Danh Mục

[1. Data binding : 1](#_Toc511407798)

[2. Cách dùng : 2](#_Toc511407799)

[3. Thực Tế : 4](#_Toc511407800)

[4. Sử dụng radiobuttons : 5](#_Toc511407801)

[5. Select và Multi Select : 5](#_Toc511407802)

[6. Kiểm tra lỗi ( validation ) phía Server 7](#_Toc511407803)

[7. Kiểm tra lỗi Client : 13](#_Toc511407804)

[8. Dùng chung message cho Client & Server : 14](#_Toc511407805)

[9. Mở Rộng ( Server ) : 14](#_Toc511407806)

[10. Custom Validation 15](#_Toc511407807)

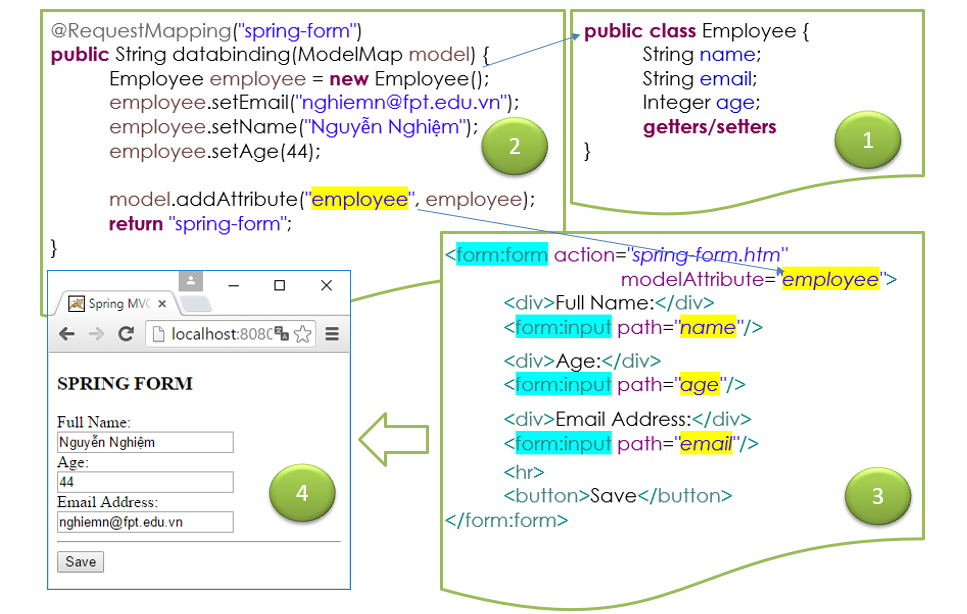
1. **Data binding :**

\_ Liên kết dữ liệu của bean trong model đến các giao diện ( input, checkbox, radio, hidden, … ). Dữ liệu bean thay đổi => các control thay đổi theo.

\_ Có 2 cách để binding :

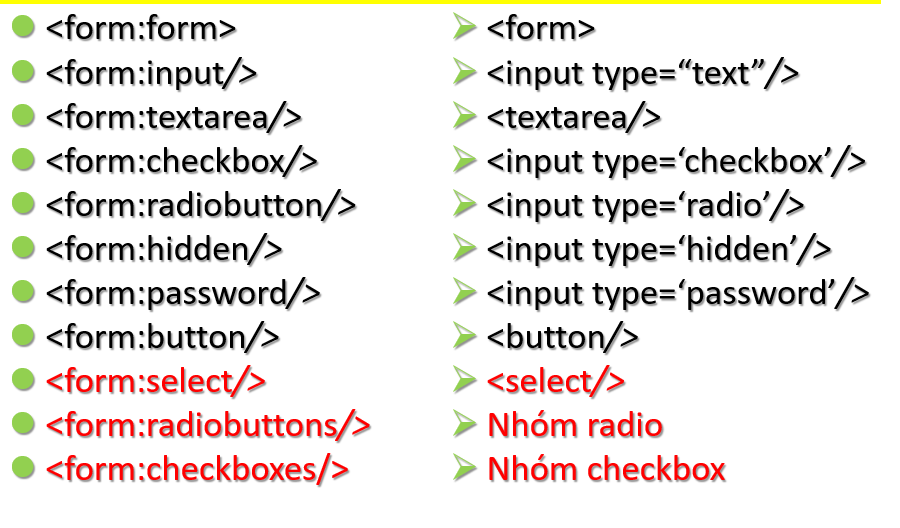
+ **C1** : Nhận tham số ( theo 4 cách ) kết hợp với model.addAttribute. Sau đó dùng EL để gán giá trị cho các thẻ form ( thường là thuộc tính value ). Hạn chế : code phức tạp, xử lý khó khăn với các control phức tạp : combobox, listbox, radiobuttons, checkboxs, khó kiểm tra lỗi ( validation )

+ **C2** : Spring Form : đơn giản, hỗ trợ validation, dễ dàng buộc dữ liệu của bean lên các control.



