Thymeleaf with Spring Boot

© luv2code LLC

What is Thymeleaf?

- Thymeleaf is a Java templating engine

www.thymeleaf.org

Separate project
Unrelated to spring.io

- Commonly used to generate the HTML views for web apps

- However, it is a general purpose templating engine
  - Can use Thymeleaf outside of web apps (more on this later)

© luv2code LLC

What is Thymeleaf?

www.thymeleaf.org

Separate project
Unrelated to spring.io

- Thymeleaf is a Java templating engine
- Commonly used to generate the HTML views for
- However, it is a general purpose templating engine

web ap

You can create Java apps with Thymeleaf

No need for Spring

But there is a lot of synergy between the two projects

How to pronounce Thymeleaf?

(tymeleaf)

like "time"

The "h" is silent.
Similar to Thomas, Thailand ...

What is a Thymeleaf template?

- Can be an HTML page with some Thymeleaf expressions
- Include dynamic content from Thymeleaf expressions

© luv2code LLC

Where is the Thymeleaf template processed?

- In a web app, Thymeleaf is processed on the server
- Results included in HTML returned to browser

© luv2code LLC

Thymeleaf Demo

Output

Time on the server is Sun Jan 06 17:00:40

© luv2code LLC

Development Process

Step-By-Step

1. Add Thymeleaf to Maven POM file

2. Develop Spring MVC Controller

3. Create Thymeleaf template

© luv2code LLC

Step 1: Add Thymeleaf to Maven pom file

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

start.spring.io

Dependencies

Spring Web WEB

Build web, including RESTful, applications using Spring MVC.
Uses Apache Tomcat as the default embedded container.

Thymeleaf TEMPLATE ENGINES

A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.

Based on this,
Spring Boot will auto configure to use Thymeleaf templates

© luv2code LLC

Step 2: Develop Spring MVC Controller

File: DemoController.java

```
@Controller
public class DemoController {

  @GetMapping("/")
  public String sayHello(Model theModel) {

    theModel.addAttribute("theDate", new java.util.Date());

    return "helloworld";
  }
}
```

```
src/main/resources/templates/helloworld.html
```


© luv2code LLC

Where to place Thymeleaf template?

- In Spring Boot, your Thymeleaf template files go in
  - **src/main/resources/templates**

- For web apps, Thymeleaf templates have a **.html** extension

© luv2code LLC

Step 3: Create Thymeleaf template

Thymeleaf accesses "theDate" from the Spring MVC Model

File: DemoController.java

```
@Controller
public class DemoController {

   @GetMapping("/")
   public String sayHello(Model theModel) {

      Attribute("theDate", new java.util.Date());

      world";

   }
```

1

To use Thymeleaf expressions

File: src/main/resources/templates/helloworld.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head> … </head>

<body>
    <p th:text="'Time on the server is ' + ${theDate}" />
</body>

</html>
```

Thymeleaf expression

2

Time on the server is Sun Jan 06 17:00:40

© luv2code LLC

Step 3: Create Thymeleaf template

Thymeleaf accesses "theDate" from the Spring MVC Model

File: DemoController.java

```
@Controller
public class DemoController {

   @GetMapping("/")
   public String sayHello(Model theModel) {

      theModel.addAttribute("theDate", new java.util.Date());

      return "helloworld";
   }
}
```

1

File: src/main/resources/templates/helloworld.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head> … </head>

<body>
    <p th:text="'Time on the server is ' + ${theDate}" />
</body>

</html>
```

2

Thymeleaf expression

Time on the server is Sun Jan 06 17:00:40

© luv2code LLC

Additional Features

- Looping and conditionals
- CSS and JavaScript integration
- Template layouts and fragments

www.thymeleaf.org

© luv2code LLC

CSS and Thymeleaf

© luv2code LLC

Let's Apply CSS Styles to our Page

Before

After

```
font-style: italic;
color: green;
```


© luv2code LLC

Using CSS with Thymleaf Templates

- You have the option of using
  - Local CSS files as part of your project
  - Referencing remote CSS files

- We'll cover both options in this video

© luv2code LLC

Development Process

Step-By-Step

1. Create CSS file

2. Reference CSS in Thymeleaf template

3. Apply CSS style

© luv2code LLC

Step 1: Create CSS file

- Spring Boot will look for static resources in the directory
  - **`src/main/resources/static`**

