Thymeleaf and Validation

Thymeleaf Helpers, Validators

Table of Content

Thymeleaf Helpers

Helpers

- Objects that provide built-in functionalities that helps you enhance your view
  - Dates
  - Strings
  - List
  - Numbers, etc.

Dates – Custom Format

WhiskeyController.java

```
@GetMapping("/home")
public String getHomePage(Model model) {
   model.addAttribute("myDate", new Date( ));
   return "whiskey-home";
}
```

whiskey-home.html

```
<div th:text="${#dates.format(myDate, 'yyyy-MMM-dd')}"></div>
```

```
2017-Mar-10
```

```
<div th:text="${#dates.dayOfWeekName(myDate)}"></div>
```

```
Friday
```

Dates – List Days

WhiskeyController.java

```
@GetMapping("/home")
public String getHomePage(Model model) {
   // List of dates -> 2016-1212, 2017-04-09 -> yyyy-MM-dd
   model.addAttribute("myDates", myDates);
   return "whiskey-home";
}
```

whiskey-home.html

```
<div th:text="${#dates.listDay(myDates)}"></div>
```

```
[12, 9]
```

Date – Get Current Date

WhiskeyController.java

@GetMapping("/home")
public String getHomePage( ) {
    return "whiskey-home";
}

whiskey-home.html
<div th:text="${#dates.createNow( )}"></div>
Fri mar 10 15:54:39 EET 2017

LocalDate and Thymeleaf

WhiskeyController.java

@GetMapping("/home")
public String getHomePage(Model model) {
    model.addAttribute("myDate", LocalDate.now( ));
    return "whiskey-home";
}

whiskey-home.html
<div th:text="${#temporals.format(myDate, 'dd-MMM-yyyy')}"></div>
10-Mar-2017

- To use LocalDate we need to add new dependency

```
<dependency>
        <groupId>org.thymeleaf.extras</groupId>
        <artifactId>thymeleaf-extras-java8time</artifactId>
        <version>3.0.4.RELEASE</version>
</dependency>
```

Strings – is Empty

WhiskeyController.java

@GetMapping("/home")
public String getHomePage(Model model) {
    String whiskeyNull = null;
    model.addAttribute("whiskey", whiskeyNull);
    return "whiskey-home";
}

whiskey-home.html
<div th:text="${#strings.isEmpty(whiskey)}"></div>
TRUE

Strings – Substring

WhiskeyController.java

```
@GetMapping("/home")
public String getHomePage(Model model) {
    String whiskeyNull = "Jack Daniels";
    model.addAttribute("whiskey", whiskeyNull);
    return "whiskey-home";
}
```

whiskey-home.html

```
<div th:text="${#strings.substring(whiskey,0,4)}"></div>
```

```
Jack
```

Strings – Join

WhiskeyController.java

```
@GetMapping("/home")
public String getHomePage(Model model) {
    model.addAttribute("whiskeys", whiskeys);
    // Jack Daniels, Jameson
    return "whiskey-home";
}
```

whiskey-home.html

```
<div th:text="${#strings.listJoin(whiskeys,-)}"></div>
```

```
Jack Daniels-Jameson
```

Strings – Capitalize

WhiskeyController.java

```
@GetMapping("/home")
public String getHomePage(Model model) {
    String whiskey = "jameson";
    model.addAttribute("whiskey", whiskey);
    return "whiskey-home";
}
```

whiskey-home.html

```
<div th:text="${#strings.capitalize(whiskey)}"></div>
```

```
Jameson
```

Numbers – Format

MathController.java

```
@GetMapping("/home")
public String getHomePage(Model model) {
    double num = 2.71828
    model.addAttribute("num", num);
    return "home";
}
```

home.html

```
<div th:text="${#numbers.formatDecimal(num,1,2)}"></div>
```

```
2.71
```