1. **Cách dùng :**

\_ Khai báo module taglib : /tags/form có prefix : form ( hoặc bất kỳ )



**Truyền dữ liệu từ controller -> view :**

\_ Khai báo form:form có các thuộc tính cần thiết : **action, modelAttribute**, mặc định method là POST

\_ **modelAttribute** bên view phải trùng với tên của model.addAttribute tại controller.

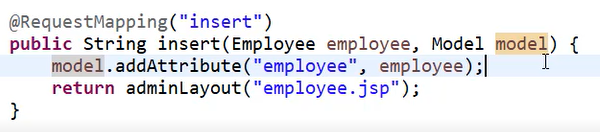
\_ Truy vấn thuộc tính của model thông qua .

\_ form:input path( sinh ra **id** và **name** )

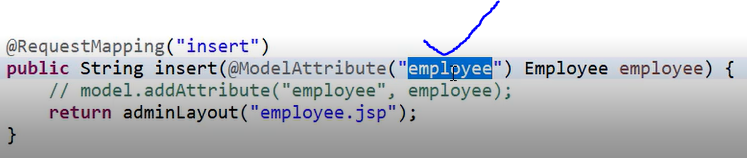
**Nhận dữ liệu từ view -> controller :**

Cách 1 :

Demo : sử dụng java bean để nhận dữ liệu. Sau khi nhận được dùng model binding lên form lại => dữ liệu không bị mất.



Cách 2 : sử dụng **@ModelAttribute**

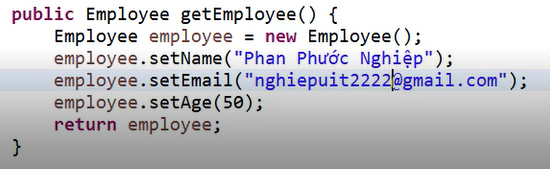


* **Lưu ý** : @ModelAttribute có name trùng với name bên view.

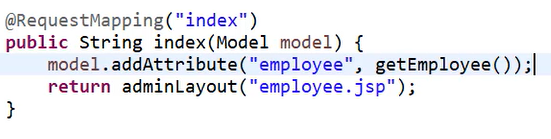
Có 2 cách để sử dụng **@ModelAttribute :** trước 1 đối số và trên 1 phương thức.

\_ Trước 1 đối số : như trên

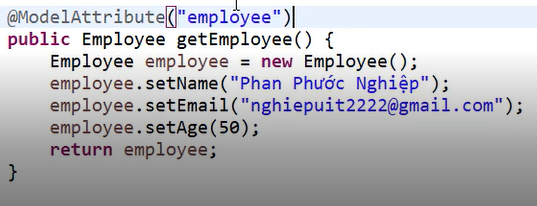
\_ Trên 1 phương thức :



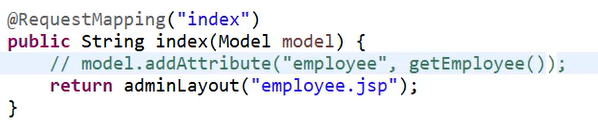
Sau đó ta binding giá trị lên form :



Thay vì như vậy ta sẽ viết cách như sau :



Và bỏ đi bước addAttribute



1. **Thực Tế :**

**\_** Đọc dữ liệu từ CSDL lên.

**\_** Gán vào model lập tức form có giá trị. ( model.addAttribute )

1. **Sử dụng radiobuttons :**

\_ Cách thông thường :

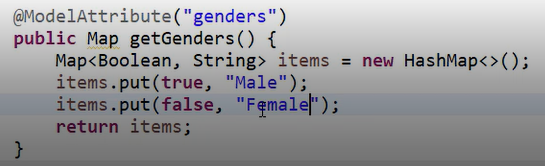


\_ Cách thường hay sử dụng : khai báo giá trị trong **controller**

+ radiobuttons có thuộc tính **items** là danh sách các giá trị ( vd : nam, nữ đối với giới tính ).

