Spring MVC with Thymeleaf

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Have a Question?

Software University

sli.do

#java-web

Table of Content

1. **Thymeleaf**
2. **Binding Requests** and **Response Model**
3. **Validation** and **Thymeleaf**

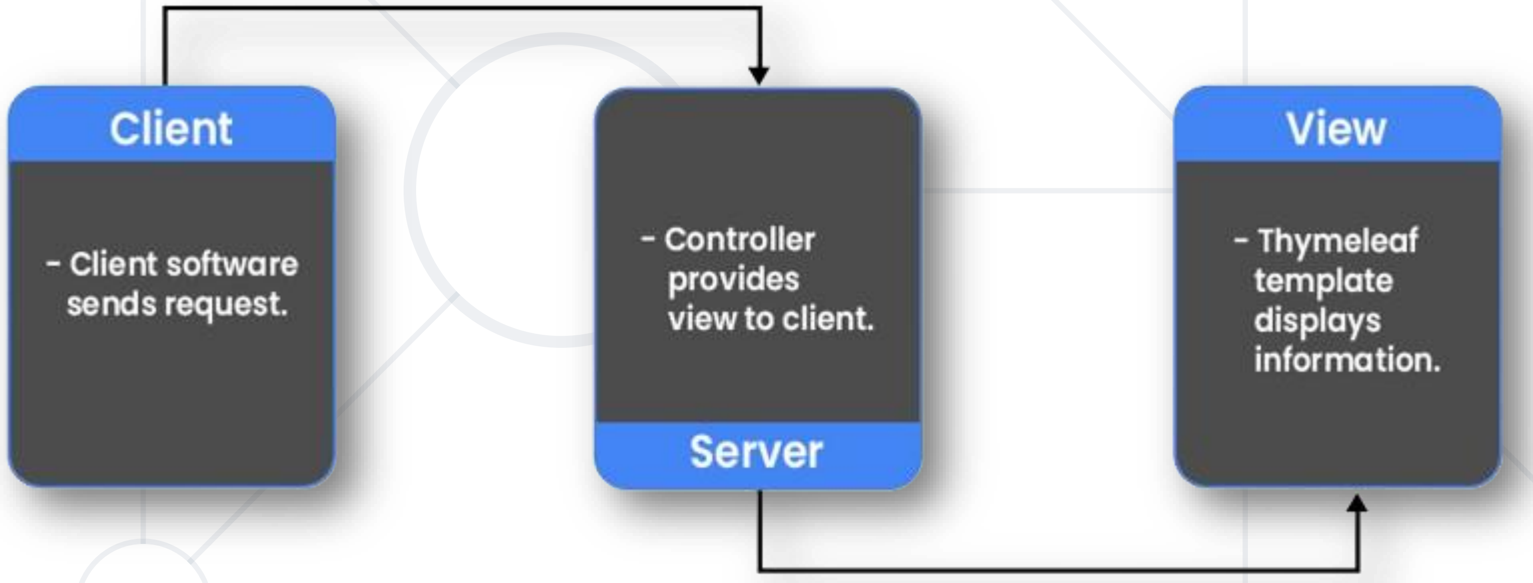
Thymeleaf

What is Thymeleaf?

- **Thymeleaf** is a modern **server-side** Java **template engine** for both web and standalone environments
- It enables natural **templates** with a well-structured format that is both **human-readable** and **editable**
- Types of templates it can **process** are – **HTML**, **JAVASCRIPT**, **CSS**, **XML**, **TEXT** and **RAW**

