



THYMELEAF 2

SLIDE 4.1

PHẦN I: FORM

 RÀNG BUỘC DỮ LIỆU (DATA BINDING)

 KIỂM TRA DỮ LIỆU (DATA VALIDATION)

PHẦN II: LAYOUT & I18N

 TỔ CHỨC LAYOUT (LAYOUT ORGANIZATION)

 TỔ CHỨC TÀI NGUYÊN ĐA NGÔN NGỮ (MESSAGE SOURCE & INTERNATIONALIZATION)





DATA BINDING

- ❑ Data binding được hiểu là buộc dữ liệu từ 1 bean với form nhập và ngược lại.
 - ❖ GET: Hiển thị các thuộc tính của bean lên các trường của form
 - ❖ POST: Chuyển dữ liệu các trường trên form vào các thuộc tính của bean

Student
+ id: String
+ fullname: String
+ gender: Boolean
+ birthday: Date
+ photo: String
+ mark: Double



Data binding
Enter student info please!
Email Address: <input type="text"/>
Fullname: <input type="text"/>
Gender: <input type="radio"/> Male <input type="radio"/> Female
Birthday: <input type="text" value="2024-12-14"/>
Photo: <input type="text" value="photo.png"/>
Average Mark: <input type="text"/>
<input type="button" value="Save"/>

```

<h2>Databinding</h2>
<b th:text="\${message}"></b>
<form th:object="\${student}" action="/student/save" method="post">
    Email Address:
    <input th:field="\${id}"><br>
    Fullname:
    <input th:field="\${fullname}"><br>
    Gender:
    <input th:field="\${gender}" type="radio" value="true">Male
    <input th:field="\${gender}" type="radio" value="false">Female<br>
    Birthday:
    <input th:field="\${birthday}"><br>
    Photo:
    <input th:field="\${photo}"><br>
    Average Mark:
    <input th:field="\${mark}"></i>
    <hr>
    <button>Save</button>
</form>

```

```

@NoArgsConstructor
@AllArgsConstructor
@Data
@Builder
public class Student {
    private String id;
    private String fullname;
    private Boolean gender;
    @Default
    @DateTimeFormat(pattern = "yyyy-MM-dd")
    private Date birthday = new Date();
    @Default
    String photo = "photo.png";
    private Double mark;
}

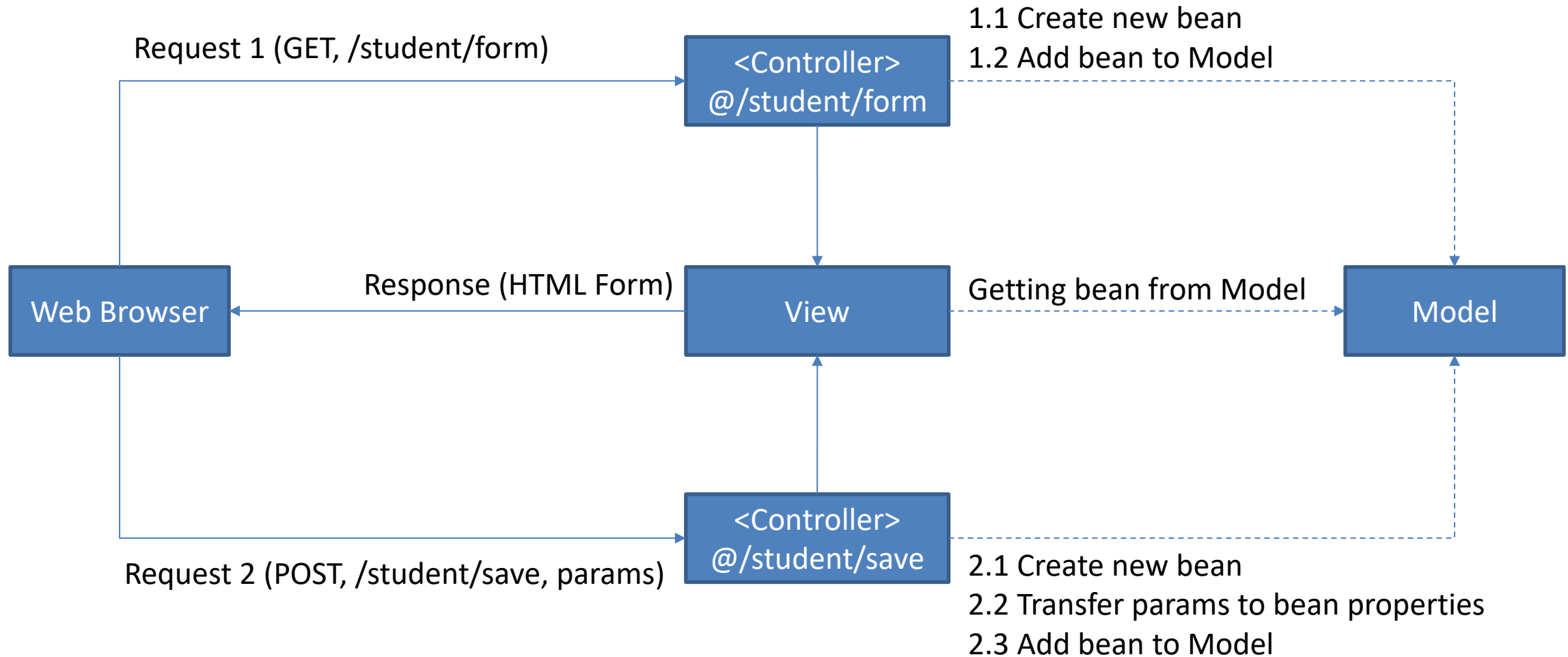
```