Numbers – Sequence

MathController.java

```
@GetMapping("/home")
public String getHomePage(Model model) {
    return "home";
}
```

home.html

```
<span th:each="number: ${#numbers.sequence(0,2)}">
    <span th:text="${number}"></span>
</span>
```

```
0 1 2
```

Aggregates – Sum

MathController.java

```
@GetMapping("/home")
public String getHomePage(Model model) {
    double[ ] whiskeyPrices = new double[ ]{29.23, 21.22, 33.50};
    model.addAttribute("whiskeyPrices", whiskeyPrices);
    return "whiskey-home";
}
```

whiskey-home.html

```
<div th:text="${#aggregates.sum(whiskeyPrices)}"></div>
```

```
83.95
```

Thymeleaf in JavaScript – Sum

MathController.java

```
@GetMapping("/js")
public String getMapPage(Model model) {
    String message = "Hi JS!";
    model.addAttribute("message", message);
    return "page";
}
```

script.js

```
<script th:inline="javascript">
    let message = [[${message}]];
</script>
```

```
83.95
```

How to Validate?

Spring Validation & Thymeleaf

- Making a simple Model validation and Error rendering

SomeModel.java

```
public class SomeModel {

    @NotNull
    @Size(min=3, max=10, message = "Invalid name")
    private String name;
}
```

SomeController.java

```
@Controller
public class SomeController {
    @GetMapping("/add")
    public String getPage(Model model) {
        if(!model.containsAttribute("bindingModel") {
            model.addAttribute("bindingModel", new BindingModel( ));
        }

        return "add";
    }
}
```

- Making a simple Model validation and Error rendering

SomeCongtroller.java

```
@PostMapping("/add")
public String add (
    @Valid @ModelAttribute("bindingModel") SomeModel bindingModel,
    BindingResult bindingResult
    RedirectAttributes rAtt
) {
    if(bindingResult.hasErrors( )) {
        rAtt.addFlashAttribute("bindingModel", bindingModel);
        rAtt.addFlashAttribute("org.springframework.validation.BindingResult.SomeModel", bindingResult);
        return "redirect:/add";
    }

    this.someService.save(bindingModel);

    return "redirect:/home";
}
```

add.html

```
<div th:object="${productBindingModel}" >
    <div class="justify-content-center">
        <label for="name" class="h4 mb-2 text-white"> Name </label>
    </div>
    <input th:field="*{name}" th:errorclass="bg-danger" type="text"
            class="form-control" id="name" name="name"/>
    <small th:if="${#fields.hasErrors('name')}"
            th:errors="*{name}" class="text-danger"> {Name error} </small>
</dev>
```

List All Errors

add.html

```
<ul th:if="${#fields.hasErrors('*')}">
    <li th:each="err : ${#fields.errors('*')}" th:text="${err}">
        {Input is incorrect}
    </li>
</ul>
```

add.html

```
<ul th:if="${#fields.hasErrors('${someModel.*}')}">
    <li th:each="err : ${#fields.errors('${someModel.*}')}" th:text="${err}">
        {Input is incorrect}
    </li>
</ul>
```

Custom Annotations (1)

- You can also implement custom validation annotations
    - Sometimes it is necessary due to complex validation functionality

PresentOrFuture.java

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.FIELD)
@Constraint(validateBy = PresentOrFutureValidator.class)
public @interface PresentOrFuture {

    String message( ) default "Invalid Date";

    Class<?>[ ] groups( ) default { };

    Class<? extends Payload>[ ] payload( ) default { };
}
```

Custom Annotations (2)

- You can also use the @PresentOrFuture validation annotations
- You will have to implement a custom validator too

SomeModel.java

```
public class SomeModel {

    @NotNull
    @PresentOrFuture
    @DateTimeFormat(pattern = "dd/MM/yyyy")
    private Date startDate;
}
```

Summary

- Objects that provide built-in functionalities that helps you enhance your view
- Thymeleaf provides helpers and validations
- Making a simple Model validation and Error rendering