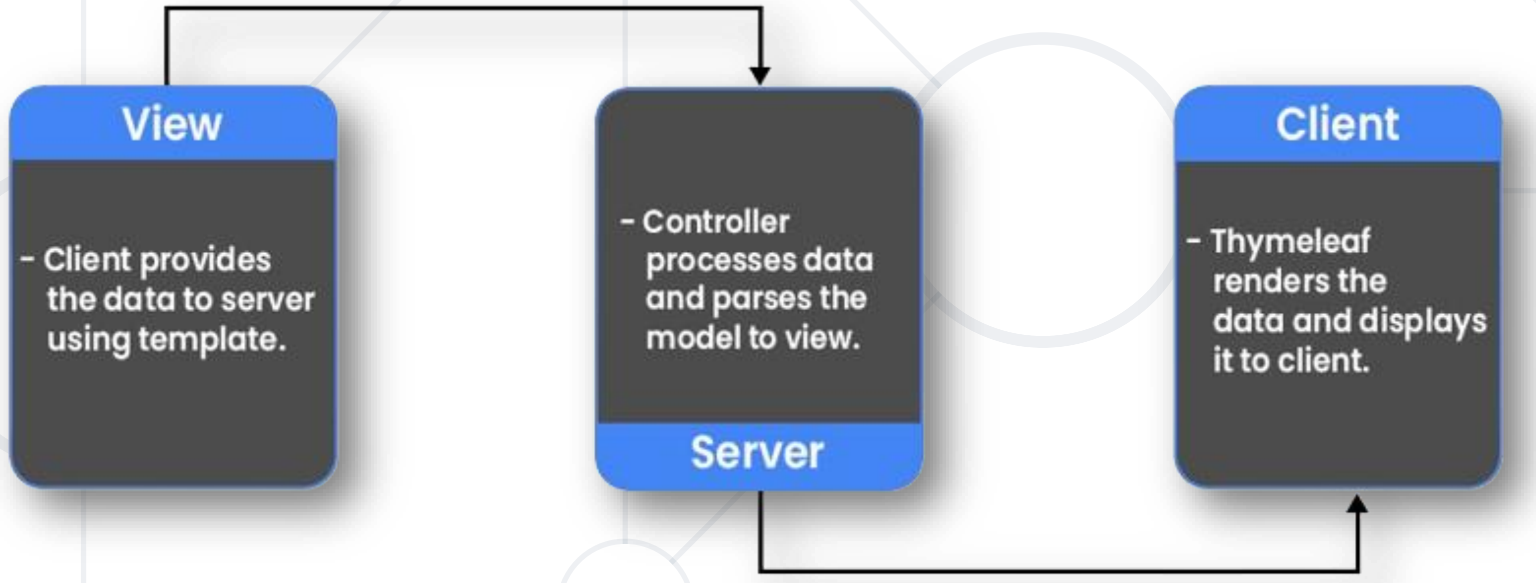
How Thymeleaf Works?

- Thymeleaf follows a **De-Coupled** Architecture – It is **unaware** of any web framework
- In the same way, it is unaware of Spring's **abstraction** of the model and thus cannot handle the data that the controller places in the Model


Client
- Client software sends request.
- Controller provides view to client.
Server
View
- Thymeleaf template displays information.