You can create your own
custom sub-directories
`static/css`
`static/images`
`static/js`
etc …

Can be any sub-directory name

File: demo.css

```
.funny {
  font-style: italic;
  color: green;
}
```


© luv2code LLC

Step 2: Reference CSS in Thymeleaf template

File: helloworld.html

```
<head>
  <title>Thymeleaf Demo</title>

  <!-- reference CSS file -->
  <link rel="stylesheet" th:href="@{/css/demo.css}" />

</head>
```

@ symbol

Reference context path of your application (app root)

© luv2code LLC

Step 3: Apply CSS

File: helloworld.html

```
<head>
  <title>Thymeleaf Demo</title>

  <!-- reference CSS file -->
  <link rel="stylesheet" th:href="@{/css/demo.css}" />
</head>

<body>
  <p th:text="'Time on the server is ' + ${theDate}" class="funny" />
</body>
```

File: demo.css

```
.funny {
  font-style: italic;
  color: green;
}
```

© luv2code LLC

Other search directories

Spring Boot will search following directories for static resources:

`/src/main/resources`

1. `/META-INF/resources`
2. `/resources`
3. `/static`
4. `/public`

Search order: top-down

© luv2code LLC

3rd Party CSS Libraries - Bootstrap

- Local Installation
- Download Bootstrap file(s) and add to **`/static/css`** directory

```
<head>
  … …

  <!-- reference CSS file -->
  <link rel="stylesheet" th:href="@{/css/bootstrap.min.css}" />

</head>
```

© luv2code LLC

3rd Party CSS Libraries - Bootstrap

- Remote Files

```
<head>
  … …

  <!-- reference CSS file -->
  <link rel="stylesheet"
       href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" />

  … …
</head>
```


© luv2code LLC

Spring MVC Form Validation

© luv2code LLC

The Need for Validation

Check the user input form for

required fields

valid numbers in a range

valid format (postal code)

custom business rule

© luv2code LLC

Java's Standard Bean Validation API

- Java has a standard Bean Validation API
- Defines a metadata model and API for entity validation
- Spring Boot and Thymeleaf also support the Bean Validation API

http://www.beanvalidation.org

© luv2code LLC

Bean Validation Features

Validation Feature
required
validate length
validate numbers
validate with regular expressions
custom validation

© luv2code LLC

Validation Annotations

Annotation	Description
@NotNull	Checks that the annotated value is not null
@Min	Must be a number >= value
@Max	Must be a number <= value
@Size	Size must match the given size
@Pattern	Must match a regular expression pattern
@Future / @Past	Date must be in future or past of given date
others …	

© luv2code LLC

Our Road Map

1. set up our development environment
2. required field
3. validate number range: min, max
4. validate using regular expression (regexp)
5. custom validation

© luv2code LLC

Spring MVC Form Validation
Required Fields

© luv2code LLC

Required Fields

Fill out the form. Asterisk (*) means required.

First name:

Last name (*):

Submit

Fill out the form. Asterisk (*) means required.

First name:

Last name (*): is required

Submit

© luv2code LLC

Pulling It All Together

customer-form.html

Customer

Customer Controller

Customer

customer-confirmation.html

The customer is confirmed: John Doe

© luv2code LLC

Development Process

Step-By-Step

1. Create Customer class and add validation rules
2. Add Controller code to show HTML form
3. Develop HTML form and add validation support
4. Perform validation in the Controller class
5. Create confirmation page

© luv2code LLC

Step 1: Create Customer class and add validation rules

File: Customer.java

```
import jakarta.validation.constraints.NotNull;
import jakarta.validation.constraints.Size;

public class Customer {

    private String firstName;

    @NotNull(message = "is required")
    @Size(min=1, message = "is required")
    private String lastName = “”;

    // getter/setter methods …

}
```

Validation rules

© luv2code LLC

Step 2: Add Controller code to show HTML form

File: CustomerController.java

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.ui.Model;

@Controller
public class CustomerController {

    @GetMapping("/")
    public String showForm(Model theModel) {

        theModel.addAttribute("customer", new Customer());

        return "customer-form";
    }

   …

}
```

Model allows us to share information between Controllers and view pages (Thymeleaf)

name

value

© luv2code LLC

Step 3: Develop HTML form and add validation support

Where to submit form data

Model attribute name

File: customer-form.html

```
<form th:action="@{/processForm}" th:object="${customer}" method="POST">

    First name: <input type="text" th:field="*{firstName}" />

    <br><br>

    Last name (*): <input type="text" th:field="*{lastName}" />

    <!-- Show error message (if present) -->
    <span th:if="${#fields.hasErrors('lastName')}"
          th:errors="*{lastName}"
          class="error"></span>

    <br><br>

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

</form>
```

Property name from Customer class

Property name from Customer class

First name:

Last name (*): is required

Submit

© luv2code LLC

Step 4: Perform validation in Controller class

File: CustomerController.java

```
...