@Controller

```
public class DataBindingController {  
    @GetMapping("/student/form")  
    public String showForm(Model model, @ModelAttribute("student") Student student) {  
        model.addAttribute("message", "Enter student info please!");  
        return "/demo/student";  
    }  
  
    @PostMapping("/student/save")  
    public String saveStudent(Model model, @ModelAttribute("student") Student student) {  
        model.addAttribute("message", "Student saved successfully!");  
        return "/demo/student";  
    }  
}
```

❑ @ModelAttribute("student") Student student

- ❖ Tạo bean (đối tượng student)
- ❖ Chuyển đổi các tham số vào các thuộc tính cùng tên của bean (nếu có)
- ❖ Bổ sung bean vào Model với tên biến là student





DATA VALIDATION

Data binding

Vui lòng nhập thông tin sinh viên!

Email Address:

Fullname:

Gender: ☐ Male ☐ Female

Birthday:

Photo:

Average Mark:

Save

Data binding

Vui lòng sửa các lỗi sau!

Email Address: *Không đúng định dạng email*

Fullname: *Chưa nhập họ tên*

Gender: ☐ Male ☐ Female *Chưa chọn giới tính*

Birthday:

Photo:

Average Mark: *Điểm phải từ 0 đến 10*

Save

- ❑ Vào <https://start.spring.io> để lấy đoạn mã sau và khai báo vào pom.xml của dự án

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

BEAN PROPERTY VALIDATION ANNOTATION

@Data @Builder

```
public class Student {  
    @NotBlank(message = "Chưa nhập email")  
    @Email(message = "Không đúng định dạng email")  
    private String id;  
    @NotBlank(message = "Chưa nhập họ tên")  
    private String fullname;  
    @NotNull(message = "Chưa chọn giới tính")  
    private Boolean gender;  
    @NotNull(message = "Chưa nhập ngày sinh")  
    @Default  
    @DateTimeFormat(pattern = "yyyy-MM-dd")  
    private Date birthday = new Date();  
    @NotNull(message = "Chưa nhập hình ảnh")  
    @Default  
    String photo = "photo.png";  
    @NotNull(message = "Chưa nhập điểm")  
    @Range(min = 0, max = 10, message = "Điểm phải từ 0 đến 10")  
    private Double mark;  
}
```

❑ Khai báo thêm các ràng buộc dữ liệu vào class Student

❖ @NotNull

❖ @Email

❖ @NotBlank

❖ @Range

❑ Hệ thống sẽ dựa vào các khai báo này để kiểm tra tính hợp lệ của dữ liệu

❑ *Chú ý: Kiểu nguyên thủy (primitive type) không thể null vì vậy muốn check null phải sử dụng lớp bao (wrapper)*