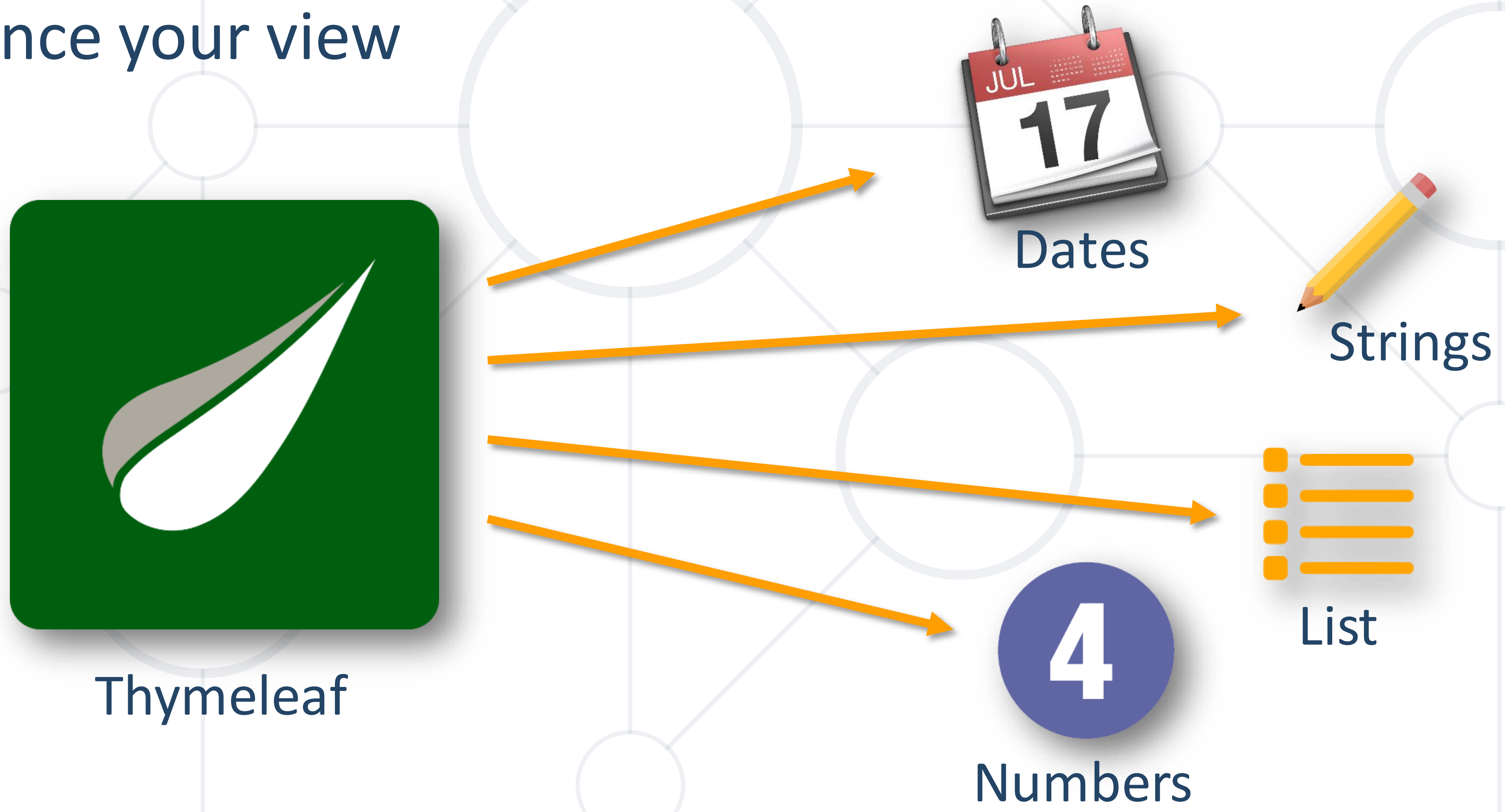
How Thymeleaf Works?

- When Spring-Boot's autoconfiguration detects **Thymeleaf** in the classpath, it creates **beans** supporting Thymeleaf view for Spring MVC
- It can work with request **attributes** of **Servlet**


View
- Client provides the data to server using template.
- Controller processes data and parses the model to view.
Server
Client
- Thymeleaf renders the data and displays it to client.

