Bài 3
View vàThymeleaf

Module: BOOTCAMP WEB-BACKEND DEVELOPMENT

Kiểm tra bài trước

Hỏi và trao đổi về các khó khăn gặp phải trong bài "Spring Controller"

Tóm tắt lại các phần đã học từ bài "Spring Controller"

Mục tiêu

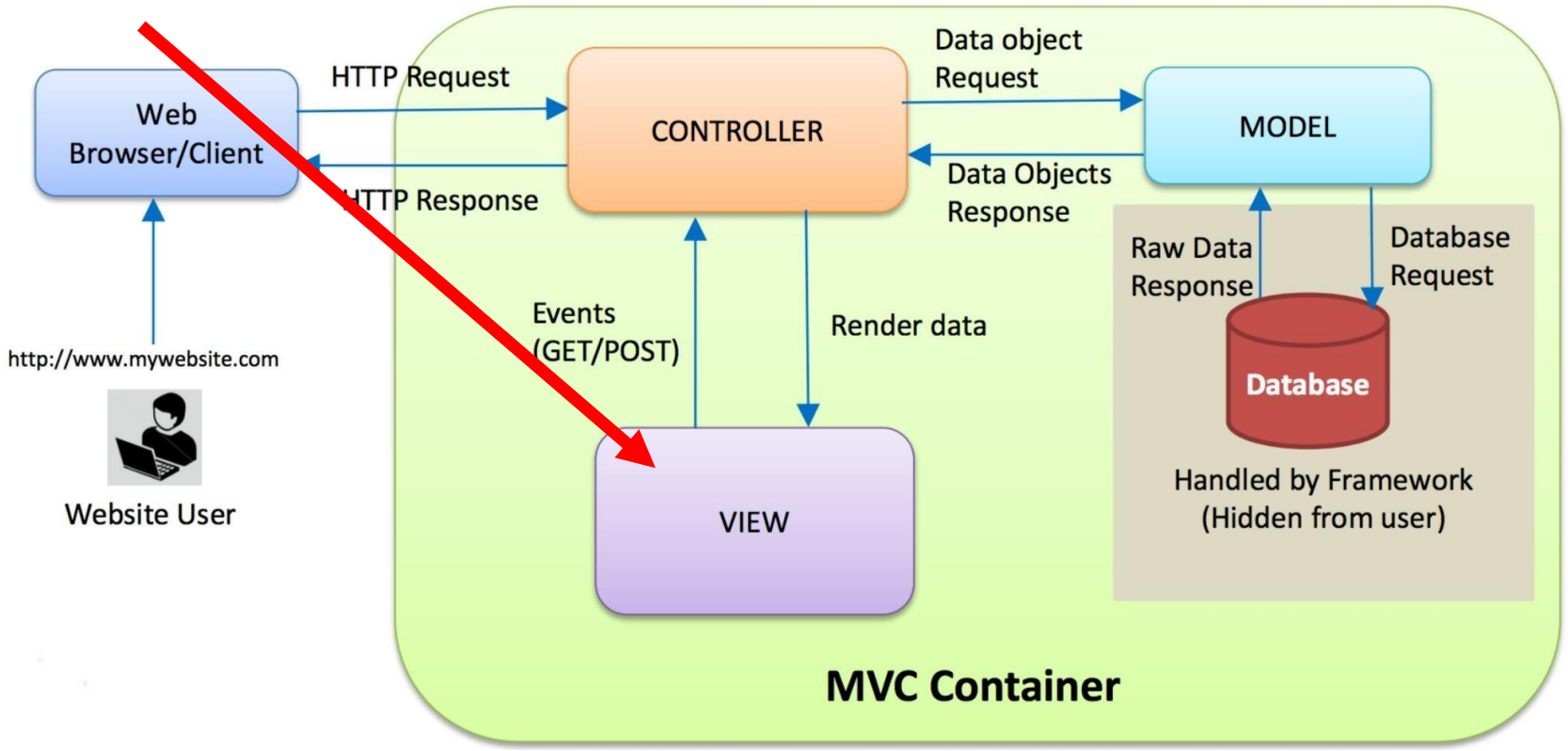
- Cấu hình được Thymeleaf trong ứng dụng Spring MVC
- Sử dụng được các biểu thức của Thymeleaf
- Sử dụng được vòng lặp trong Thymeleaf
- Sử dụng được câu lệnh điều kiện trong Thymeleaf
- Sử dụng được template layout