@Controller

```
public class ValidationController {  
    @RequestMapping("/student/form")  
    public String showForm(Model model, @ModelAttribute Student student) {  
        model.addAttribute("message", "Vui lòng nhập thông tin sinh viên!");  
        return "/demo/student";  
    }  
    @RequestMapping("/student/save")  
    public String validateForm(Model model,  
        @ModelAttribute @Valid Student student, Errors errors) {  
        if(errors.hasErrors()) {  
            model.addAttribute("message", "Vui lòng sửa các lỗi sau!");  
        } else {  
            model.addAttribute("message", "Dữ liệu đã nhập đúng!");  
        }  
        return "/demo/student";  
    }  
}
```

Ghi chú:

- + **@Valid** yêu cầu hệ thống thực hiện validate student
- + Đối số **Errors** phải đặt ngay sau bean cần validate. Đối số này là nơi tích lũy lỗi của bean được validate kể trước

```

<h2>Databinding</h2>
<b th:text="${message}"></b>
<form th:object="${student}" action="/student/save" method="post">
    Email Address:
    <input th:field="*{id}"><i th:errors="*{id}"></i><br>
    Fullname:
    <input th:field="*{fullname}"><i th:errors="*{fullname}"></i><br>
    Gender:
    <input th:field="*{gender}" type="radio" th:value="true">Male
    <input th:field="*{gender}" type="radio" value="false">Female
    <i th:errors="*{gender}"></i><br>
    Birthday:
    <input th:field="*{birthday}"><i th:errors="*{birthday}"></i><br>
    Photo:
    <input th:field="*{photo}"><i th:errors="*{photo}"></i><br>
    Average Mark:
    <input th:field="*{mark}"><i th:errors="*{mark}"></i><br>
    <button>Save</button>
</form>

```

- ❑ th:errors đưa thông báo lỗi của field tại vị trí bạn muốn đặt.
- ❑ Lỗi đã được khai báo trong bean sẽ hiển thị tại vị trí đặt thẻ chưa th:errors.

□ Number

- ❖ @Range
- ❖ @Min
- ❖ @Max
- ❖ @DecimalMin
- ❖ @DecimalMax
- ❖ @Positive
- ❖ @PositiveOrZero
- ❖ @Negative
- ❖ @NegativeOrZero
- ❖ @Digits