Helpers

- Objects that provide built-in functionalities that helps you enhance your view



Date – Custom Format

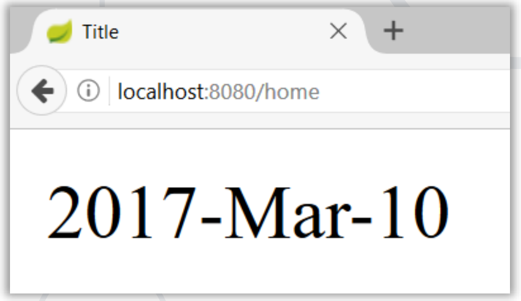
WhiskeyController.java

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

Format Date

whiskey-home.html

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



2017-Mar-10

Date – Week Name of Day

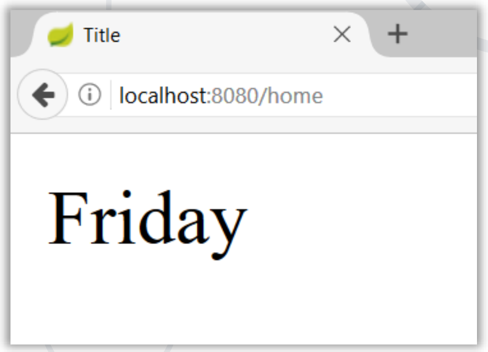
WhiskeyController.java

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

Day Name

whiskey-home.html

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



Friday

Date – List Days

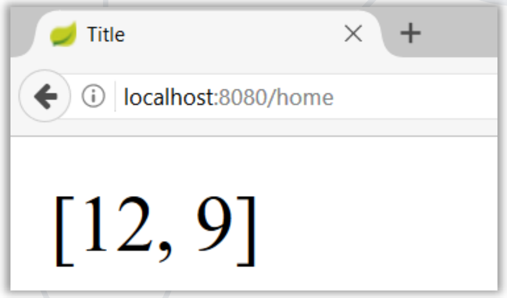
WhiskeyController.java

```
@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";
  }
```

List Days

whiskey-home.html

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



```
[12, 9]
```

Date – Get Current Date

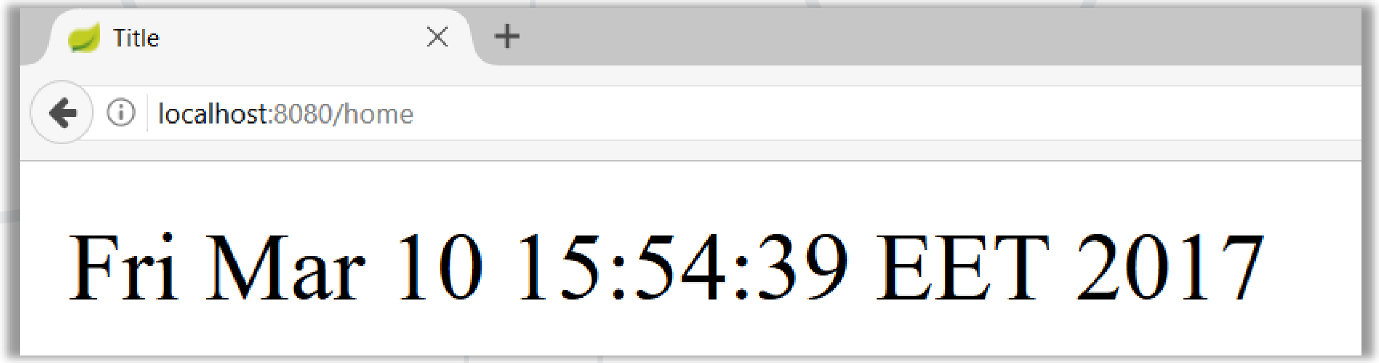
WhiskeyController.java

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

Today's Date

whiskey-home.html

```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";
}
```