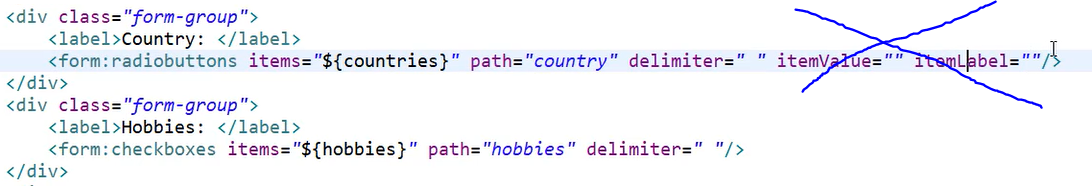


* **Items** sẽ là @ModelAttribute. Viết theo dạng phương thức. Kết quả trả về là 1 mảng Boolean[] hoặc HashMap<Boolean, String> ( bool hiển thị giá trị, string hiển thị nhãn ).

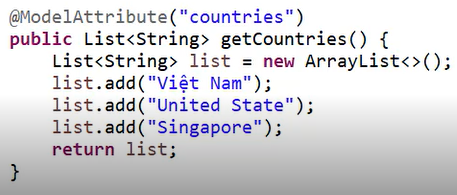


1. **Select và Multi Select :**

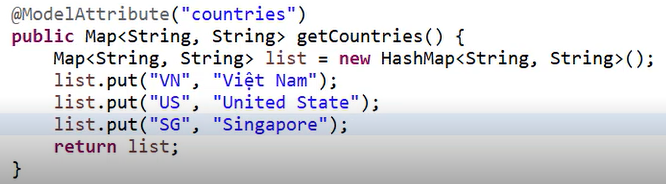
\_ Tạo control :



**+ Chuỗi String :**



**+ Dùng HashMap :** Phải thiết lập lại thuộc tính country của đối tượng ràng buộc lên form. **Student.setCountry ( “VN” )**



+ **Dùng Object :**



+ Khi sử dụng **Object** ta cần bổ sung thêm **itemValue** và **itemLabel**



\_ Gán dữ liệu lên form : kiểm tra selected



**Select :**

Tóm lại :

\_ Nguồn dữ liệu 3 loại : mảng, tập hợp, map

\_ Giá trị

\_ Kiểm lỗi phải kiểm trong model, kiểm trong action – controller.

1. **Kiểm tra lỗi ( validation ) phía Server**

\_ Thư viện : **Bean Validation API** & **Hibernate Validator**

\_ Kiểm tra lỗi trên model.

\_ Gồm 3 phần :

+ Loại lỗi

+ Tên Model

+ Tên Trường

VD :

Length.student.name = Họ tên ít nhất 10 ký tự.

Range.student.point = Điểm phải từ 1-10

Pattern.student.email = Email không đúng định dạng

NotEmpty.student.id = MSSV không được bỏ trống

**Các lỗi :**

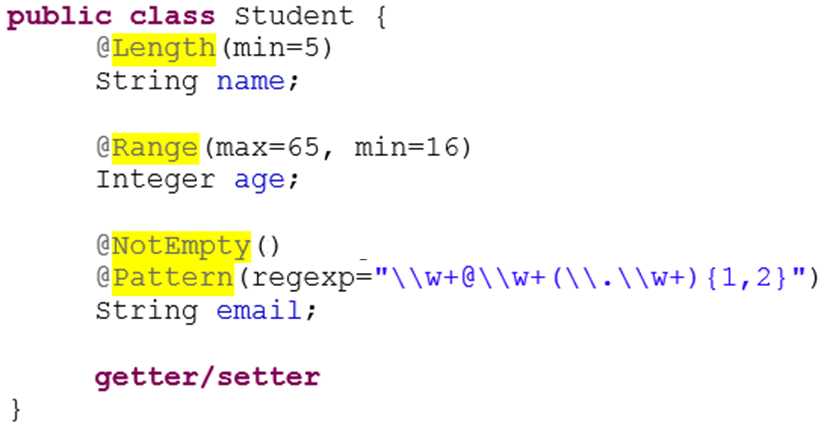
* AssertFalse: must be false
* AssertTrue : must be true
* DecimalMax : must be less than or equal to {value}
* DecimalMin : must be greater than or equal to {value}
* Digits : numeric value out of bounds (<{integer} digits>.<{fraction} digits> expected)
* Future : must be in the future
* Max : must be less than or equal to {value}
* Min : must be greater than or equal to {value}
* NotNull : may not be null
* Null : must be null
* Past : must be in the past
* Pattern : must match "{regexp}"
* Size : size must be between {min} and {max}
* CreditCardNumber: invalid credit card number
* Email : not a well-formed email address
* Length : length must be between {min} and {max}
* NotBlank : may not be empty
* NotEmpty : may not be empty
* Range : must be between {min} and {max}
* SafeHtml : may have unsafe html content
* ScriptAssert : script expression "{script}" didn't evaluate to true
* URL : must be a valid URL
* typeMismatch=type mismatch

