Leveraging the Power of SpEL: Advanced SpEL Expressions

Buddhini Samarakkody
JAVA DEVELOPER/INDEPENDENT CONSULTANT
www.buddhini-samarakkody.mystrikingly.com

Overview

The most important & exciting part!

Apply the magic of @Value

Power of collection manipulation & expression templates for

- Local based rendering of Order details :E-Commerce application

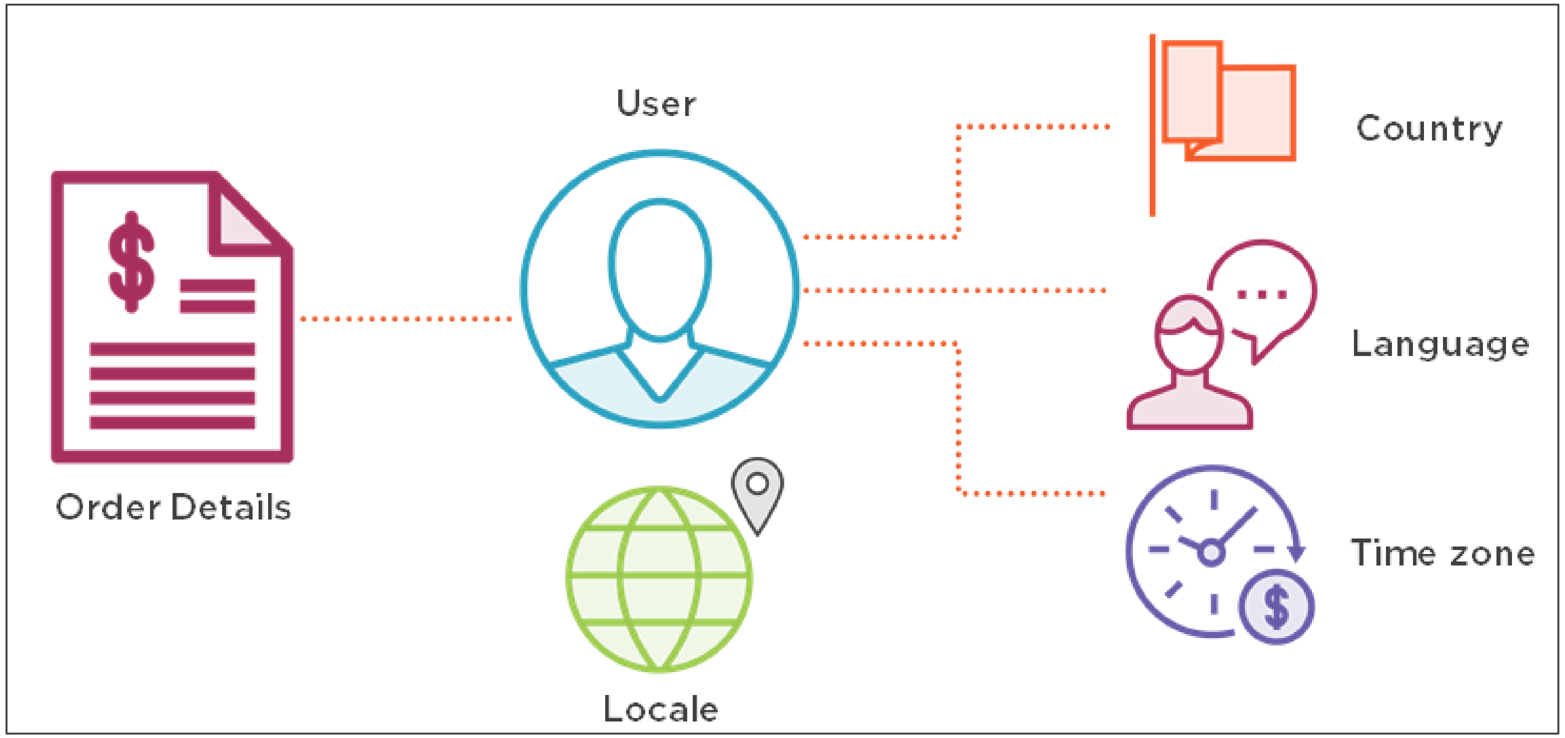
Usages with XML

Practical scenarios

Dynamic DI by manipulating objects/ object graphs at runtime

The E-commerce Application

@Value



Templating

Property wiring

Collections

The @Value Annotation

What Is It?

A Spring annotation

Placed in

- fields
- methods
- constructor parameters

To specify a default value

Used with SpEL

```java
@Value("#{'John Doe'}")

private String name;

@Value("#{30}")

private int age;
```

Usage in Fields

Ex:- Literal string and literal integer

```
@Value("#{ systemProperties['user.timezone'] }")
    public void setTimeZone(String timeZone) {
    this.timeZone = timeZone;
}
```

Usage in Method

Ex:- In setter method

```
@Autowired

public User(
  @Value("#{systemProperties['user.country']}") String country,
  @Value("#{systemProperties['user.language']}") String language) {
    this.country = country;

    this.language = language;

}
```

Usage in Constructor Parameter

Ex:- Initializing bean properties

We will see more usages in the demo!

Demo

SpEL with `@Value`: wire bean properties

- Literal expressions
- Basic operators
- Call bean properties & methods
- Setter method
- Constructor parameters

Steps

- Add an Order bean to the application
- Write expressions in User & Order beans
- Add RestController class & services
- Call the RESTful APIs

Collection Manipulation with SpEL

Accessing collections in expressions

Lists & Maps

Manipulating collections in expressions

- Collection selection

Selection operator is - .?

Filter the collection and return a new collection

Demo

SpEL: Accessing & Manipulating Collections

- Lists
- Maps

Collection selection

Steps

- Add Shipping & City beans
- Write collection manipulation expressions in Order bean
- Add RESTful services to controller
- Call the RESTful APIs

Expression Templating

What Is It?

Mixing literal text with evaluation blocks

Delimit each evaluation block with #{}

Concatenates literal text with results of evaluating expression block(s)

How Its Done?

Mix literal text with evaluation blocks

```
“literal text #{} literal
text #{}”
```

Write expressions in evaluation blocks

```
“literal text #{exp1}
literal text #{exp2}”
```

```
parser.parseExpression("#{name} your order total is",
                       new TemplateParserContext());
```

Expression Template: In Plain Java Code

Pass the `ParserContext` **as** `TemplateParserContext` **to the** `parseExpression()` **method.**

The Default TemplateParserContext

```
public class TemplateParserContext implements ParserContext {
  public String getExpressionPrefix() {
    return "#{";
  }
  public String getExpressionSuffix() {
    return "}";
  }
  public boolean isTemplate() {
    return true;
  }
}
```

You can define your own prefix and suffix characters as well!

Demo

Expression Templating

Steps

- Add property **orderSummary** to Order bean
- Use Expression Templating to populate it
- Add RESTful service to controller
- Run and check the output

Demo

SpEL with XML

Steps

- Create Spring Boot project with XML configuration
- Wire User and Order bean properties
- Collection manipulations
- Expression templating

Typical Usages of SpEL

When Can We Use SpEL?

Dependency inject existing bean in to another object

Dependency inject a bean based on environmental condition

Access & manipulate object graphs at run time

Summary

The beauty of SpEL

For DI at run time

- write & parse expressions in plain Java
- role of Evaluation Context
- the magic of @Value
- power of collection manipulation & expression templating

Excellent choice for DI in conditional situations