Format Date

whiskey-home.html

```
${#temporals.format(myDate, 'dd-MMM-yyyy')}
```

Strings – is Empty

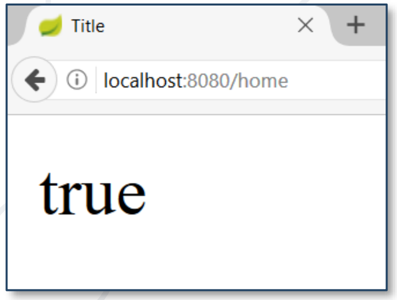
WhiskeyController.java

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

Null / Empty Check

whiskey-home.html

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

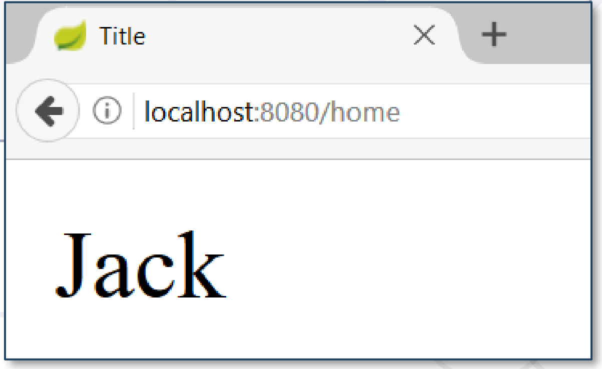


true

Strings – Substring

WhiskeyController.java

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



Jack

Substring

whiskey-home.html

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

Strings – Join

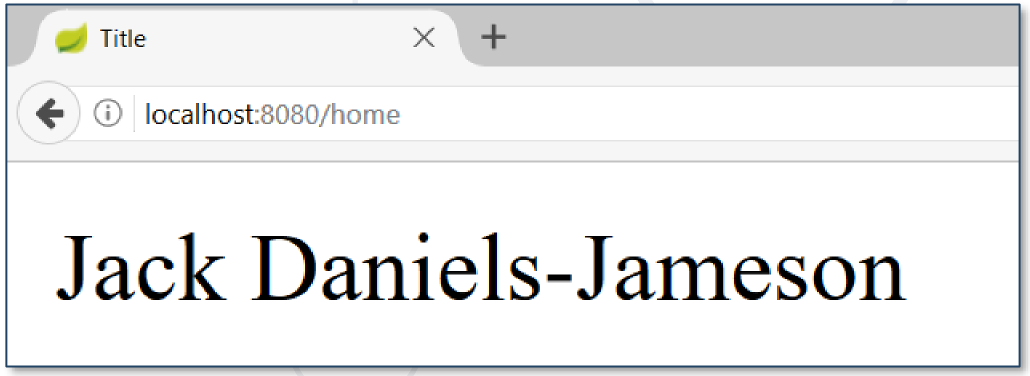
WhiskeyController.java

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

Join

whiskey-home.html

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



Jack Daniels-Jameson

Strings – Capitalize

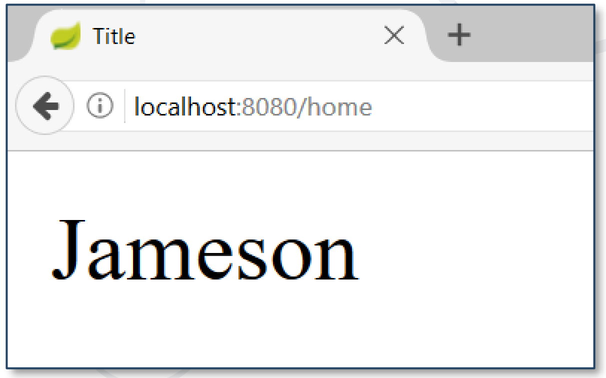
WhiskeyController.java

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

Capitalize

whiskey-home.html

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



Jameson

Numbers – Format

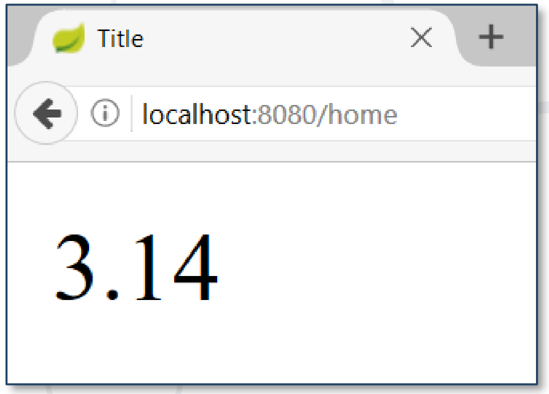
MathController.java

```
@GetMapping("/home")
   public String getHomePage(Model model) {
        double num = 3.14159;
        model.addAttribute("num", num);
        return "home";
    }
```