□ String

- ❖ @Email
- ❖ @Pattern
- ❖ @NotBlank

□ Collection

- ❖ @NotEmpty
- ❖ @Size

□ Time

- ❖ @Past
- ❖ @PastOrPresent
- ❖ @Future
- ❖ @FutureOrPresent

□ Others

- ❖ @NotNull
- ❖ @AssertFalse
- ❖ @AssertTrue
- ❖ @Null



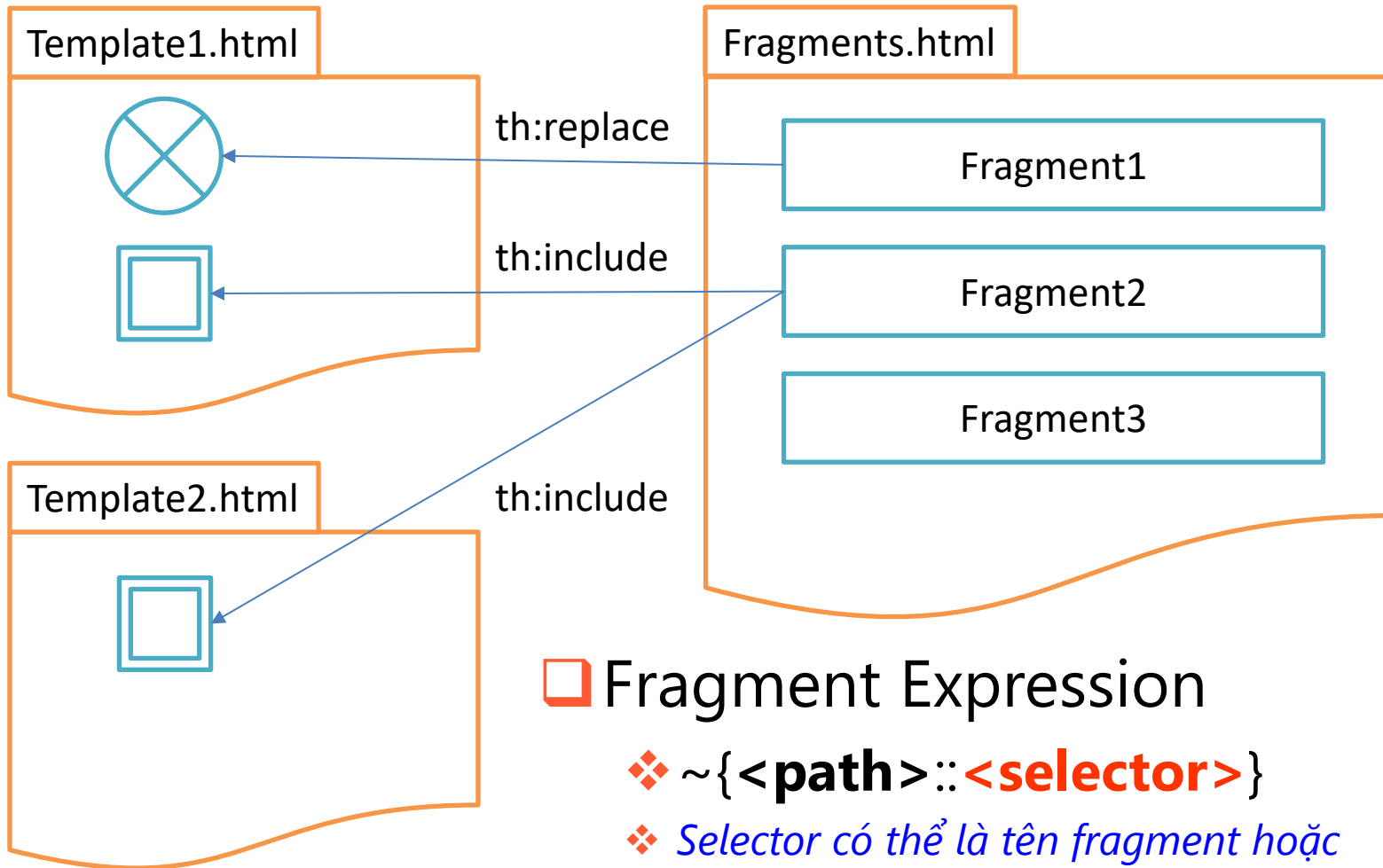
THYMELEAF 2

SLIDE 4.2



FRAGMENT & LAYOUT

FRAGMENT DIRECTIVES AND EXPRESSION



- *Fragment là 1 phân đoạn template được tái sử dụng nhiều lần trên các template khác nhau*
- *Biểu thức Fragment (~{}) được sử dụng để truy xuất fragment*

❑ Fragment Expression

- ❖ $\sim\{<\mathbf{path}>::<\mathbf{selector}>\}$
- ❖ *Selector có thể là tên fragment hoặc CSS Selector (#id, html, .class)*

❑ Fragment Directives

- ❖ th:fragment
- ❖ th:replace
- ❖ th:insert
- ❖ th:remove

frags.html

```
<h1 th:fragment="frag1">Fragment 1</h1>
<h2 th:fragment="frag2">Fragment 2</h2>
```

page.html

```
<div th:replace="~{/demo/frags.html::frag1}"></div>
<div th:insert="~{/demo/frags::frag2}"></div>
<hr>
<div th:replace="~{/demo/frags}"></div>
<hr>
<div th:replace="~{/demo/page::frag3}"></div>
<div th:replace="~{::frag3}"></div>

<template th:remove="all">
  <h3 th:fragment="frag3">Fragment 3</h3>
</template>
```

```
<h1>Fragment 1</h1>
<div><h2>Fragment 2</h2></div>
<hr>
<h1>Fragment 1</h1>
<h2>Fragment 2</h2>
<hr>
<h3>Fragment 3</h3>
<h3>Fragment 3</h3>
```

❑ Fragment Directives

- ❖ th:replace thay thế thẻ chứa directive bằng fragment
- ❖ th:insert chèn fragment vào nội dung thẻ chứa directive
- ❖ th:remove="all" xóa thẻ chứa directive và các thẻ con bên trong. Chú ý: directive này luôn luôn chạy sau cùng vì thứ tự ưu tiên của nó thấp nhất.

