VALIDATION FORM

Validation form là gì?

Dùng kiểm tra các giá trị nhập từ các thành phần của form có đúng hay không, có phù hợp với các yêu cầu được khai báo hay không.

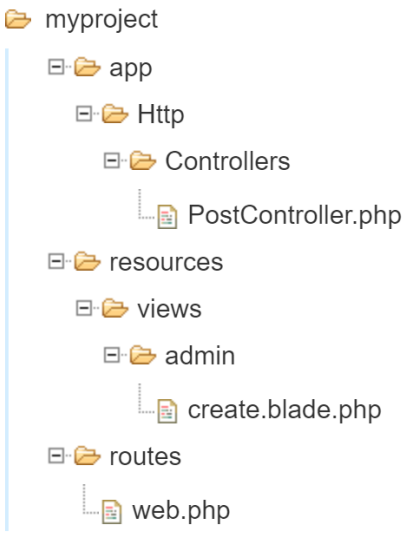
Các giá trị validation dễ điều khiển, đa dạng.

Kết quả sẽ trả về lỗi nếu không đạt yêu cầu validation, nội dung lỗi có thể tùy biến thay đổi dễ dàng.

Thực hiện một validation form

Trước tiên ta cần một trang View có nội dung là một form với vài nội dung nhập dữ liệu. Một Controller điều khiển các giá trị validation. Một Routing điều hướng trang hiển thị nhập và trang hiện thị dữ liệu lỗi hay thông báo thành công sau khi form được nhấn Submit.

Tạo nội dung các file trên theo cấu trúc thư mục sau:



TẠO TRANG VIEW: create.blade.php

Đặt file này trong thư mục */resources/views/admin/*, với nội dung sau:

```
<!doctype html>
<html>
<head>
    <title>Login Form</title>
</head>
<body>
    <form method="post" action="/admin/create">
        @csrf
        <p>Title<br>
            <input type="text" name="title" value=""></p>
```

```
            <p>Description<br>
                <textarea rows="5" cols="40" name="description"></textarea></p>
            <p><button type="submit">Submit</button></p>
        </form>
    </body>
    </html>
```

- `method="post"` - phương thức post của form.
- `@csrf` - tạo một chuỗi CSRF ngẫu nhiên
- `action="/admin/create"` - khi click button Submit, sẽ post dữ liệu tới */admin/create*, cũng là chính trang View này.
- `name="title"` - dữ liệu sẽ được nhận thông qua name là title.
- `name="description"` - dữ liệu sẽ được nhận thông qua name là description.
- `type="submit"` - Dữ liệu sẽ gửi thông qua hành động click button.

TẠO MỘT CONTROLLER: PostController.php

Tạo Controller bằng lệnh Artisan:

```
    php artisan make:controller PostController
```

Controller *PostController.php* này được đặt trong thư mục */app/Http/Controllers/*, với nội dung mặc định sau:

```
    <?php
    namespace App\Http\Controllers;
    use Illuminate\Http\Request;
    class PostController extends Controller
    {
        //
    }
```

Thêm nội dung cho Controller này với function *showform()* - hiển thị trang xem nội dung form:

```
    <?php
    namespace App\Http\Controllers;
    use Illuminate\Http\Request;
    class PostController extends Controller
    {
        public function showform() {
            return view('admin/create');
        }
```

```
    }
```

- `return view('admin/create');` - hiển thị nội dung View *create.blade.php*, khi thực hiện điều hướng từ Routing tới Controller `PostController@showform`

TẠO MỘT ROUTING điều hướng nội dung function **`showform()`** trong Controller

```
Route::get('/admin/create', 'PostController@showform');
```

- `Route::get` - nhận yêu cầu từ người dùng, trả về function `showform`.
- `/admin/create` - Khi gõ đường dẫn này lên trình duyệt, thì sẽ tiến hành xử lý Controller `PostController` tại function `showform` của Controller *PostController.php*
- Nếu tạo file với nội dung giống như trên, gõ lên trình duyệt đường dẫn *http://localhost:82/admin/create* sẽ thấy được nội dung trang */admin/create* vừa tạo:

TẠO MỘT FUNCTION TRONG CONTROLLER ĐIỀU KHIỂN validation: PostController.php

Tạo function *validationform* bên trong controller *PostController.php* để tiến hành kiểm tra các giá trị nhập từ form:

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;

class PostController extends Controller
{
    public function showform() {
        return view('admin/create');
    }
    public function validationform(Request $request) {
        echo "<pre>";
            print_r($request->all());
        echo "</pre>";
        $this->validation($request,[
```