@PostMapping("/processForm")
public String processForm(
        @Valid @ModelAttribute("customer") Customer theCustomer,
        BindingResult theBindingResult) {

    if (theBindingResult.hasErrors()) {
        return "customer-form";
    }
    else {
        return "customer-confirmation";
    }
}
```

Model attribute name

Tell Spring MVC to perform validation

The results of validation

© luv2code LLC

Step 5: Create confirmation page

File: customer-confirmation.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">

<body>

The customer is confirmed: <span th:text="${customer.firstName + ' ' + customer.lastName}" />

</body>

</html>
```


© luv2code LLC

Spring MVC Validation
@InitBinder

© luv2code LLC

White Space

- Our previous example had a problem with white space
  - **Last name** field with all whitespace **passed** … YIKES!
  - Should have **failed**!
- We need to trim whitespace from input fields

© luv2code LLC

@InitBinder

Advanced

- **@InitBinder** annotation works as a pre-processor
- It will pre-process each web request to our controller
- Method annotated with **@InitBinder** is executed

© luv2code LLC

@InitBinder

- We will use this to trim Strings
  - Remove leading and trailing white space
- If String only has white spaces … trim it to **null**
- Will resolve our validation problem … whew :-)

© luv2code LLC

Register Custom Editor in Controller

CustomerController.java

```
...

@InitBinder
public void initBinder(WebDataBinder dataBinder) {

    StringTrimmerEditor stringTrimmerEditor = new StringTrimmerEditor(true);

    dataBinder.registerCustomEditor(String.class, stringTrimmerEditor);
}

...
```

Defined in Spring API

true - means trim to null

Pre-process every String form data

Remove leading and trailing white space

If String only has white space ... trim it to null

StringTrimmerEditor: removes whitespace - leading and trailing

© luv2code LLC

Spring MVC Validation
Number Range: @Min and @Max

© luv2code LLC

Validate a Number Range

- Add a new input field on our form for: **Free Passes**

- User can only enter a range: 0 to 10

© luv2code LLC

Development Process

Step-By-Step

1. Add validation rule to Customer class

2. Display error messages on HTML form

3. Perform validation in the Controller class

4. Update confirmation page

© luv2code LLC

Step 1: Add validation rule to Customer class

```
import jakarta.validation.constraints.Min;
import jakarta.validation.constraints.Max;

public class Customer {

   @Min(value=0, message="must be greater than or equal to zero")
   @Max(value=10, message="must be less than or equal to 10")
   private int freePasses;

   // getter/setter methods

}
```

Error message

© luv2code LLC

Development Process

Step-By-Step

1. Add validation rule to Customer class

2. Display error messages on HTML form

3. Perform validation in the Controller class

4. Update confirmation page

Cover these steps in the coding video

© luv2code LLC

Spring MVC Validation
Regular Expressions

© luv2code LLC

Regular Expressions

Advanced

- A sequence of characters that define a search pattern
  - This pattern is used to find or match strings

- Regular Expressions is like its own language (advanced topic)
  - I will assume you already know about regular expressions

- If not, then plenty of free tutorials available
  - https://docs.oracle.com/javase/tutorial/essential/regex/

© luv2code LLC

Validate a Postal Code

- Add a new input field on our form for: **Postal Code**
- User can only enter 5 chars / digits
- Apply ***Regular Expression***

© luv2code LLC

Development Process

Step-By-Step

1. Add validation rule to Customer class

2. Display error messages on HTML form

3. Update confirmation page

© luv2code LLC

Step 1: Add validation rule to Customer class

Advanced

```
import jakarta.validation.constraints.Pattern;

public class Customer {

   @Pattern(regexp="^[a-zA-Z0-9]{5}", message="only 5 chars/digits")
   private String postalCode;

   // getter/setter methods

}
```