❑ Fragment Expressions

- ❖ ~{/demo/frags::frag2} được hiểu là ~{/demo/frags.html::frag2}
- ❖ ~{/demo/frags} toàn template frags.html
- ❖ ~{::frag3} được hiểu là ~{/demo/page::frag3} và frag3 được khai báo trong cùng 1 template page.html
- ❖ Sử dụng CSS Selector (trong trường hợp này là HTML Selector)
 - ~{/demo/frags::h1} thay cho ~{/demo/frags::frag1}
 - ~{::h3} thay cho ~{::frag3}

page.html

```
<div th:fragment="frag(x, y)">
  <div th:replace="${x}"></div>
  <hr>
  <div th:replace="${y}"></div>
</div>
```



```
<div>
  <h4 class="my-title">Page Title</h4>
  <hr>
  <main>Page Content</main>
</div>
```

frags.html

```
<div th:replace="~/demo/frags::frag(~{::my-title}, ~{::main})">
  <h4 class="my-title">Page Title</h4>
  <main>Page Content</main>
</div>
```

FPT Polytechnic

[Tiếng Việt](#) | [English](#)

[Trang chủ](#) | [Giới thiệu](#) | [Liên hệ](#)

TRANG CHỦ

© 2024. All rights reserved.

FPT Polytechnic

[Tiếng Việt](#) | [English](#)

[Trang chủ](#) | [Giới thiệu](#) | [Liên hệ](#)

GIỚI THIỆU

© 2024. All rights reserved.

FPT Polytechnic

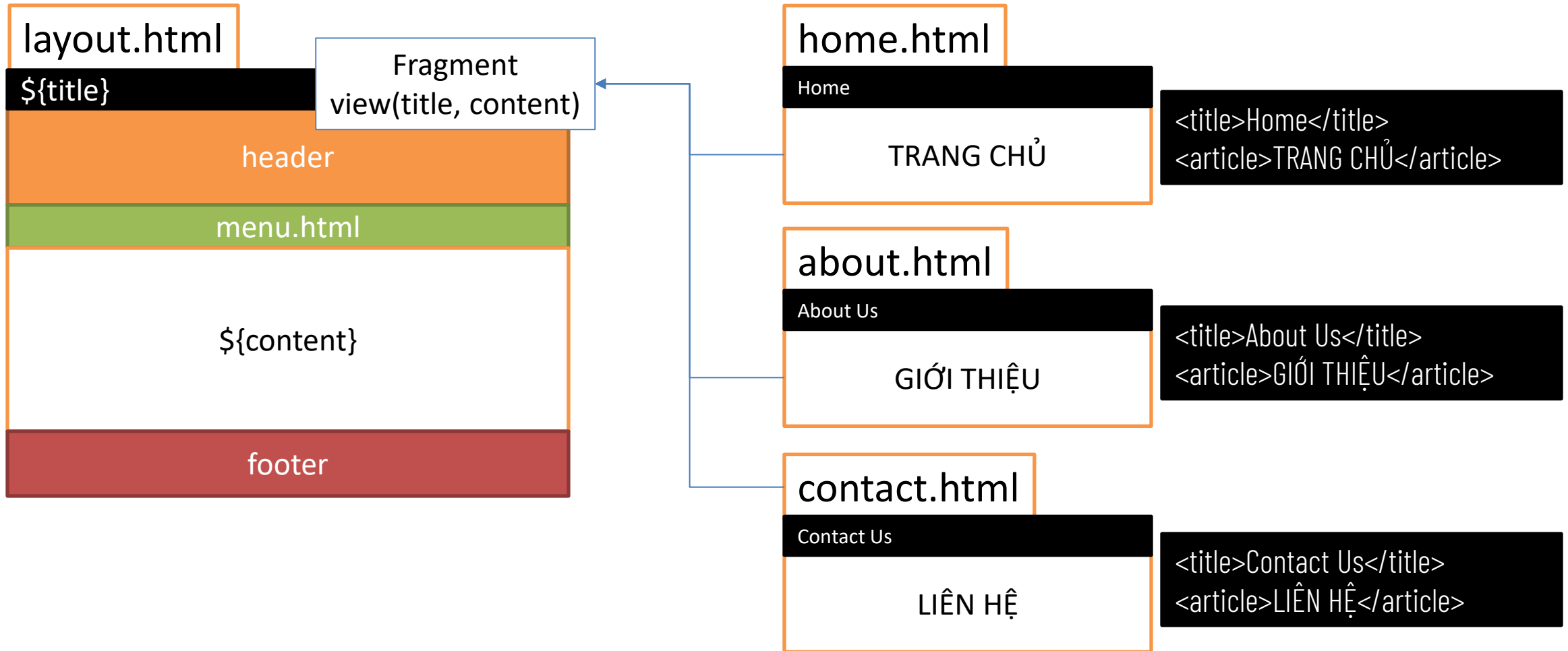
[Tiếng Việt](#) | [English](#)

[Trang chủ](#) | [Giới thiệu](#) | [Liên hệ](#)

LIÊN HỆ

© 2024. All rights reserved.

