employee details in model object & returning **employee-details.html** to the browser.

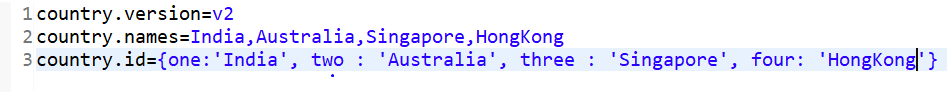
In employee-details.html, by making use of Spring Expression Language (SpEL) we are performing null checks of Employee attributes. So, only if the attributes are not null, we are allowing the browser to render & print the values.

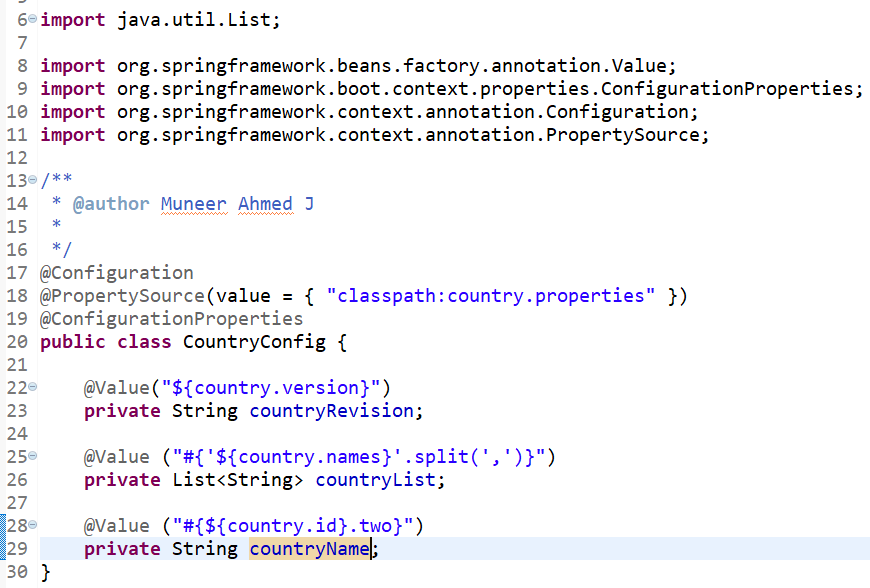
On Line 10, **th:if="${firstName != null}"** – with this, we are performing null check for firstName attribute. If the firstName attribute is not null, then we are printing the value of employee’s first name by using, **th:text="${firstName}"**.

**Note**: Here th: refers to Thymeleaf template engine.

Let us see another real time example for querying & manipulating the values using SpEL in Spring Boot Application.

**country.properties**





From the above code snippet, we could see that in **CountryConfig** class, by making use of **@Value** annotation, we are fetching the property values from **country.properties** file.

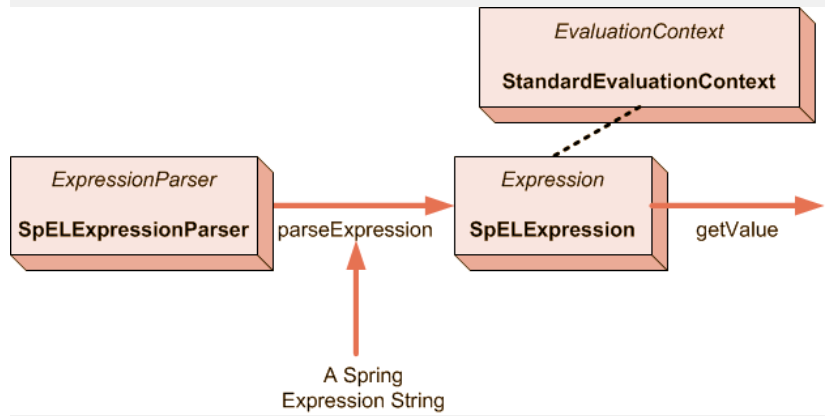
On Line 25, **@Value ("#{'${country.names}'.split(',')}")** we could see that we are making use of SpEL to query the value from country.properties & splitting it based on comma(,) and assigning it to a countryList variable.

**Output of countryList: [India, Australia, Singapore, HongKong]**

Similarly, on Line 28, **@Value ("#{${country.id}.two}")**, we are making use of SpEL to query the value from a Map in country.properties file and taking out the value which is matching the key value as two.

**Output of countryName: Australia**

**How Spring Expression Language works?**



**Reference:** <https://data-flair.training/>

SpEL as an expression language evaluates the given expression at run time & these results are injected into the browser for rendering or into other beans.

Internally, SpEL is a collection of classes which Spring uses for parsing & evaluating the given expression. Mainly, the **parseExpression()** method on the **SpELExpressionParser** takes in the input expression & returns an **SpELExpression** object. This returned object is finally used to evaluate the expression by making a call to **getValue()**.

SpEL provides supports for both XML & annotation-based configurations. In advanced version, such as in Spring Boot, **@value** is being used to evaluate the expression.

Some of the interfaces & classes available in SpEL are,

* Expression (Interfaces)
* SpelExpression (class)
* ExpressionParser (Interface)
* SpelExpressionParser (class)
* EvaluationContext (Interface)
* StandardEvaluationContext (class)

**Features of Spring Expression Language**

Spring Expression Language offers good number features for evaluating expression such as

* Literal Expression
* Mathematical Operations
* Relational Operations
* Logical Operations
* Ternary Operations
* Elvis Operations
* Regular Expressions

Let us see in detail about these operations.