Thảo luận

View

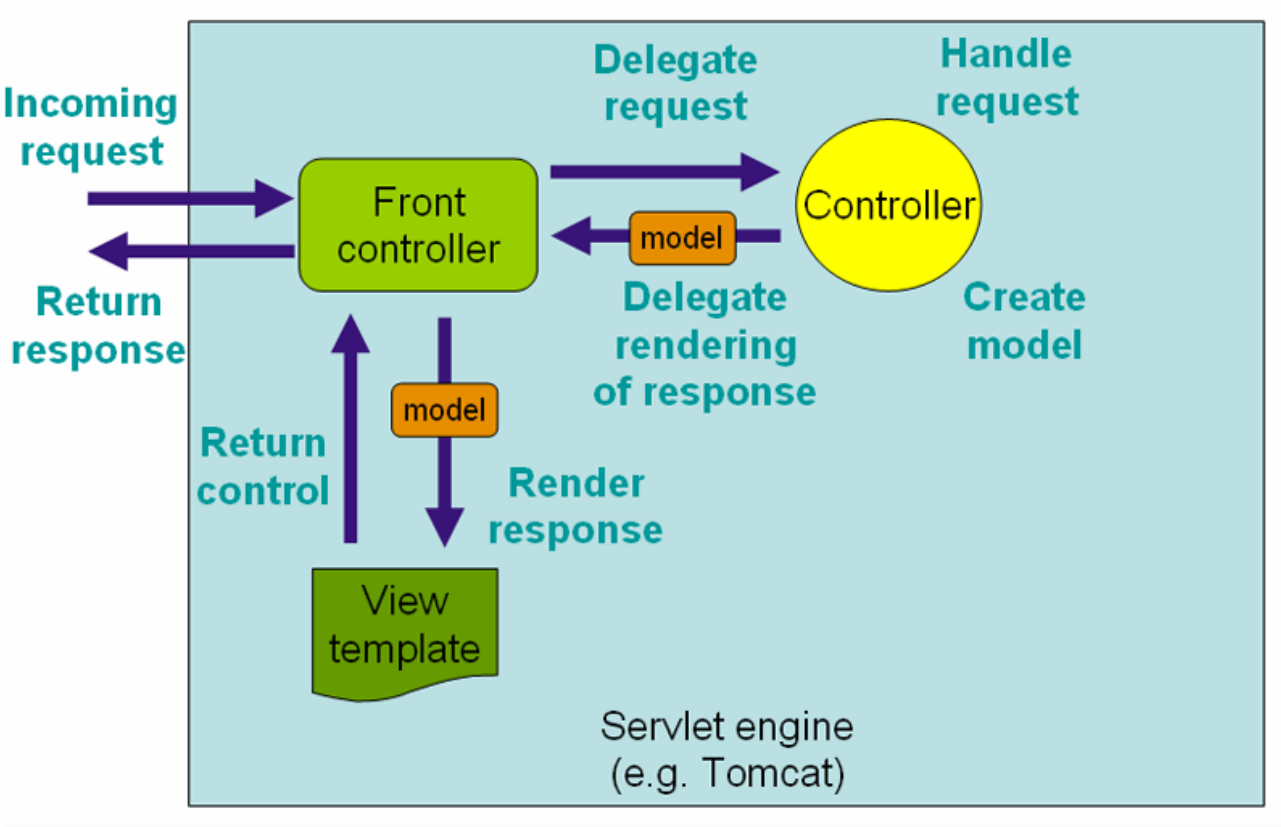
View trong mô hình MVC

- View là một trong 3 thành phần của mô hình ứng dụng MVC
- View có nhiệm vụ hiển thị dữ liệu và là giao diện giúp người dung tương tác với ứng dụng



View trong Spring MVC

- ViewResolver là cơ chế để xử lý tầng view của Spring MVC
- ViewResolver ánh xạ tên của view sang đối tượng view tương ứng
- ViewResolver là bộ phận quyết định xem Front Controller sẽ trả về View thực tế nào.
  Front Controller sẽ xây dựng view trả về bằng cách truy xuất các giá trị đã được gắn cho dữ liệu ở Model, kết quả này sau đó sẽ được trả về cho người dùng.