*Nội dung trang
thay đổi*



/demo/layout.html

```
<html th:fragment="view(title, content)">
  <head>
    <title th:replace="${title}">TITLE</title>
  </head>
  <body>
    <header><h1>FPT Polytechnic</h1></header>
    <nav th:replace="~/demo/menu">MENU</nav>
    <article th:replace="${content}">CONTENT</article>
    <footer><hr>&copy; 2024. All rights reserved.</footer>
  </body>
</html>
```

/demo/menu.html

/demo/menu.html

```
<nav>
```

```
<hr>
```

```
<a href="/home/index">Trang chủ</a> |
```

```
<a href="/home/about">Giới thiệu</a> |
```

```
<a href="/home/contact">Liên hệ</a>
```

```
<hr>
```

```
<div style="position: fixed; top: 5px; right: 5px;">
```

```
<a href="?lang=vi">Tiếng Việt</a> |
```

```
<a href="?lang=en">English</a>
```

```
</div>
```

```
</nav>
```

FPT Polytechnic

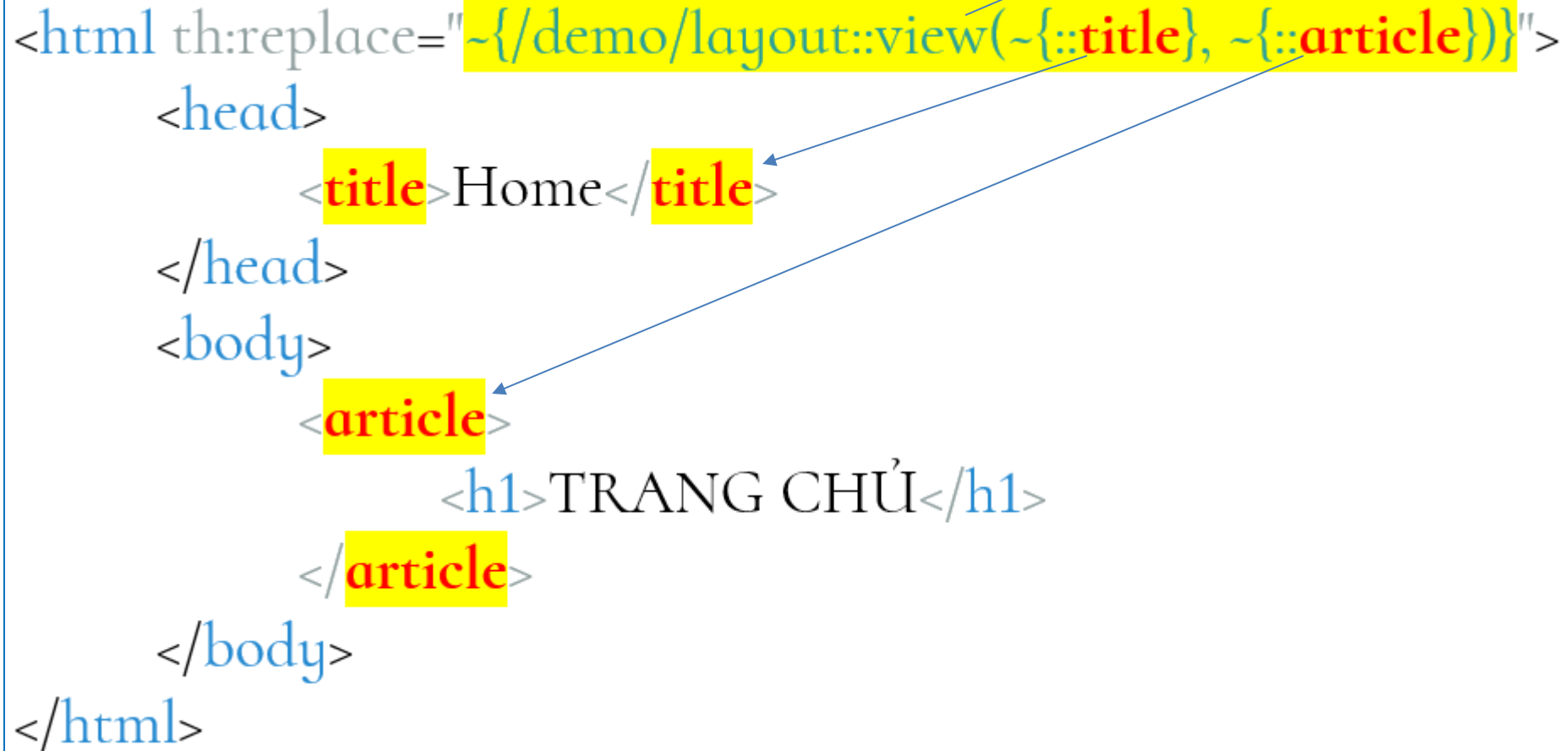
[Tiếng Việt](#) | [English](#)