**Mathematical Operations**

Mathematical Operations such as Addition, Subtraction, Multiplication, Divisions & Modulus are supported by SpEL.

**Relational Operations**

Relational Operations such as equals (==), not equals(!=), greater than (>), greater than or equals (>=), less than (<) less than or equals to are supported in SpEL.

**Logical Operations**

Logical Operations such as AND (&&), OR (||) & NOT(!) are supported in SpEL.

**Ternary Operations**

Ternary Operations used for evaluating if-else-then conditions are supported in SpEL.

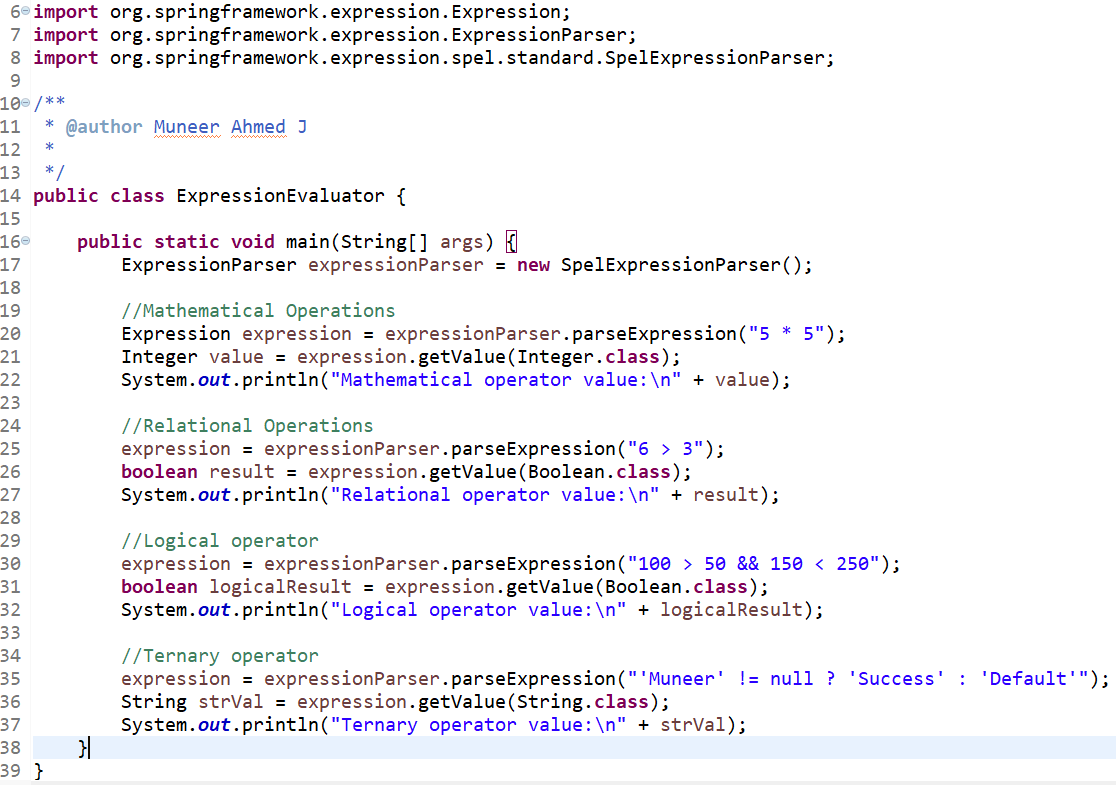
**Regular Expression**

Regular Expressions which are used for checking whether the given string matches a given regular expression or not are supported in SpEL.

**Literal Expression**

Literal Expressions for manipulating the string operations such as concat, trim etc., are supported by SpEL.

Let us witness few real time examples for the above operations.



**Output:**

|  |
| --- |
| **Mathematical operator value**: 25  **Relational operator value**: true  **Logical operator value**: true  **Ternary operator value**: Success |

From the code snippet, we can interpret that by instantiating **SpELExpressionParser**, SpEL can be used for evaluating expressions & provide results for Mathematical, Logical, Relational & Ternary Operations.

**Advantages of Spring Expression Language**

Some of the advantages of SpEL are as follows.

* SpEL provides supports for querying & manipulating the object value at run time.
* SpEL provides supports for both XML Configurations as well as Annotation based Configurations
* SpEL provides a great support in View Layers such as Thymeleaf (Spring’s Template Engine) and JSP for evaluating & manipulating the objects and values to be rendered in the browser.
* With SpEL, we can perform various operations such as Mathematical, Logical, Relational, Conditional & Ternary Operations as well as perform Literal & Regular Expression.
* Though it is one of the modules in Spring Core, it is not tied up to Spring. SpEL can also be used independently by making use of bootstrapping infrastructure class such as Parser.
* Syntax used in View Layer are very simple & easy to grasp by Developers.

**Conclusion**

From this article, we have covered the basics of Spring Expression Language with real time examples for justifying the need of Spring’s Expression Language in View and Service Layer. Also, we have looked at the working principle of SpEL & its API’s, core features & advantages of SpEL with real time examples & code snippets.