Format

home.html

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



3.14

Numbers – Sequence

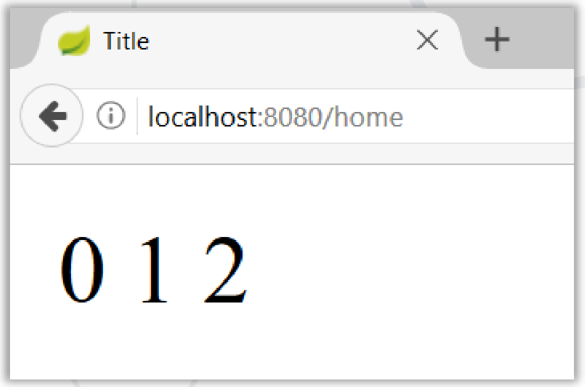
MathController.java

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

Sequence

home.html

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



```
0 1 2
```

Aggregates – Sum

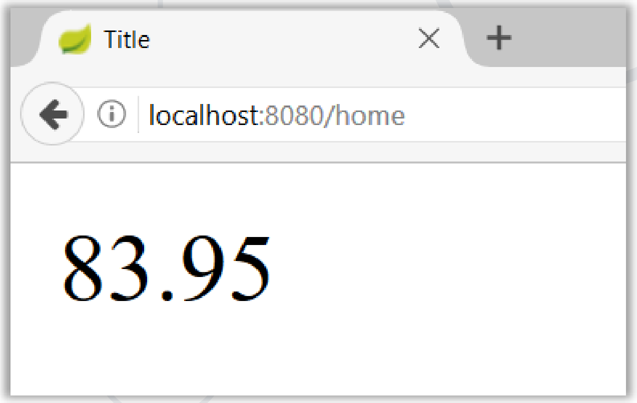
WhiskeyController.java

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

Sum

whiskey-home.html

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



83.95

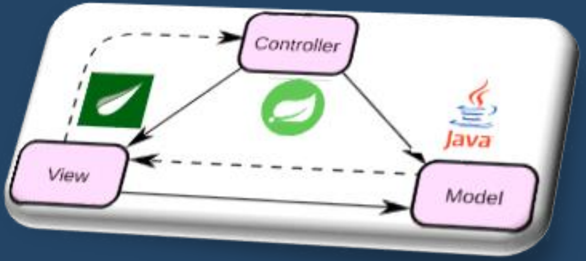
Thymeleaf in JavaScript

JSController.java

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

script.js

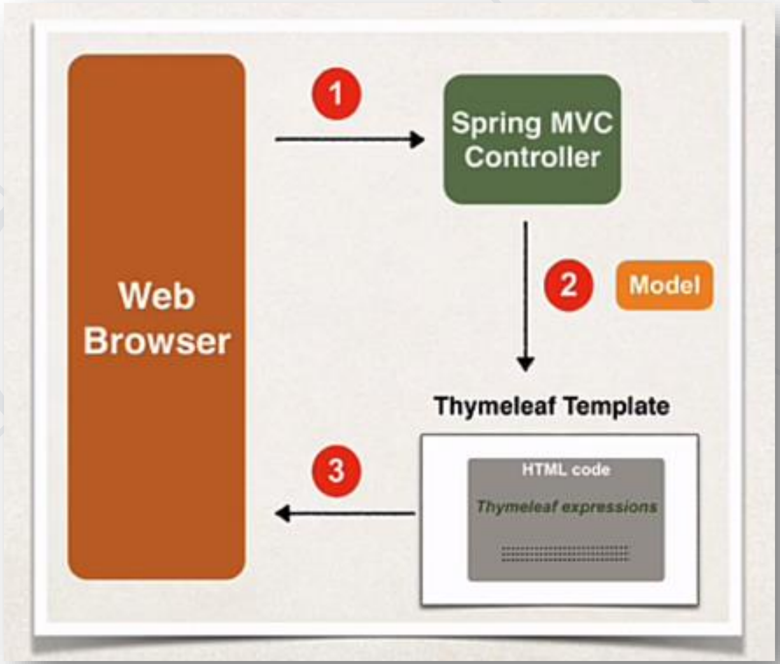
```html
<script th:inline="javascript">
   let message = [[${message}]];
<script>
```