The "regular expression" pattern

© luv2code LLC

Spring MVC Validation
Make an Integer Field Required

© luv2code LLC

Spring MVC Validation
Handle String Input for Integer Fields

© luv2code LLC

Development Process

Step-By-Step

1. Create custom error message

- src/resources/messages.properties

© luv2code LLC

File location is very important: src/main/resources/messages.properties

Spring MVC Validation
Custom Validation

© luv2code LLC

Custom Validation Demo

First name:

Last name:

Course Code:

Submit

© luv2code LLC

First name: John

Last name: Doe

Course Code: ABC1234 Course code must start with LUV

Submit

The student is confirmed: John Doe

Course Code: LUV123

Custom Validation

- Perform custom validation based on your business rules
  - Our example: Course Code must start with “LUV”
- Spring MVC calls our custom validation
- Custom validation returns boolean value for pass/fail (true/false)

© luv2code LLC

Create a custom Java Annotation … from scratch

Advanced

- So far, we've used predefined validation rules: @Min, @Max, …
- For custom validation … we will create a **Custom Java Annotation**
  - **@CourseCode**

The value the course code must start with

Errorr message if validation fails

```
@CourseCode(value="LUV", message="must start with LUV")
private String courseCode;
```

Field in our entity class: Student.java

© luv2code LLC

Development Process

Step-By-Step

1. Create custom validation rule
2. Add validation rule to Customer class
3. Display error messages on HTML form
4. Update confirmation page

© luv2code LLC

Development Process - Drill Down

Step-By-Step

1. Create custom validation rule
   a. Create **@CourseCode** annotation
   b. Create **CourseCodeConstraintValidator**

Helper class

Contains our custom business logic for validation

© luv2code LLC

Step 1a: Create @CourseCode annotation

Advanced

Usage Example

```
@CourseCode(value="LUV", message="must start with LUV")
private String courseCode;
```


© luv2code LLC

Step 1a: Create @CourseCode annotation

Advanced

```
@Constraint(validatedBy = CourseCodeConstraintValidator.class)
@Target( { ElementType.METHOD, ElementType.FIELD } )
@Retention(RetentionPolicy.RUNTIME)
public @interface CourseCode {

    ...

}
```

Helper class that contains business rules / validation logic

Can apply our annotation to a method or field

Retain this annotation in the Java class file.

Process it at runtime

Note the "@interface" syntax

```
@CourseCode(value="LUV", message="must start with LUV")
private String courseCode;
```


© luv2code LLC

Step 1a: Create @CourseCode annotation

Advanced

```
@Constraint(validatedBy = CourseCodeConstraintValidator.class)
@Target( { ElementType.METHOD, ElementType.FIELD } )
@Retention(RetentionPolicy.RUNTIME)
public @interface CourseCode {

   // define default course code
   public String value() default "LUV";



   ...
}
```

Default value

```
@CourseCode(value="LUV", message="must start with LUV")
private String courseCode;
```


© luv2code LLC

Step 1a: Create @CourseCode annotation

Advanced

```
@Constraint(validatedBy = CourseCodeConstraintValidator.class)
@Target( { ElementType.METHOD, ElementType.FIELD } )
@Retention(RetentionPolicy.RUNTIME)
public @interface CourseCode {

   // define default course code
   public String value() default "LUV";

   // define default error message
   public String message() default "must start with LUV";

   ...
}
```

Default value

```
@CourseCode(value="LUV", message="must start with LUV")
private String courseCode;
```


© luv2code LLC

Step 1b: Create CourseCodeConstraintValidator

Advanced

```
import jakarta.validation.ConstraintValidator;
import jakarta.validation.ConstraintValidatorContext;

public class CourseCodeConstraintValidator
    implements ConstraintValidator<CourseCode, String> {

    private String coursePrefix;

    @Override
    public void initialize(CourseCode theCourseCode) {
        coursePrefix = theCourseCode.value();
    }

    @Override
    public boolean isValid(String theCode,
                           ConstraintValidatorContext theConstraintValidatorContext) {

        boolean result;

        if (theCode != null) {
            result = theCode.startsWith(coursePrefix);
        }
        else {
            result = true;
        }

        return result;
    }
}
```

Our annotation

Type of data to validate against

HTML Form Data Entered by the user

Helper class for additional error messages

Validation logic

Test if the form data starts with our course prefix

does it start with "LUV"?

If nothing entered (null) ... just return true

The course code is not a required field

© luv2code LLC