FrontBackEnd

Using Enums in Thymeleaf

January 04, 2020 No comments Thymeleaf Enum

1. Introduction

In this article, we are going to show how to use **enums** in Thymeleaf templates. Starting with listing all available items in dropdown components, then showing how to use localization messages with enums, and finishing with using enums in comparison statements.

2. Setup

We prepared a simple enum that represents the planets of our solar system:

Copy

```
package com.frontbackend.thymeleaf.enums.model;

public enum Planet {

    MERCURY,
    VENUS,
    EARTH,
    MARS,
    JUPITER,
    SATURN,
    URANUS,
    NEPTUNE
}
```

To save submitted form we use `Home` class that will be our main `command object` :

Copy

```java
package com.frontbackend.thymeleaf.enums.model;

import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@ToString
@Setter
@Getter
@NoArgsConstructor
public class Home {

    private String title;
    private Planet planet;

}
```

As usual, we used `Lombok` library to get rid of setters, getters, constructors and toString() methods for the sake of readability.

2. Using enums in dropdowns

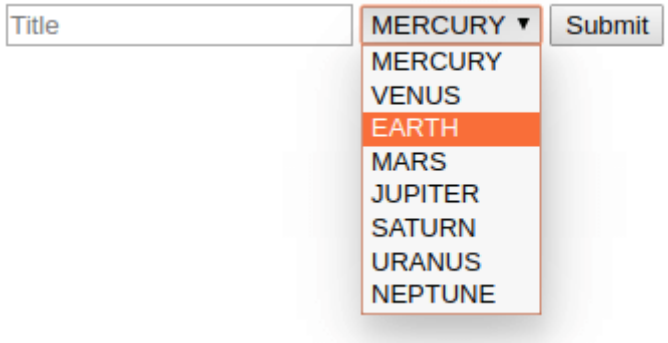
The Java enum objects contains a static method that returns all available items - it is a `values()` method. Let's use it to generate dropdown with all planets available to select:

Copy

```
<!DOCTYPE HTML>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8"/>
    <title>Spring Boot Thymeleaf Application - Enums</title>
</head>
<body>

<form th:action="@{/selectedPlanet}" method="post" th:object="${home}">
    <input th:field="*{title}" placeholder="Title"/>

    <select name="planet" th:field="*{planet}">
        <option th:each="planet :
${T(com.frontbackend.thymeleaf.enums.model.Planet).values()}"
                th:value="${planet}" th:text="${planet}"></option>
    </select>

    <input type="submit" value="Submit"/>
</form>

</body>
</html>
```

The special `T` operator is a part of Spring EL (Expression Language). We will use it to access a static method `values()` available an Enum `Planet`. Additionally, we put a simple INPUT field where a user can provide a title.

Rendered form for this template will look like the following:



As you can see we have all planets in the rendered dropdown with the same names that are provided in our Enum.

What if we want to have different names for select options? We simply need to use locale messages functionality described in the next chapter.

3. Using local messages for enum values

To change default locale configuration in our Spring Boot application we need to add a special `WebMvcConfigurer` class:

Copy

```
package com.frontbackend.thymeleaf.enums;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.LocaleResolver;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.springframework.web.servlet.i18n.LocaleChangeInterceptor;
import org.springframework.web.servlet.i18n.SessionLocaleResolver;

import java.util.Locale;

@Configuration
public class WebConfig implements WebMvcConfigurer {

    @Bean
    public LocaleResolver localeResolver(){
        SessionLocaleResolver localeResolver = new SessionLocaleResolver();
        localeResolver.setDefaultLocale(Locale.US);
        return  localeResolver;
    }

    @Bean
    public LocaleChangeInterceptor localeChangeInterceptor() {
        LocaleChangeInterceptor localeChangeInterceptor = new
LocaleChangeInterceptor();
        localeChangeInterceptor.setParamName("lang");
        return localeChangeInterceptor;
    }

    @Override
    public void addInterceptors(InterceptorRegistry registry){
        registry.addInterceptor(localeChangeInterceptor());
```

```
    }
}
```

From now we can change locale by providing additional request parameter `lang` to any internal URL:

- `?lang=EN` - for English,
- `?lang=ES` - for Spanish.

The main form looks a little bit different now. Options in dropdown menu have now message expressions `th:text="#{${'planet.' + planet}}"` that concatenate `planet.` text with enum value.