**Các bước :**

* B1: Thêm luật kiểm lỗi vào các trường dữ liệu.
* B2: Kiểm lỗi trong action
* B3: Hiển thị thông báo lỗi trên view
* B2: Khai báo thông báo lỗi

**Thực hiện :**

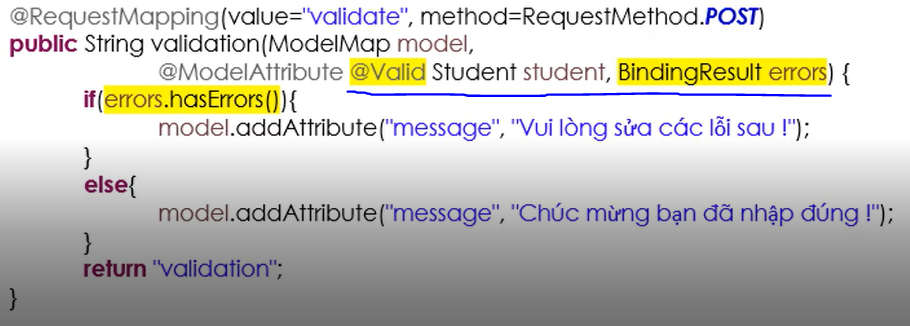
**B1 :**



**B2 :**

Trong controller

Khi ta submit form :



@ **Valid** có thể thay thế bằng **@Validated**

BindingResult : chứa các lỗi.

**B3** : Hiển thị lỗi trên giao diện : **path** quan trọng.



**Lưu ý :** hiển thị tất cả lỗi để **path=”\*”**

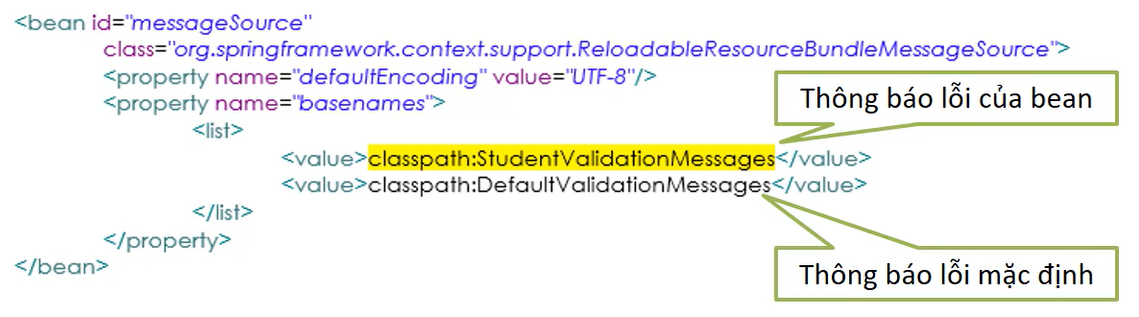
**B4 : Khai báo câu thông báo lỗi : khai báo file properties**

StudentValidationMessages.**properties**

Length.student.name = tên quá ngắn, quá dài.

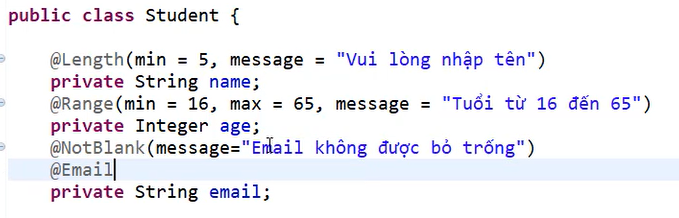
Quy luật : key – value

**B4.1 :** Cấu hình thông báo lỗi - chỉ ra file properties ở trên



Không có lấy lỗi mặc định.