```
            'title'=>'required|max:50',
            'description'=>'required'
        ]);
    }
}
```

- `$this->validation()`: Kiểm tra các giá trị form
- `'title'=>'required|max:255'`: Kiểm tra giá trị `title` nhập từ form, với giá trị `required` (bắt buộc), `max:255` (giới hạn 50 ký tự).
- `'description'=>'required'`: Kiểm tra giá trị `description` nhập từ form, với giá trị `required` (bắt buộc)
- `print_r($request->all())`: in tất cả các giá trị nhập từ form.

TẠO MỘT ROUTING điều hướng nội dung function **`validationform()`** trong Controller

```
Route::get('/admin/create', 'PostController@showform');
Route::post('/admin/create', 'PostController@validationform');
```

- `Route::post` - Lấy dữ liệu post từ form trả về function `validationform` bên trong Controller.
- `/admin/create` - điều hướng trả kết quả về trang */admin/create*.

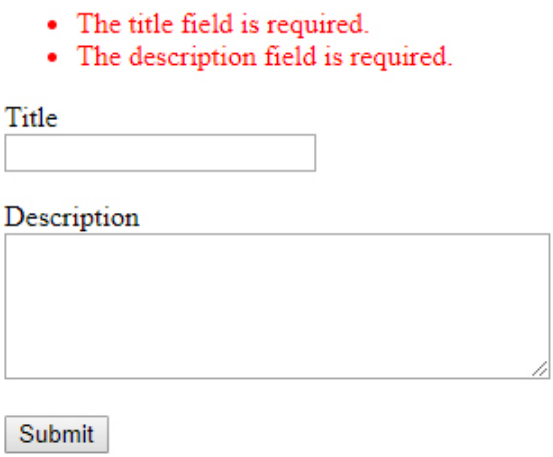
VIẾT THÔNG BÁO LỖI TẠI VIEW: create.blade.php

Xuất ra thông báo lỗi, nếu không thỏa các điều kiện validation được viết tại controller *PostController.php*

```
<!doctype html>
<html>
<head>
    <title>Login Form</title>
    <style type="text/css">
        .error-message { color: red; }
    </style>
</head>
<body>
    @if (count($errors) > 0)
    <div class="error-message">
        <ul>
            @foreach ($errors->all() as $error)
            <li>{{ $error }}</li>
            @endforeach
```

```
            </ul>
        </div>
        @endif
        <form method="post" action="/admin/create">
            @csrf
            <p>Title<br>
                <input type="text" name="title" value=""></p>

            <p>Description<br>
                <textarea rows="5" cols="40" name="description"></textarea></p>

            <p><button type="submit">Submit</button></p>
        </form>
    </body>
    </html>
```

- `count($errors) > 0` - Đếm lỗi lớn hơn 0.
- `@foreach ($errors->all() as $error)` - Chạy vòng lặp @foreach để hiển thị các lỗi thỏa điều kiện if bên trên.
- `.error-message { color: red; }` - CSS hiển thị text màu đỏ cho text báo lỗi.
- Khi này, nếu nhập title và description không đúng điều kiện trong Controller, thì sẽ báo lỗi, ví dụ ta bỏ trống 2 field này và click button `Submit`:



- Nội dung thông báo lỗi là mặc định có trong Laravel.
- Nếu nhập title và description đúng các điều kiện trong Controller, thì sẽ in ra dữ liệu của title và description vừa nhập:

```
    Array
    (
```

```
        [_token] => wYhCZEcm1mlzud8NlFD0ITDSoTShNHLSKAJdppvt
        [title] => hello
        [description] => Đây là description
    )
```

Các giá trị validation thường dùng

Các giá trị validation thường dùng

- Laravel hỗ trợ rất nhiều các giá trị validation hữu ích.
- Các giá trị validation có thể tham khảo thêm tại trang chính của Laravel Available validation Rules

Cú pháp viết một validation

Cú pháp cơ bản

```
    'name_từ_form'=>'property1|property2|...'
```

Ví dụ

```
    public function validationform(Request $request) {
        $this->validation($request,[
            'title'=>'required|string',
            'tel'=>'required|numeric'
        ]);
    }
```

Cú pháp với giá trị