Copy

```
<!DOCTYPE HTML>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8"/>
    <title>Spring Boot Thymeleaf Application - Enums</title>
</head>
<body>

<form th:action="@{/selectedPlanet}" method="post" th:object="${home}">
    <input th:field="*{title}" placeholder="Title"/>

    <select name="planet" th:field="*{planet}">
        <option th:each="planet :
${T(com.frontbackend.thymeleaf.enums.model.Planet).values()}"
                th:value="${planet}" th:text="#{${'planet.' + planet}}">
</option>
    </select>

    <input type="submit" th:value="#{submit.button}"/>
</form>

<br/>

<a th:href="@{/(lang=en)}">EN</a> |
<a th:href="@{/(lang=es)}">ES</a>

</body>
</html>
```

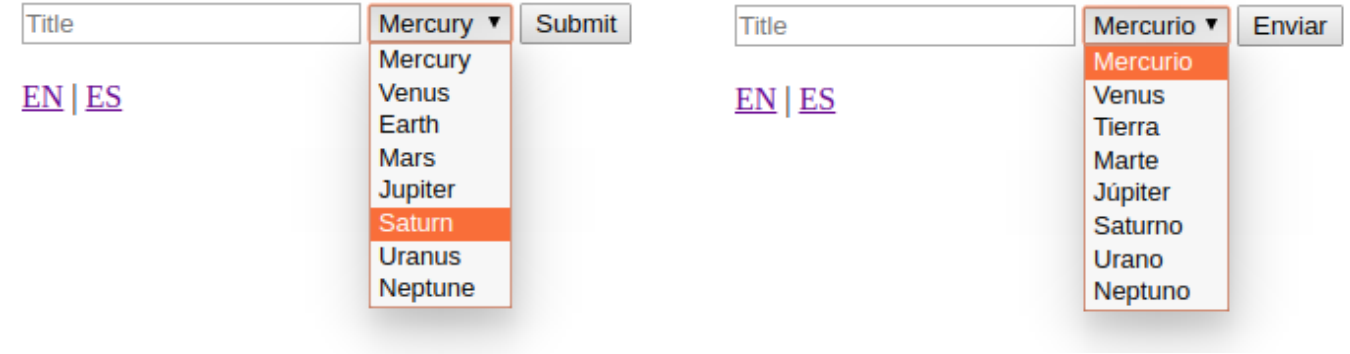
In the following table we've put together a message properites with a screens from rendered websites:

English (messages_en_US.properties)

```
planet.MERCURY=Mercury
planet.VENUS=Venus
planet.EARTH=Earth
planet.MARS=Mars
planet.JUPITER=Jupiter
planet.SATURN=Saturn
planet.URANUS=Uranus
planet.NEPTUNE=Neptune
submit.button=Submit

home.title=Title
home.planet=Planet
home.selected=Selected planet
```

Spanish (messages_es.properties)

```
planet.MERCURY=Mercurio
planet.VENUS=Venus
planet.EARTH=Tierra
planet.MARS=Marte
planet.JUPITER=Júpiter
planet.SATURN=Saturno
planet.URANUS=Urano
planet.NEPTUNE=Neptuno
submit.button=Enviar

home.title = Título
home.planet = Planet
home.selected = Planeta seleccionado
```



4. Using enum in comparision statements

We can easily use an enum to control what we want to display on the resulted website. We can use our `Planet` with Thymeleaf condition statements.

4.1. `If` statement

We can use Thymeleaf `if` statement with our enum to conditionally display additional text, like in the below example:

```
<th:block th:each="planet :
${T(com.frontbackend.thymeleaf.enums.model.Planet).values()}">
    <span th:if="${planet ==
T(com.frontbackend.thymeleaf.enums.model.Planet).EARTH}"
          th:text="${planet + ' = our home'}"></span>
</th:block>
```

Copy

We can also use string comparision:

```
<th:block th:each="planet :
${T(com.frontbackend.thymeleaf.enums.model.Planet).values()}">
    <span th:if="${planet.name() == 'EARTH'}" th:text="${planet + ' = our
home'}"></span>
</th:block>
```

Copy

4.2. `Switch` statement

Thymeleaf fully supports enums in `switch-case` statements.

Let's use our `Planet` enum to display the order of the planet from the sun:

Copy

```
<th:block th:each="planet :
${T(com.frontbackend.thymeleaf.enums.model.Planet).values()}">
    <div>
        <span th:text="${planet}"></span>
        <th:block th:switch="${planet}">
            <span
th:case="${T(com.frontbackend.thymeleaf.enums.model.Planet).MERCURY}"> - First
planet from the sun</span>
            <span
th:case="${T(com.frontbackend.thymeleaf.enums.model.Planet).VENUS}"> - Second
planet from the sun</span>
            <span
th:case="${T(com.frontbackend.thymeleaf.enums.model.Planet).EARTH}"> - Third
planet from the sun</span>
            <span
th:case="${T(com.frontbackend.thymeleaf.enums.model.Planet).MARS}"> - Fourth
planet from the sun</span>
            <span
th:case="${T(com.frontbackend.thymeleaf.enums.model.Planet).JUPITER}"> - Sixth
planet from the sun</span>
            <span
th:case="${T(com.frontbackend.thymeleaf.enums.model.Planet).SATURN}"> - Seventh
planet from the sun</span>
            <span
th:case="${T(com.frontbackend.thymeleaf.enums.model.Planet).URANUS}"> - Eighth
planet from the sun</span>
            <span
th:case="${T(com.frontbackend.thymeleaf.enums.model.Planet).NEPTUNE}"> - Ninth
planet from the sun</span>
        </th:block>
    </div>
</th:block>
```