[Trang chủ](#) | [Giới thiệu](#) | [Liên hệ](#)

/demo/home.html

/demo/layout.html::view(title, content)

```
<html th:replace="~{/demo/layout::view(~{::title}, ~{::article})}">
  <head>
    <title>Home</title>
  </head>
  <body>
    <article>
      <h1>TRANG CHỦ</h1>
    </article>
  </body>
</html>
```



FPT Polytechnic

[Trang chủ](#) | [Giới thiệu](#) | [Liên hệ](#)

TRANG CHỦ

© 2024. All rights reserved.

[Tiếng Việt](#) | [English](#)

```
<html>
  <head>
    <title>Home</title>
  </head>
  <body>
    <header><h1>FPT Polytechnic</h1></header>
    <nav>
      <hr>
      <a href="/home/index">Trang chủ</a> |
      <a href="/home/about">Giới thiệu</a> |
      <a href="/home/contact">Liên hệ</a>
      <hr>
      <div style="position: fixed; top: 5px; right: 5px;">
        <a href="?lang=vi">Tiếng Việt</a> |
        <a href="?lang=en">English</a>
      </div>
    </nav>
    <article>
      <h1>TRANG CHỦ</h1>
    </article>
    <footer><hr>&copy; 2024. All rights reserved.</footer>
  </body>
</html>
```



MESSAGE SOURCE & i18N

The screenshot illustrates the configuration of a multi-language application in an IDE. The Package Explorer on the left shows the project structure, including the `MessageConfig.java` file. The main editor displays the `MessageConfig.java` code, which uses `@Configuration`, `@Bean`, and a custom `MessageSource` to load messages from the classpath:resources/layout. The `layout.properties` file is shown below, defining menu items in Vietnamese. The `menu.html` file is shown on the right, using Thymeleaf syntax to display the menu items.

```
8 @Configuration
9 public class MessageConfig {
10     @Bean("messageSource")
11     public MessageSource getMessageSource() {
12         ReloadableResourceBundleMessageSource ms = new ReloadableResourceBundleMessageSource();
13         ms.setBasenames("classpath:resources/layout");
14         ms.setDefaultEncoding("utf-8");
15         return ms;
16     }
17 }
```

layout.properties

```
1 menu.home=Home
2 menu.about=About
3 menu.contact=Contact
```

menu.html

```
1 <nav>
2   <hr>
3   <a href="/home/index" th:text="#{menu.home}">Trang chủ</a> |
4   <a href="/home/about" th:text="#{menu.about}">Giới thiệu</a> |
5   <a href="/home/contact" th:text="#{menu.contact}">Liên hệ</a>
6   <hr>
7   <div style="position: fixed; top: 5px; right: 5px;">
8     <a href="?lang=vi">Tiếng Việt</a> |
9     <a href="?lang=en">English</a>
10  </div>
11 </nav>
```

❑ Cú pháp đặt tên file tài nguyên đa ngôn ngữ

- ❖ `<basename>_<language code>.properties`
- ❖ Mỗi ngôn ngữ 1 file properties có mã ngôn ngữ khác nhau nhưng cùng key
 - `layout_en.properties` (mặc định là `_en`)
 - `layout_vi.properties`