**Demo : tạo dự án mới – demo**



**Tiến hành hiển thị lỗi.**

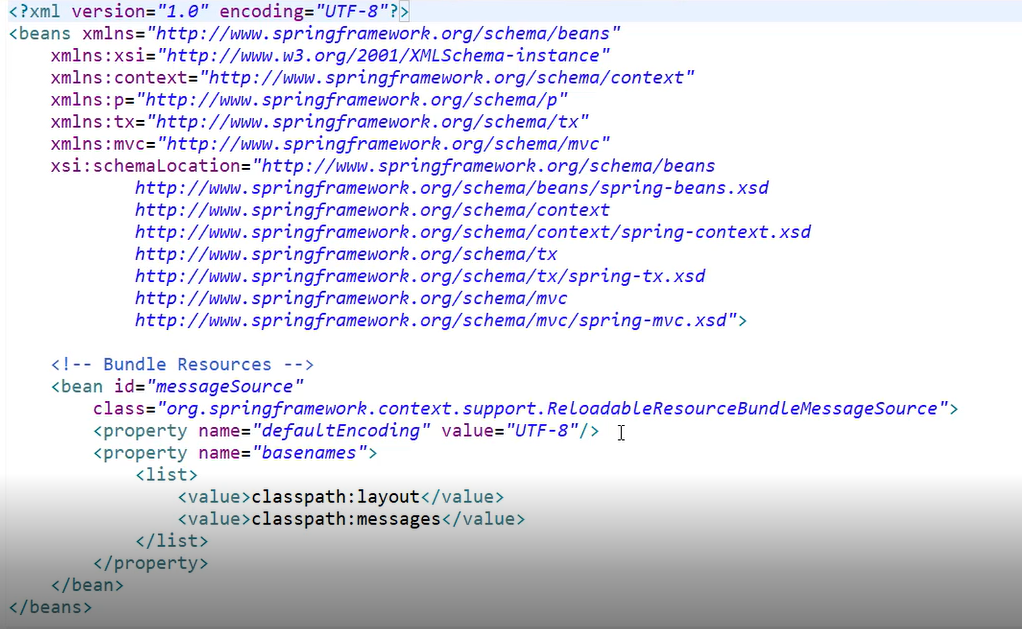
\_ Tạo Source Folder tên : srcx ( có thể tạo file properties trong src/main/resource nếu là project maven )

\_ Tạo File bên trong srcx đặt tên : message.**properties**

\_ Save chỉnh UTF-8

Length.student.name = Họ tên quá ngắn.

**Cấu hình : spring-config-resources.xml**



Liệt kê các file properties trong list. ( Không lấy đuôi .properties )

**\*\* Lưu ý :** phân biệt **NotEmpty, NotBlank, NotNull**

**NotNull :** thuộc javax.validation

**Tập hợp : NotEmpty**

**Chuỗi : Tất cả**

**Integer** : NotNull

**Int** : Không kiểm được.

**Sai Kiểu dữ liệu** : **typeMismatch** bên properties. Không có bên class.

**Digit :** số số nguyên.

**SafeHtml :** chống javascript hack, phá, tấn công website, **XSS**

**ScriptAssert**

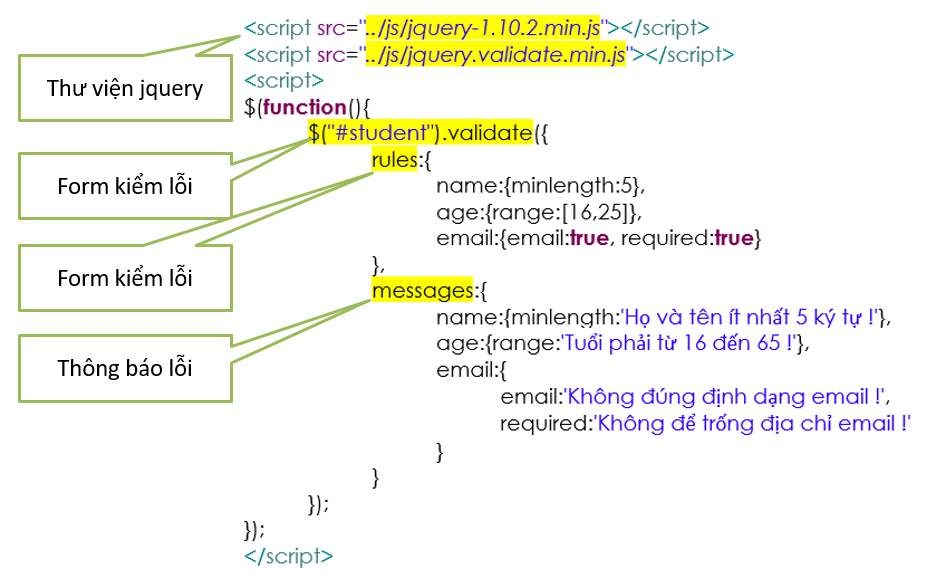
**URL :** có phải http:// địa chỉ trang web hay ko?