```
    'name_từ_form'=>'property1:value1|property2:value2|...'
```

Ví dụ

```
    public function validationform(Request $request) {
        $this->validation($request,[
            'code'=>'digits:3',
            'age'=>'between:18,30'
        ]);
    }
```

Validation thường dùng

Rule	Mô tả	VD
accepted	Dùng khi kiểm tra các điều khoản sử dụng.	'option' =>'accepted'
active_url	Dữ liệu nhập phải phải là url theo function checkdnsrr của PHP.	'path' =>'active_url'

after:date	Ngày nhập phải sau ngày đã cho.	'start_date' => 'required|date|*after:tomorrow*'
after_or_equal:date	Ngày nhập phải sau hoặc bằng ngày đã cho.	'start_date' => 'required|date|*after_or_equal:tomorrow*'
alpha	Dữ liệu nhập phải là chữ.	'name' =>'alpha'
alpha_dash	Dữ liệu nhập phải là chữ hoặc số, bao gồm dấu gạch ngang "-" và dưới "_".	'address' =>'alpha_dash'
alpha_num	Dữ liệu nhập phải là số.	'number' =>'alpha_num'
array	Dữ liệu nhập phải là mảng PHP.	'my_array' =>'array'
bail	Sẽ dừng kiểm tra nếu 1 validate đầu tiên không thỏa mãn.	'name' =>'bail|require|max:255'
before:date	Ngày nhập phải trước ngày đã cho.	'end_date' => 'required|date|*before:today*'
before_or_equal:date	Ngày nhập phải trước hoặc bằng với ngày đã cho.	'end_date' => 'required|date|*before_or_equal:today*'
between:min,max	Dữ liệu nhập phải nằm trong giá trị min và max, giá trị có thể là chuỗi, số và file.	'age' =>'between:18,30'
boolean	Dữ liệu nhập phải là có giá trị boolean: true hoặc false, 1 hoặc 0, "1" hoặc "0".	'remember_me' =>'boolean'
confirmed	Dữ liệu nhập phải trùng khớp với name_confirmation	có 2 field: *name="password"* và *name="password_confirmation"* thì giá trị Validation sẽ là: 'password' =>'required|*confirmed*|min:8'
date	Dữ liệu nhập phải là ngày tháng, phù hợp với hàm strtotime của PHP.	'date' =>'date'
date_equals:date	Dữ liệu nhập phải bằng với ngày tháng đã cho, phù hợp với hàm strtotime của PHP.	'date' =>'date_equals:22/03/2019'
date_format:*format*	Dữ liệu nhập phải giống định dạng với *format*, định dạng phải phù hợp với hàm date_parse_from_format của PHP.	'date' =>'date_format:d/m/Y'
different:field	Dữ liệu nhập phải khác với giá trị *field*.	VD: giá trị *name="Nguyễn Văn Tèo"*, chúng ta muốn giá trị *other* nhập vào khác với name, khi đó ta viết: 'other' =>'name'
digits:value	Dữ liệu nhập phải là số, có độ dài bằng *value*.	'code' =>'digits:3'
digits_between:min,max	Dữ liệu nhập phải là số, nằm trong khoảng min và max.	'code' =>'digits:3,8'

dimensions	Dữ liệu nhập phải là một ảnh, có kích thước theo quy định, có thể là: min_width, max_width, min_height, max_height, width, height, ratio.	'image' =>'dimensions:min_width=50, min_height=100'
distinct	Dữ liệu nhập phải là mảng, và không có giá trị lặp lại.	'foo.*.id' => 'distinct'
email	Dữ liệu nhập phải là địa chỉ email.	'email' =>'email'
exists:table,column	Dữ liệu nhập phải là cột có trong bảng tồn tại trong Database.	'column' =>'exists:news,title'
file	Dữ liệu nhập phải là một file tải lên thành công.	'file' =>'file'
filled	Dữ liệu nhập không được trống.	'name' =>'filled'
gt:field	Dữ liệu nhập phải lớn hơn trường field, và giống kiểu dữ liệu như: Strings, numerics, arrays.	'number' =>'gt:200'
gte:field	Dữ liệu nhập phải lớn hơn hoặc bằng trường field, và giống kiểu dữ liệu như: Strings, numerics, arrays.	'number' =>'gte:200'
image	Dữ liệu nhập phải là hình có định dạng: jpeg, png, bmp, gif, svg.	'photo' =>'image'
in:foo,bar,...	Dữ liệu nhập phải thuộc danh sách các giá trị.	
in_array:anotherfield	Dữ liệu nhập phải tồn tại trong giá trị của anotherfield	$A = array("1", "2", "3", "4"); 'val' =>'in_array:$A'
integer	Dữ liệu nhập phải thuộc kiểu integer.	'number' =>'integer'
ip	Dữ liệu nhập phải là kiểu địa chỉ ip.	'pathIp' =>'ip'
json	Dữ liệu nhập phải là chuỗi JSON.	'jString' =>'json'
lt:field	Dữ liệu nhập phải nhỏ hơn trường field, và giống kiểu dữ liệu như: Strings, numerics, arrays.	'number' =>'lt:200'
lte:field	Dữ liệu nhập phải nhỏ hơn hoặc bằng trường field, và giống kiểu dữ liệu như: Strings, numerics, arrays.	'number' =>'lte:200'