❑ Vị trí đặt file tài nguyên

 `src/main/resources`

▼  `i18n`

 `layout_vi.properties`

 `layout.properties`

`layout_vi.properties`

menu.home=Trang chủ
menu.about=Giới thiệu
menu.contact=Liên hệ

`layout.properties`

menu.home=Home
menu.about=About
menu.contact=Contact

CẤU HÌNH NẠP FILE TÀI NGUYÊN ĐA NGÔN NGỮ

@Configuration

```
public class MessageConfig {  
    @Bean("messageSource")  
    public MessageSource getMessageSource() {  
        ReloadableResourceBundleMessageSource ms  
            = new ReloadableResourceBundleMessageSource();  
        ms.setBasenames("classpath:i18n/layout");  
        ms.setDefaultEncoding("utf-8");  
        return ms;  
    }  
}
```

Chú ý:

- ❖ Gọi phương thức `setBasenames()` để chỉ ra vị trí các file tài nguyên đa ngôn ngữ.
- ❖ Basename không bao gồm mã ngôn ngữ và phần mở rộng của file tài nguyên

```
<nav>  
  <hr>  
  <a href="/home/index" th:text="#{menu.home}">Trang chủ</a> |  
  <a href="/home/about" th:text="#{menu.about}">Giới thiệu</a> |  
  <a href="/home/contact" th:text="#{menu.contact}">Liên hệ</a>  
  <hr>  
  <div style="position: fixed; top: 5px; right: 5px;">  
    <a href="?lang=vi">Tiếng Việt</a> |  
    <a href="?lang=en">English</a>  
  </div>  
</nav>
```

- ❑ Sử dụng Message Expression `{key}` để truy xuất tài nguyên theo đa ngôn ngữ

CHỌN VÀ DUY TRÌ NGÔN NGỮ ĐÃ CHỌN

@Configuration

```
public class MessageConfig implements WebMvcConfigurer{
```

```
    @Bean("messageSource")
```

```
    public MessageSource getMessageSource() {...}
```

```
    @Override
```

```
    public void addInterceptors(InterceptorRegistry registry) {
```

```
        LocaleChangeInterceptor interceptor = new LocaleChangeInterceptor();
```

```
        interceptor.setParamName("lang");
```

```
        registry.addInterceptor(interceptor);
```

```
    }
```

```
    @Bean("localeResolver")
```

```
    public LocaleResolver getLocaleResolver() {
```

```
        CookieLocaleResolver localeResolver = new CookieLocaleResolver();
```

```
        localeResolver.setCookiePath("/");
```

```
        localeResolver.setCookieMaxAge(Duration.ofDays(30));
```

```
        localeResolver.setDefaultLocale(new Locale("vi"));
```

```
        return localeResolver;
```

```
    }
```

```
}
```

1. Lớp cấu hình cần thực thi theo interface **WebMvcConfigurer** và override phương thức **addInterceptor()** để xử lý liên kết chọn ngôn ngữ

```
<a href="?lang=vi">Tiếng Việt</a> |  
<a href="?lang=en">English</a>
```

2. Cấu hình bổ sung bean **LocaleResolver** để thiết lập phương pháp duy trì ngôn ngữ đã chọn (Session hoặc Cookie). Thiết lập các thông số phù hợp cho Cookie.

[Tiếng Việt](#) | [English](#)

FPT Polytechnic

[Trang chủ](#) | [Giới thiệu](#) | [Liên hệ](#)

TRANG CHỦ

© 2024. All rights reserved.

Menu thay đổi theo ngôn ngữ được chọn

[Tiếng Việt](#) | [English](#)

FPT Polytechnic

[Home](#) | [About](#) | [Contact](#)

TRANG CHỦ

© 2024. All rights reserved.

☑ PHẦN I:

☑ RÀNG BỘC DỮ LIỆU (DATA BINDING)

☑ KIỂM TRA DỮ LIỆU (DATA VALIDATION)

☑ PHẦN II:

☑ TỔ CHỨC LAYOUT (LAYOUT ORGANIZATION)

☑ TỔ CHỨC TÀI NGUYÊN ĐA NGÔN NGỮ (MESSAGE SOURCE & INTERNATIONALIZATION)





FPT Education

FPT POLYTECHNIC

Thank you