**Pattern :** bao quát.

1. **Kiểm tra lỗi Client :**

\_ Sử dụng Jquery & Jquery Validation Plugin. ( Thứ tự )

\_ Tham khảo trang chủ :



\_ Nhúng thư viện vào.

\_ **rules :** các quy luật kiểm lỗi



\_ **messages :** thông báo lỗi

\_ Khi nhập đúng hết mới submit server. Ngược lại hiển thị lỗi

1. **Dùng chung message cho Client & Server :**

B1 : Nhúng thư viện tags

<%@ taglib uri="http://www.springframework.org/tags" prefix="s" %>

B2 : Tại message gọi : gọi key đã khai báo trên server ( file properties )

‘<s:message code=”Length.student.name” />’

1. **Mở Rộng ( Server ) :**
   1. Regular Expression :

**@Pattern(regexp=”^[a-zA-z0-9]{5}”)**

VD : nhập biển số xe, sđt, cmnd.

* 1. Giới thiệu typeMismatch

VD : khi nhập string vào interger.

1. **Custom Validation :**

**Step 1a :** Create @**Annotation\_Name** annotation.

\_ Tạo package : com.nghiepuit.validation

\_ New Annotation => Đặt tên ( VD : CourseCode ) => sinh ra **@interface**

\_ Sử dụng **@Constraint(validatedBy = CourseCodeConstraintValidator.class)** : đây là helper class chứa quy luật validation, logic, business.

\_ Sử dụng **@Target({ ElementType.METHOD, Element.FIELD })**

\_ Sử dụng **@Rerention(RetentionPolicy.RUNTIME) :** chú thích thời điểm annotation này được dùng. Lúc chương trình chạy ( run time ).

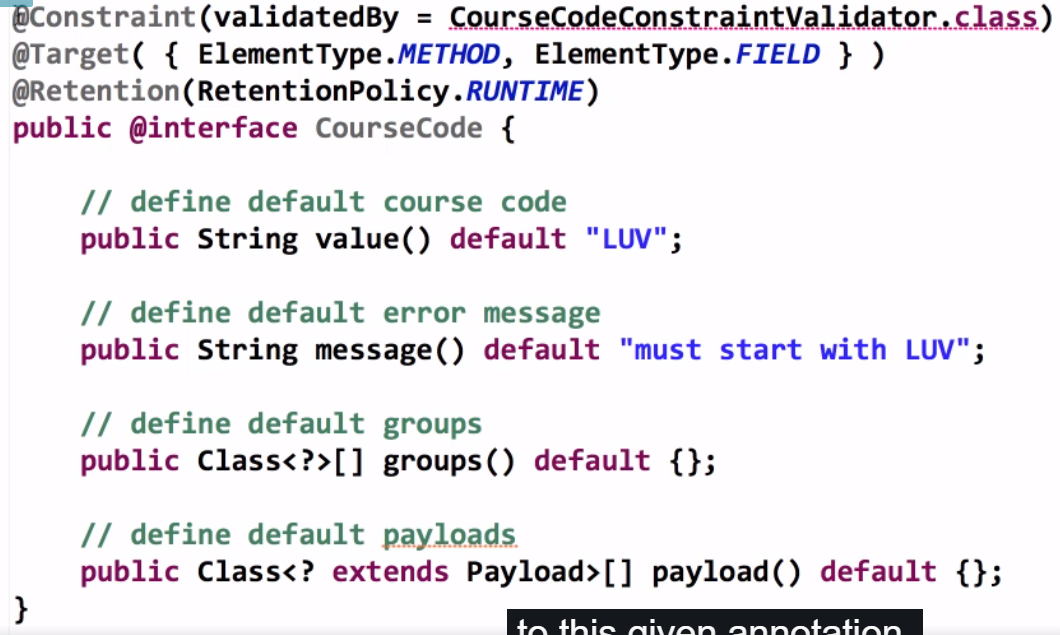
\_ Khai báo thuộc tính cho annotation này :

+ Giá trị mặc định.