max:value	Dữ liệu nhập phải <= value.	'number' =>'max:10'
mimetypes:text/plain,..	Dữ liệu nhập phải đúng với kiểu MIME.	'video' => 'mimetypes:video/avi, video/mpeg, video/quicktime'
MIME	Sử dụng các quy định cơ bản của MIME.	'photo' => 'mimes:jpeg,bmp,png'
min:value	Dữ liệu nhập phải phải có giá trị tối thiểu bằng value.	'number' =>'min:5'
not_in:foo, bar, ...	Dữ liệu nhập không thuộc danh sách được cung cấp.	'toppings' =>'Rule::notIn(['sprinkles', 'cherries'])'

not_regex:pattern	Dữ liệu nhập phải khác với dạng thức cung cấp.	'email' => 'not_regex:/^.+$/i'
nullable	Dữ liệu nhập có thể là null, hữu ích dành cho việc nhập các số hay chuỗi có khả năng null.	'text' =>'nullable'
numeric	Dữ liệu nhập phải có dạng chữ số.	'phone' =>'numeric'
present	Dữ liệu nhập phải xuất hiện trong input, nhưng có thể trống.	'value' =>'present'
regex: pattern	Dữ liệu nhập phải giống với dạng thức cung cấp.	'email' =>'regex:/^.+@.+$/i'
required	Dữ liệu bắt buộc phải được nhập, không được để trống.	'name' =>'required'
required_if:anotherfield, value,...	Dữ liệu nhập là bắt buộc và thỏa mãn điều kiện nào đó.	'role_id' => Rule::requiredIf($request->user()->is_admin)
required_unless:another field, value,...	Dữ liệu nhập là bắt buộc và không được theo điều kiện nào đó	'role_id' => Rule::requiredIf($request->user()->is_guess)
required_with:foo, bar	Dữ liệu nhập là bắt buộc và phải chứa ít nhất các giá trị cho trước.	'number' => 'required_with:3,5,7'
required_with_all:foo, bar,...	Dữ liệu nhập là bắt buộc và phải chứa tất cả các giá trị cho trước.	'number' => 'required_with_all:3,5,7'
required_without:foo, bar, ...	Dữ liệu nhập là bắt buộc và không được chứa ít nhất các giá trị cho trước.	'number' => 'required_without:3,5,7'
required_without_all:foo , bar, ...	Dữ liệu nhập là bắt buộc và không được chứa tất cả các giá trị cho trước.	'number' => 'required_without_all:3,5,7'
same:field	Dữ liệu nhập phải trùng khớp với *field*.	'email' =>'same:old_email'
size:value	Dữ liệu nhập phải cùng kiểu với value, chuỗi thì là string, số thì là integer,...	'file' =>'size:300'
starts_with:foo,bar,...	Dữ liệu nhập phải bắt đầu một giá trị đưa trước.	'number' =>'starts_with:5'
string	Dữ liệu nhập phải là string, nếu muốn dữ liệu nhập có thể mang giá trị *null*, thì cần thêm điều kiện *nullable*.	'title' =>'string'
timezone	Dữ liệu nhập phải có giá trị timezone được xác định theo function timezone_identifiers_list của PHP.	'time' =>'timezone'
unique:table, column, except, idColumn	Dữ liệu nhập phải là duy nhất trong bảng CSDL, nếu tên colum không được sử dụng thì trường name sẽ được dùng.	'email' =>'unique:users,email_address'
unique:()->ignore()	Dữ liệu nhập phải là duy nhất và bỏ qua trường nào đó.	'email' =>Rule::unique('users')->ignore($user->id)
unique:()->where()	Dữ liệu nhập phải là duy nhất và thỏa mãn phương thức where.	'email' =>Rule::unique('users')->where(function ($query) {

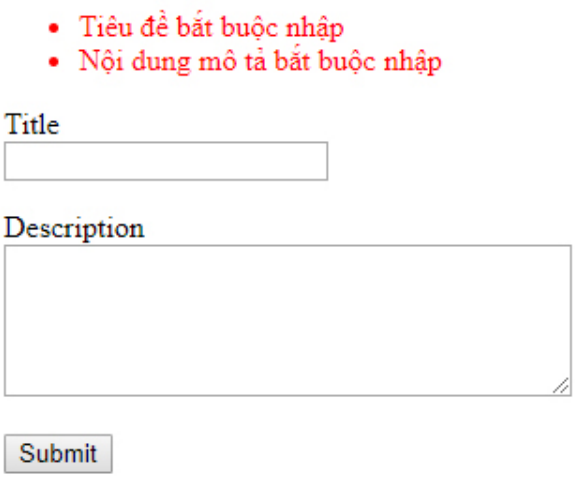
		return $query->where('account_id', 1);})
url	Dữ liệu nhập phải là dạng url.	'path' =>'url'
uuid	Dữ liệu nhập phải mang giá trị RFC 4122 (phiên bản 1,3,4 hoặc 5) định danh duy nhất trên toàn cầu (UUID).	'uuid' =>'uuid'

Tùy chỉnh nội dung hiển thị lỗi

- Trong phần này, chúng ta sẽ tùy chỉnh câu hiển thị lỗi như mong muốn, ví dụ hiển thị bằng tiếng Việt chẳng hạn.
- Ta thêm vào controller *PostController.php* một biến *messages* với nội dung như sau:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PostController extends Controller
{
    public function showform() {
        return view('admin/create');
    }
    public function validationform(Request $request) {
        echo "<pre>";
            print_r($request->all());
        echo "</pre>";

        $messages = [
            'title.required' => 'Tiêu đề bắt buộc nhập',
            'title.max' => 'Tiêu đề không được vượt quá 50 ký tự',
            'description.required' => 'Nội dung mô tả bắt buộc nhập'
        ];

        $this->validate($request,[
            'title'=>'required|max:255',
            'description'=>'required'
        ], $messages);
    }
}
```

