

Thymeleaf and Validation

Thymeleaf Helpers, Validators

Table of Content

1. Thymeleaf Helpers
 - Dates
 - Strings
 - Numbers
 - Aggregates
2. Validations



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
<pre>@GetMapping("/home") public String getHomePage(Model model) { model.addAttribute("myDate", new Date()); return "whiskey-home"; }</pre>
whiskey-home.html
<pre><div th:text="\${#dates.format(myDate, 'yyyy-MMM-dd')}"></div></pre>
2017-Mar-10
<pre><div th:text="\${#dates.dayOfWeekName(myDate)}"></div></pre>
Friday

Dates – List Days

WhiskeyController.java
<pre>@GetMapping("/home") public String getHomePage(Model model) { // List of dates -> 2016-12-12, 2017-04-09 -> yyyy-MM-dd model.addAttribute("myDates", myDates); return "whiskey-home"; }</pre>
whiskey-home.html
<pre><div th:text="\${#dates.listDay(myDates)}"></div></pre>
[12, 9]

Date – Get Current Date

WhiskeyController.java
<pre>@GetMapping("/home") public String getHomePage() { return "whiskey-home"; }</pre>
whiskey-home.html
<pre><div th:text="\${#dates.createNow()}"></div></pre>
Fri mar 10 15:54:39 EET 2017

LocalDate and Thymeleaf

WhiskeyController.java
<pre>@GetMapping("/home") public String getHomePage(Model model) { model.addAttribute("myDate", LocalDate.now()); return "whiskey-home"; }</pre>
whiskey-home.html
<pre><div th:text="\${#temporals.format(myDate, 'dd-MMM-yyyy')}"></div></pre>
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
<pre>@GetMapping("/home") public String getHomePage(Model model) { String whiskeyNull = null; model.addAttribute("whiskey", whiskeyNull); return "whiskey-home"; }</pre>
whiskey-home.html
<pre><div th:text="\${#strings.isEmpty(whiskey)}"></div></pre>
TRUE

Strings – Substring

WhiskeyController.java
<pre>@GetMapping("/home") public String getHomePage(Model model) { String whiskeyNull = "Jack Daniels"; model.addAttribute("whiskey", whiskeyNull); return "whiskey-home"; }</pre>
whiskey-home.html
<pre><div th:text="\${#strings.substring(whiskey,0,4)}"></div></pre>
Jack

Strings – Join

WhiskeyController.java
<pre>@GetMapping("/home") public String getHomePage(Model model) { model.addAttribute("whiskeys", whiskeys); // Jack Daniels, Jameson return "whiskey-home"; }</pre>
whiskey-home.html
<pre><div th:text="\${#strings.listJoin(whiskeys,-)}"></div></pre>
Jack Daniels-Jameson

Strings – Capitalize

WhiskeyController.java
<pre>@GetMapping("/home") public String getHomePage(Model model) { String whiskey = "jameson"; model.addAttribute("whiskey", whiskey); return "whiskey-home"; }</pre>
whiskey-home.html
<pre><div th:text="\${#strings.capitalize(whiskey)}"></div></pre>
Jameson

Numbers – Format

MathController.java
<pre>@GetMapping("/home") public String getHomePage(Model model) { double num = 2.71828 model.addAttribute("num", num); return "home"; }</pre>
home.html
<pre><div th:text="\${#numbers.formatDecimal(num,1,2)}"></div></pre>
2.71

Numbers – Sequence

MathController.java
<pre>@GetMapping("/home") public String getHomePage(Model model) { return "home"; }</pre>
home.html
<pre> </pre>
0 1 2

Aggregates – Sum

MathController.java
<pre>@GetMapping("/home") public String getHomePage(Model model) { double[] whiskeyPrices = new double[] {29.23, 21.22, 33.50}; model.addAttribute("whiskeyPrices", whiskeyPrices); return "whiskey-home"; }</pre>
whiskey-home.html
<pre><div th:text="\${#aggregates.sum(whiskeyPrices)}"></div></pre>
83.95

Thymeleaf inJavaScript – Sum

MathController.java
<pre>@GetMapping("/js") public String getMapPage(Model model) { String message = "Hi JS!"; model.addAttribute("message", message); return "page"; }</pre>
script.js
<pre><script th:inline="javascript"> let message = [[\${message}]]; </script></pre>
83.95



How to Validate?

Spring Validation & Thymeleaf

- Making a simple Model validation and Error rendering

SomeModel.java	SomeController.java
<pre>public class SomeModel { @NotNull @Size(min=3, max=10, message = "Invalid name") private String name; }</pre>	<pre>@Controller public class SomeController { @GetMapping("/add") public String getPage(Model model) { if(!model.containsKey("bindingModel")) { model.addAttribute("bindingModel", new BindingModel()); } return "add"; } }</pre>

- Making a simple Model validation and Error rendering

SomeCongtroller.java
<pre>@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"; }</pre>

add.html
<pre><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> </div></pre>

List All Errors

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

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