Binding Requests and Response Model

Binding Requests

- In **Thymeleaf**, binding requests refers to the process of accessing and processing data sent from a client-side form **submission** or HTTP **request**
- Thymeleaf provides various mechanisms for binding requests, allowing developers to **seamlessly** integrate form data into their Spring MVC applications



Binding Requests

- **Form Submission:** Thymeleaf enables developers to create HTML **forms** in templates for users to input data. When a user submits a form, the form data is sent to the **server** as part of an HTTP request
- **Form Fields:** Thymeleaf allows developers to **bind** form fields to corresponding model attributes or command objects using the **th:field** attribute. This attribute specifies the field name and instructs Thymeleaf to bind the form field's value to the specified attribute

Binding Requests

- **Binding** form field to a model **attribute**

someForm.html

```
<form th:action="@{/submitForm}" th:object="${user}" method="post">
    <input type="text" th:field="*{username}" />
    <input type="password" th:field="*{password}" />
    <button type="submit">Submit</button>
</form>
```

Binding

Binding Requests

- **Command Object** is the name Spring MVC gives to **form-backing** beans, this is, to **objects** that model a form's fields and provide **getter** and **setter** methods that will be used by the framework for establishing and obtaining the values input by the user at the browser side

User.java

```java
public class User {
    private String username;
    private String password;

    // Getters and setters
}
```

UserController.java

```java
@Controller
public class UserController {
    @PostMapping("/submitForm")
    public String
handleSubmitForm(@ModelAttribute("user") User user) {
        // Process form data
        return "successPage";
    }
}
```

Binding Requests

- **Request Parameters:** Thymeleaf also allows developers to **access** request parameters directly in templates using the **${param}** syntax

```
some.html
<p th:text="${param.message}"></p>
```

Retrieving the value of the request parameter

Response Model

- **In Thymeleaf,** the response model refers to the data passed from a Spring MVC **controller** to the Thymeleaf **template** for rendering
- It typically consists of **attributes** or **objects** that contain the data needed to dynamically **generate** the HTML content displayed to the user
- The response model allows you to pass data from Spring MVC **controllers** to Thymeleaf templates, enabling dynamic content generation based on the data provided by the **server-side logic**

Response Model

- **Passing Data to Thymeleaf Template:** in a Spring MVC controller method, you can add **attributes** to the **model** object, which will be passed to the Thymeleaf template for rendering

MyController.java

```java
@Controller
public class MyController {

    @GetMapping("/hello")
    public String hello(Model model) {
        model.addAttribute("message", "Hello, World!");
        return "hello"; // Thymeleaf template name
    }
}
```

Response Model

- **Accessing Model Attributes in Thymeleaf Template:** In the Thymeleaf template, you can access the model attributes using Thymeleaf expressions

some.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Hello Page</title>
</head>
<body>
    <h1 th:text="${message}"></h1>
</body>
</html>
```

Thymeleaf expression

Response Model

- **Object Binding:** You can also pass Java objects to the Thymeleaf template and access their properties in the template

MyController.java

```java
@Controller
public class MyController {

    @GetMapping("/user")
    public String user(Model model) {
        User user = new User("John", "Doe");
        model.addAttribute("user", user);
        return "user"; // Thymeleaf template
name
    }
}
```

some.html