Views và View Resolvers trong Spring MVC

- Có hai interface trong Spring MVC để phù hợp với cốt lõi hệ thống template của nó đó là:
  - org.springframework.web.servlet.View
  - org.springframework.web.servlet.ViewResolver
- Các view mô hình các trang trong ứng dụng và cho phép sửa đổi và định nghĩa trước các hành vi thông qua việc định nghĩa chúng như các bean.
- ViewResolvers là các đối tượng phụ trách thu thập các đối tượng View cho một hoạt động và khu vực cụ thể.

Views và View Resolvers trong Thymeleaf

- Thymeleaf cung cấp các triển khai cho hai giao diện kể trên cụ thể là:
  - org.thymeleaf.spring5.view.ThymeleafView
  - org.thymeleaf.spring5.view.ThymeleafViewResolver
- Hai lớp này chịu trách nhiệm xử lý các template Thymeleaf như là một kết quả thực thi của các controller.

Cấu hình của ThymeleafViewResolver

```
@Bean
public ThymeleafViewResolver viewResolver(){
     ThymeleafViewResolver viewResolver =
                                   new ThymeleafViewResolver();

     viewResolver.setTemplateEngine(templateEngine());
     viewResolver.setOrder(1);
     viewResolver.setViewNames(new String[] {".html", ".xhtml"});

     return viewResolver;
}
```

Thảo luận

Thymeleaf

Giới thiệu Thymeleaf

- Thymeleaf là một bộ xử lý view được sử dụng cho các ứng dụng web và các ứng dụng độc lập.
- Thymeleaf có thể xử lý HTML, XML, JavaScript, CSS và cả text.
- Thymeleaf cung cấp một cách thức xây dựng template (mẫu) hiện đại và dễ bảo trì.
- Thymeleaf được xây dựng phù hợp với các tiêu chuẩn của web, đặc biệt là HTML5.

Cài đặt Thymeleaf

Thymeleaf cho phép xử lý sáu loại template, mỗi loại được gọi là Template Mode:

- HTML
- XML
- TEXT
- JAVASCRIPT
- CSS
- RAW

Cài đặt Thymeleaf

- Để cài đặt Thymleaf vào trong ứng dụng chúng ta thêm thư viện sau vào trong file build.gradle:

```
compile group: 'org.thymeleaf', name: 'thymeleaf-spring5', version: '3.0.11.RELEASE'
```

Cấu hình Thymeleaf

- Thymeleaf có thể được cấu hình bằng file XML hoặc qua các lớp config (Spring 4).

Cấu hình Thymeleaf với file XML

```
<beans:bean class="org.thymeleaf.spring4.view.ThymeleafViewResolver">
   <beans:property name="templateEngine" ref="templateEngine" />
   <beans:property name="order" value="1" />
   <beans:property name="viewNames" value="*.html,*.xhtml" />
</beans:bean>
<beans:bean id="templateResolver" class=
"org.thymeleaf.templateresolver.ServletContextTemplateResolver">
   <beans:property name="prefix" value="/WEB-INF/templates/" />
   <beans:property name="suffix" value=".html" />
   <beans:property name="templateMode" value="HTML5" />
</beans:bean>
<beans:bean id="templateEngine"
class="org.thymeleaf.spring4.SpringTemplateE
ngine">
   <beans:property name="templateResolver" ref="templateResolver" />
</beans:bean>
```

Hiển thị text

- th:text được sử dụng để hiển thị text
- Ví dụ:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
   <title>Good Thymes Virtual Grocery</title>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
   <link rel="stylesheet" type="text/css" media="all"
       href="../../css/gtvg.css" th:href="@{/css/gtvg.css}" />
</head>
<body>
   <p th:text="#{home.welcome}">Welcome to our grocery store!</p>
</body>
</html>
```

Ví dụ: th:text sử dụng file bên ngoài