+ Thông báo lỗi

+ Nhóm mặc định : có thể nhóm các rang buộc liên quan.

+ Payloads mặc định : thuộc lớp javax.validation



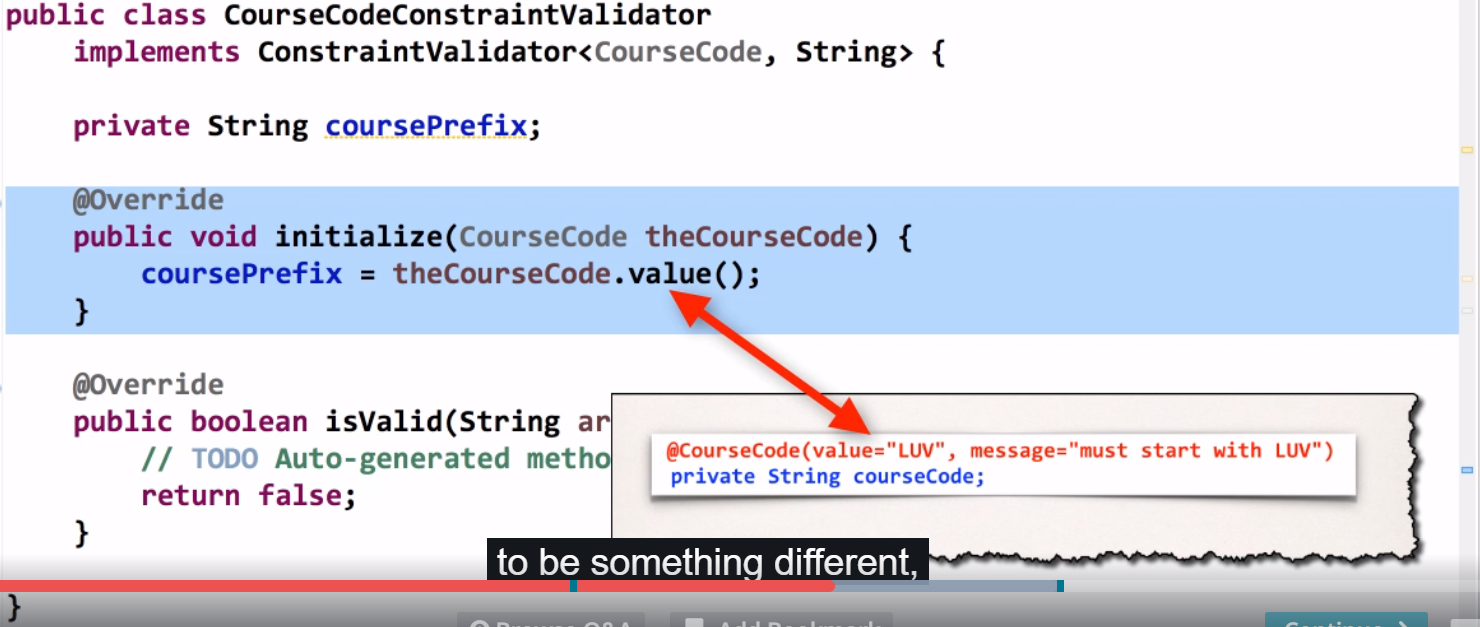
Sử dụng : 