```html
<!DOCTYPE html>
<html lang="en"
xmlns:th="http://www.thymeleaf.org"
>
<head>
    <meta charset="UTF-8">
    <title>User Page</title>
</head>
<body>
    <p th:text="${user.firstName +
' ' + user.lastName}"></p>
</body>
</html>
```

Validation and Thymeleaf

Spring Validation & Thymeleaf

- Making a simple **Model validation** and **Error rendering**

SomeModel.java

```java
public class SomeModel {

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

SomeController.java

```java
@Controller
public class SomeController {

 @GetMapping("/add")
 public String getPage(Model model) {

   if(!model.containsAttribute("bindingModel")){
       model.addAttribute("bindingModel",
                          new BindingModel());
       }

        return "add";
   }
}
```

Adding a model to the view

Spring Validation & Thymeleaf

- Making a simple **Model validation** and **Error rendering**

SomeController.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";
    }
```

Validate the model

Validation Result

Spring Validation & Thymeleaf

- Making a simple **Model validation** and **Error rendering**

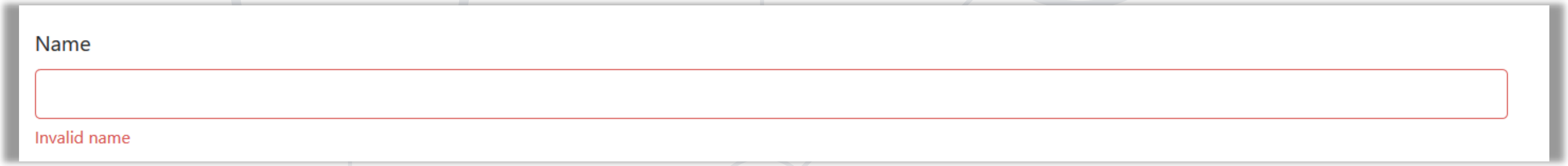
add.html

```
<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>
</div>
```

Set Binding Model

Access Field

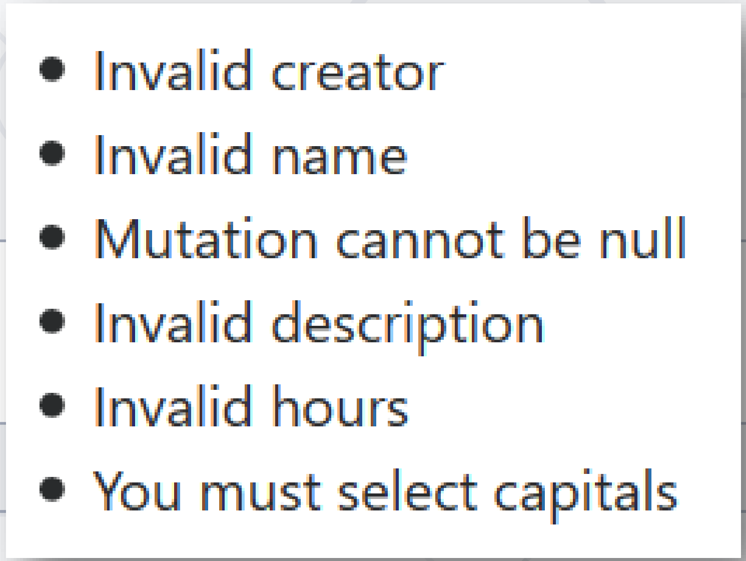
Render Error



List All Errors

add.html

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



- Invalid creator
- Invalid name
- Mutation cannot be null
- Invalid description
- Invalid hours
- You must select capitals

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

- 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(validatedBy = PresentOrFutureValidator.class)
public @interface PresentOrFuture {

    String message() default "Invalid Date";

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

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

Custom Annotations

- 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;
}
```

Annotation

Summary

- Thymeleaf
- Binding Requests and Response Model
- Validation and Thymeleaf

Software University

Questions?

SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

СофтУни БУДИТЕЛ

Finance Academy

SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank

Решения за твоето утре

DRAFT KINGS

VIVACOM

AMBITIONED

INDEAVR

Serving the high achievers

SOFTWARE GROUP

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License