```
<p th:text="#{home.welcome}">Welcome to our grocery store!</p>
```

- Các file chứa nội dung có thể là:
  - /WEB-INF/templates/home_en.properties
  - /WEB-INF/templates/home_es.properties
  - /WEB-INF/templates/home_pt_BR.properties
  - /WEB-INF/templates/home.properties
- Nội dung của file home_es.properties:

```
home.welcome=¡Bienvenido a nuestra tienda de comestibles!
```

Demo

Cấu hình View và Thymeleaf

Thảo luận

Cú pháp trong Thymeleaf: Biểu thức, các hằng giá trị, thao tác với chuỗi, toán tử, điều kiện, lặp và hiển thị.

Biểu thức

- Biểu thức với biến: ${...}
- Biểu thức với thuộc tính: *{...}
- Biểu thức với Message: #{...}
- Biểu thức với URL: @{...}
- Biểu thức với phân đoạn: ~{...}

Các hằng giá trị

- Hằng văn bản: 'CodeGym', 'Việt Nam',...
- Hằng số: 0, 34, 3.0, 12.3,...
- Hằng giá trị lô gic: true, false
- Hằng null: null
- Hằng token: one, sometext, main,...

Thao tác với chuỗi

- Toán tử cộng chuỗi: +
- Thay thế giá trị của biến: |The name is ${name}|

Toán tử toán học

- Toán tử số học: +, -, *, /, %
- Toán tử một ngôi (lấy số đối): -

Toán tử logic

- Toán tử nhị phân: and, or
- Toán tử một ngôi: !, not

Toán tử so sánh

- Các toán tử so sánh: >, <, >=, <=
- Toán tử so sánh bằng: ==, !=

Toán tử điều kiện

- Nếu-thì: (if) ? (then)
- Nếu-thì-trái lại: (if) ? (then) : (else)
- Giá trị mặc định: (value) ?: (defaultvalue)

Hiển thị thông điệp

- Ví dụ:

```
home.welcome=¡Bienvenido  a nuestra tienda  de comestibles!
```

```
<p th:utext="#{home.welcome}">Welcome to our grocery store!</p>
```

- Hoặc bao gồm biến:

```
home.welcome=¡Bienvenido  a nuestra tienda  de comestibles, {0}!
```

```
<p th:utext="#{home.welcome(${session.user.name})}">
  Welcome to our grocery store, Sebastian Pepper!
</p>
```

Hiển thị giá trị của biến

- Ví dụ:

```
<p>Today is: <span th:text="${today}">13 february 2011</span>.</p>
```

- Hoặc:

```
<p>
  Today is: <span th:text="${#calendars.format(today,'dd MMMM yyyy')}">13 May
2011</span>
</p>
```

Đánh giá biểu thức đang được lựa chọn

- Ví dụ:

```
<div th:object="${session.user}">
   <p>Name: <span th:text="*{firstName}">Sebastian</span>.</p>
   <p>Surname: <span th:text="*{lastName}">Pepper</span>.</p>
   <p>Nationality: <span th:text="*{nationality}">Saturn</span>.</p>
</div>
```

- Tương đương với:

```
<div>
   <p>Name: <span th:text="${session.user.firstName}">Sebastian</span>.</p>
   <p>Surname: <span th:text="${session.user.lastName}">Pepper</span>.</p>
   <p>Nationality: <span th:text="${session.user.nationality}">Saturn</span>.</p>
</div>
```

Hiển thị liên kết

- Ví dụ:

```
<!-- Will produce 'http://localhost:8080/gtvg/order/details?orderId=3' (plus rewriting) -->
<a href="details.html"
  th:href="@{http://localhost:8080/gtvg/order/details(orderId=${o.id})}">vi
  ew</a>

<!-- Will produce '/gtvg/order/details?orderId=3' (plus rewriting) -->
<a href="details.html" th:href="@{/order/details(orderId=${o.id})}">view</a>