**Step 1b : Tạo class CourseCodeConstraintValidator**

**\_** Thực thi interface : **implements ConstraintValidator<CourseCode, String>**

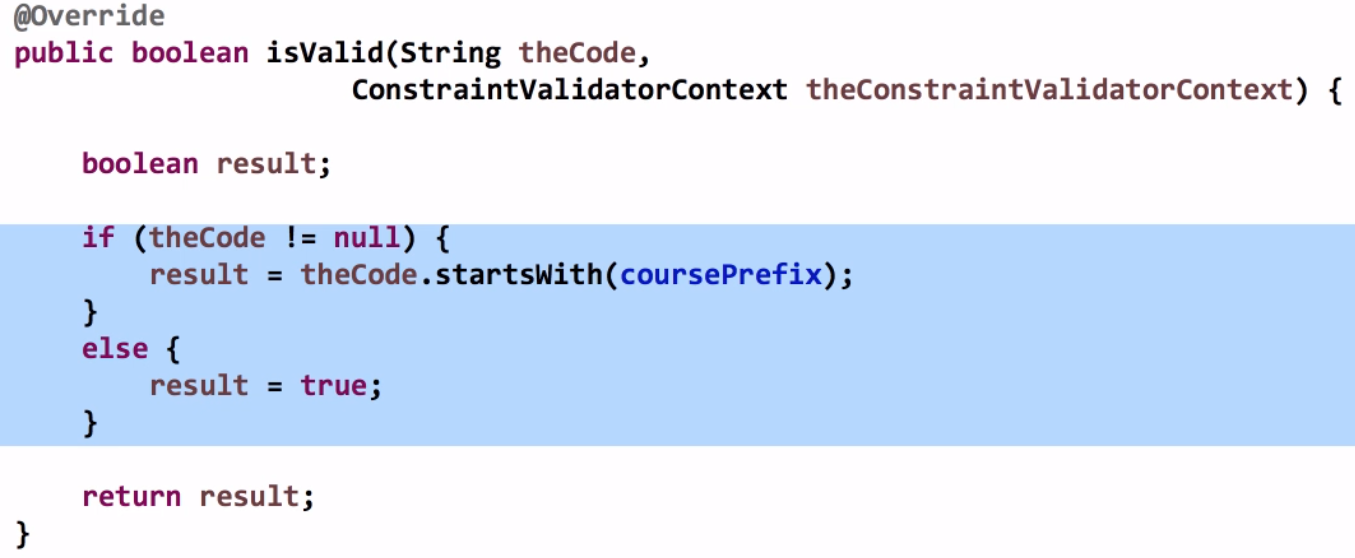
\_ @Override 2 phương thức : init và isValid.

Tạo biến coursePrefix để sử dụng.

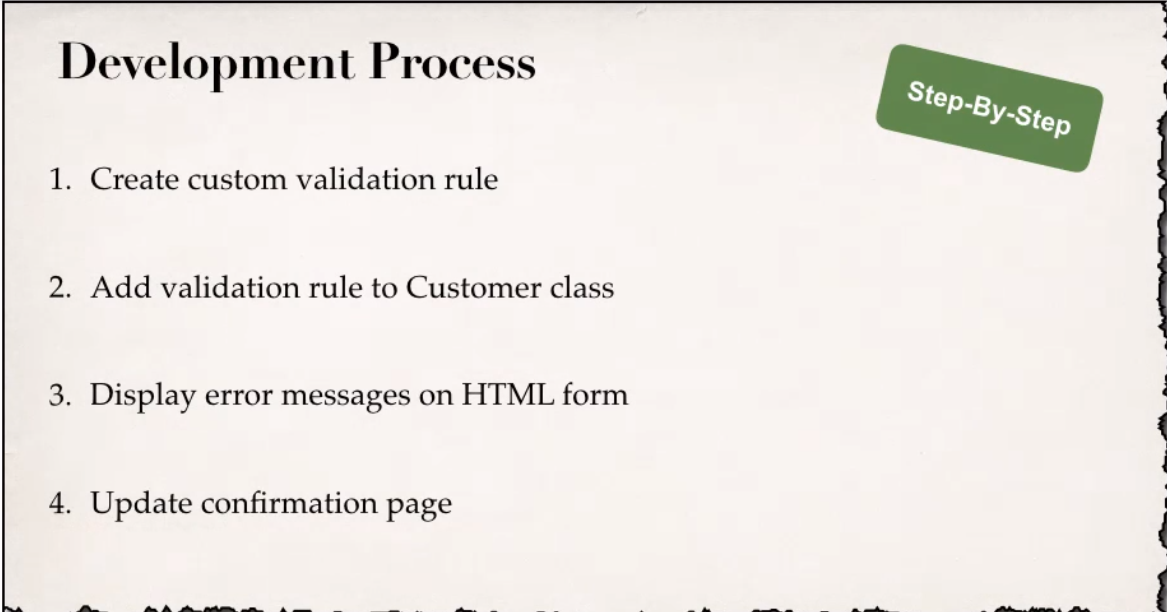


\_ Spring MVC sẽ gọi isValid : là nơi xử lý business logic validation

+ Tham số 1 : giá trị nhận từ form do người dùng gửi lên.



Kiểm tra bắt đầu với coursePrefix . Trả về true hoặc false.



Bước 1 : làm rồi

Bước 2 : Add vào class cần validate



Do class custom validation đã khai báo **value** và **message** nên ta không cần khai báo lại.

Bước 3 : Hiển thị ra form bằng form:error

Bước 4 : Mở rộng test như sau :

