**Bài 20: Validation trong Laravel**

**1, Validation là gì?**

-Validation là tiến trình kiểm tra cái gì đó có tuân theo một qui tắc đã cho không. Và cụ thể đối với form nó là các ràng bộc về dữ liệu của form nhằm đảm bảo tính chặt chẽ của ứng dụng.

**2, Validation Form Trong Laravel.**

-Trong Laravel hỗ trợ chúng ta mạnh về validation dữ liệu.

**Validation Trực tiếp trên Controller.**

-Để validation dữ liệu thì bắt buộc trên controller của chúng ta phải khai báo lớp **http Request**có namespace

use Illuminate\Http\Request;

-Tiếp đó chúng ta cần gọi hàm với cú pháp:

$this->validate($request,$pattern,$messenger,$customName);

Trong đó:

* **$request:** Là biến tham chiếu đối tượng Request mà các bạn khai báo ở đầu hàm.
* **$pattern:** Là mảng định nghĩa dữ liệu đầu vào của các trường trong form.
* **$messenger:** Là mảng chứa nội dung báo lỗi (Nếu muốn thay đổi).
* **$customName:** Là mảng chứa các tên cho các trường trong form.

-**Chú ý:**Nếu không cần thay đổi nội dung lỗi,tên input mặc định thì không cần truyền giá trị cho 2 tham số này.

**VD1:**Mình muốn validation dữ liệu cho một form có 2 input: name,age thì mình sẽ phải viết đoạn validation như sau:

public function handleForm(Request $request)

{

$this->validate($request,

[

'name' => 'required|min:5|max:255',

'age' => 'required|integer|max:3',

],

[

'required' => ':attribute Không được để trống',

'min' => ':attribute Không được nhỏ hơn :min',

'max' => ':attribute Không được lớn hơn :max',

'integer' => ':attribute Chỉ được nhập số',

],

[

'name' => 'Tiêu đề',

'age' => 'Tuổi',

]

);

}

-Đoạn trên mình đã validation cho trường name không được để trống và dữ liệu nhập vào phải có độ dài lơn hơn 5 và nhỏ hơn 255, trường age không được để trống và dữ liệu nhập vào phải là số.

**Validation với lớp validator.**

-Để sử dụng được lớp này trong controller thì chúng ta cần phải khai báo namespace của 2 lớp : **http Request** và **Validator**.

use Validator;

use Illuminate\Http\Request;

Tiếp đó khởi tạo validator với cú pháp:

Validator::make($request,$pattern,$messenger,$customName);

-Nếu validator thành công thì sẽ trả về true và ngược lại

Trong đó:

* **$request:** Là biến tham chiếu đối tượng Request mà các bạn khai báo ở đầu hàm (Nhưng ở đây phải là mảng).
* **$pattern:** Là mảng định nghĩa dữ liệu đầu vào của các trường trong form.
* **$messenger:** Là mảng chứa nội dung báo lỗi (Nếu muốn thay đổi).
* **$customName:** Là mảng chứa các tên cho các trường trong form.

-Chú ý: Cũng giống validate nếu không cần thay đổi nội dung lỗi,tên input mặc định thì không cần truyền giá trị cho 2 tham số này.

**VD2:** Giống VD1 giờ mình sẽ làm theo validator.

$validate = Validator::make(

$request->all(),

[

'title' => 'required|min:5|max:255',

'content' => 'required',

],

[

'required' => ':attribute Không được để trống',

'min' => ':attribute Không được nhỏ hơn :min',

'max' => ':attribute Không được lớn hơn :max',

],

[

'title' => 'Tiêu đề',

'content' => 'Nội dung',

]

);

if ($validate->fails()) {

return View('ValidationView')->withErrors($validate);

}

-Ở đây ValidationView là view ban đầu submit lên còn các thông số withErrors mình đã giới thiệu ở các phần trước rồi.

**3, Hiển Thị Thông báo lỗi Trong Laravel.**

-Khi mà form submit không thành công do không nhập đúng yêu cầu thì bạn cần phải thông báo lỗi ra trình duyệt. Và Laravel có hỗ trợ chúng ta một số phương thức để hiển thị lỗi.

**Kiểm tra có tồn tại lỗi không.**

-Để kiểm tra có tồn tại lỗi hay không thì Laravel có cung cấp cho chúng ta phương thức **has()**với cú pháp:

$errors->has('inputName');

// output true||false

-inputName: Ở đây là tên của input cần kiểm tra.

VD: Kiểm tra xem input title có tồn tại lỗi không.

if($errors->has('title')){

//true

}

**Lấy hết lỗi.**

-Để lấy hết lỗi mọi người sử dụng phương thức **all()** với cú pháp

$errors->all();

**Lấy lỗi đầu tiên của input.**

-Để lấy lỗi đầu tiên của input Laravel có cung cấp cho chúng ta phương thức **first()** với cú pháp:

$errors->first('inputName')

VD: Lấy lỗi đầu tiên của input có name='content' nếu có:

if ($errors->has('content') {

echo $errors->first('content');

}

**4, Các validation Hay dùng.**

| **Rules** | **Chú Thích** |
| --- | --- |
| Required | Không được để trống ô dữ liệu. |
| Boolean | Chỉ cho phép nhập vào các giá trị 0,1,true,false,'0','1' . |
| confirmed | Kiểm tra 2 input có trùng nhau không, input sau bắt buộc phải có tiền tố \_confirmation. VD: password,password\_confirmation. |
| dimensions | -Giới hạn chiều rộng chiều cao của ảnh. VD: 'avatar' => 'dimensions:min\_width=100,min\_height=200'  - Các thông số khác: *min\_width*, *max\_width*, *min\_height*, *max\_height*, *width*, *height*, *ratio.* |
| email | -Dữ liệu nhập vào phải là email. |
| file | -Dữ liệu nhập vào phải là file. |
| image | -Dữ liệu nhập vào phải là file ảnh(jpeg, png, bmp, gif, or svg). |
| in:foo,bar,.. | -Dữ liệu nhập vào phải nằm trong list. |
| integer | -Dữ liệu nhập vào phải là số nguyên. |
| json | -Dữ liệu nhập vào phải là json |
| max:value | -Độ dài lớn nhất của dữ liệu nhập vào. VD: max:255. |
| min:value | -Độ dài nhỏ nhất của dữ liệu nhập vào. VD: min:5. |
| mimetypes:text/pain,... | -Kiểu dữ liệu của file phải nằm trong list.  VD: 'video' => 'mimetypes:video/avi,video/mpeg,video/quicktime' |
| numeric | -Dữ liệu nhập vào pải là số. |
| unique:table | -Dữ liệu nhập vào phải là duy nhất trong bảng.  VD: 'username'=>'unique:users'. |

-Ngoài ra các bạn có thể xem thêm tại: [đây](https://laravel.com/docs/5.3/validation#available-validation-rules)