- Khi này, nếu nhập title và description không đúng điều kiện trong Controller, thì sẽ báo lỗi, ví dụ ta bỏ trống 2 field này và click button `Submit`:



- Ta thấy nội dung lỗi đã được thay đổi theo tùy chỉnh của biến `$messages`.

Tùy chỉnh vị trí hiển thị lỗi

- Giả sử chúng ta muốn hiển thị nội dung báo lỗi ngay bên dưới trường cần nhập thì làm như thế nào? Laravel cung cấp một cách tùy chỉnh khá hay giúp điều khiển vị trí xuất hiện lỗi khá thuận tiện, đó là cách sử dụng hàm có sẵn `$errors->first('name')`.

- Để thực hiện việc này, chúng ta cần viết lại controller *PostController.php* như:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PostController extends Controller
{
    public function showform() {
        return view('admin/create');
    }
    public function validationform(Request $request) {
        echo "<pre>";
            print_r($request->all());
        echo "</pre>";

        $messages = [
            'title.required' => 'Tiêu đề bắt buộc nhập',
            'title.max' => 'Tiêu đề không được vượt quá 50 ký tự',
```

```
            'description.required' => 'Nội dung mô tả bắt buộc nhập'
        ];

        $this->validate($request,[
            'title'=>'required|max:255',
            'description'=>'required'
        ], $messages);

        $errors = $validate->errors();
    }
}
```

VIẾT LẠI THÔNG BÁO LỖI TẠI VIEW: create.blade.php

```
<!doctype html>
<html>
<head>
    <title>Login Form</title>
    <style type="text/css">
        .error-message { color: red; }
    </style>
</head>

<body>
    <form method="post" action="/admin/create">
        @csrf
        <p>Title<br>
            <input type="text" name="title" value=""><br>
            <span class="error-message">{{ $errors->first('title') }}</span></p>

        <p>Description<br>
            <textarea rows="5" cols="40" name="description"></textarea><br>
            <span class="error-message">{{ $errors->first('description') }}</span></p>

        <p><button type="submit">Submit</button></p>
    </form>
</body>
</html>
```

- Khi này, nếu nhập title và description không đúng điều kiện trong Controller, thì nội dung lỗi sẽ xuất hiện như sau:

Title

Tiêu đề bắt buộc nhập

Description

Nội dung mô tả bắt buộc nhập

Submit