<!-- Will produce '/gtvg/order/3/details' (plus rewriting) -->
<a href="details.html"
th:href="@{/order/{orderId}/details(orderId=${o.id})}">view</a>
```

Hiển thị fragment

- Fragment là các đoạn mã riêng biệt, có thể được nhúng vào các vị trí khác nhau trong template
- Có thể sử dụng th:insert hoặc th:replace để nhúng fragment
- Ví dụ:

```
<div th:insert="~{commons :: main}">...</div>
```

- Hoặc sử dụng như một biến:

```
<div th:with="frag=~{footer :: #main/text()}">
  <p th:insert="${frag}">
</div>
```

Toán tử điều kiện

- Ví dụ:

```
<tr th:class="${row.even}? 'even' :'odd'">
    ...
</tr>
```

Vòng lặp th:each

- Ví dụ:

```
<h1>Product list</h1>

<table>
    <tr>
        <th>NAME</th>
        <th>PRICE</th>
        <th>IN STOCK</th>
    </tr>
    <tr th:each="prod : ${prods}">
        <td th:text="${prod.name}">Onions</td>
        <td th:text="${prod.price}">2.41</td>
        <td th:text="${prod.inStock}? #{true} : #{false}">yes</td>
    </tr>
</table>
```

Trạng thái của vòng lặp th:each

- Vòng lặp th:each cung cấp các trạng thái sau:
  - index: chỉ số của lần lặp hiện tại, bắt đầu từ giá trị 0
  - count: chỉ số của lần lặp hiện tại, bắt đầu từ giá trị 1
  - size: tổng số các phần tử
  - current: phần tử của vòng lặp hiện tại
  - even hoặc odd: vòng lặp chẵn hay lẻ
  - first: vòng lặp đầu tiên
  - last: vòng lặp cuối cùng

```
<tr th:each="prod,iterStat : ${prods}" th:class="${iterStat.odd}? 'odd'">
    ...
</tr>
```

Demo

Cú pháp trong Thymeleaf

Biểu thức điều kiện th:if

- Ví dụ:

```
<a href="comments.html"
  th:href="@{/product/comments(prodId=${prod.id})}"
  th:if="${not #lists.isEmpty(prod.comments)}">view</a>
```

- th:if trả về **true** trong các trường hợp:
  - Một giá trị boolean là true.
  - Một số có giá trị là non-zero (khác 0)
  - Một ký tự có giá trị là non-zero (khác 0)
  - Một chuỗi có giá trị khác với "false", "off" hoặc "no"
  - Nếu một giá trị không phải là boolean, số, ký tự hoặc chuỗi.

Câu lệnh điều kiện th:switch

- Ví dụ:

```
<div th:switch="${user.role}">
   <p th:case="'admin'">User is an administrator</p>
   <p th:case="#{roles.manager}">User is a manager</p>
</div>
```

- Hoặc với giá trị mặc định:

```
<div th:switch="${user.role}">
   <p th:case="'admin'">User is an administrator</p>
   <p th:case="#{roles.manager}">User is a manager</p>
   <p th:case="*">User is some other thing</p>
</div>
```

Thảo luận

Template layout

Sử dụng fragment

- Ví dụ, file *footer.html*:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<body>
<div th:fragment="copy">
  &copy; 2011 The Good Thymes Virtual Grocery
</div>
</body>
</html>
```

- Sử dụng trong file *home.html*:

```
<body>
  <div th:insert="~{footer :: copy}"></div>
</body>
```

Truyền tham số vào fragment

- Ví dụ, khai báo fragment:

```
<div th:fragment="frag (onevar,twovar)">
   <p th:text="${onevar} + ' - ' + ${twovar}">...</p>
</div>
```

- Sử dụngfragment:

```
<div th:replace="::frag (${value1},${value2})">...</div>
<div th:replace="::frag (onevar=${value1},twovar=${value2})">...</div>
```

Demo

Template layout

Hướng dẫn

Hướng dẫn làm bài thực hành và bài tập

Chuẩn bị bài tiếp theo: *Data Binding Form*