

LAB 7 MÔN LẬP TRÌNH PHP3: VALIDATION & SEND MAIL

MỤC TIÊU:

Kết thúc bài thực hành này bạn có khả năng

- ✓ Triển khai validate các loại dữ liệu đầu vào
- ✓ Send mail qua API Server

NỘI DUNG

Chuẩn bị:

1. Cài Laravel vào folder **la7**
2. Tạo database có tên **la7** và file .env cấu hình kết nối đến database vừa tạo.

Bài 1: Kiểm tra dữ liệu với hàm validate() của đối tượng Request

1. Tạo controller, định nghĩa action và nạp view

- Tạo controller có tên HsController, trong đó định nghĩa 2 action:

```
public function hs(){
    return view("nhaphs");
}
function hs_store(Request $request){
    echo "Code xử lý lưu thông tin học sinh";
}
```

2. Tạo route

Tạo route dẫn vào 2 action vừa định nghĩa

```
Route::get("hs",[App\Http\Controllers\HsController::class,'hs']);
Route::post("hs",[App\Http\Controllers\HsController::class,'hs_store'])-
>name('hs_store');
```

3. Tạo view

- Tạo file views/nhaphs.blade.php

```
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/b
ootstrap.min.css">
<div class="col-6 m-auto">
<form method="post" class="p-3 border border-primary">
    <h3 class="h4 bg-info p-2 mx-n3 mt-n3 text-white">NHẬP THÔNG TIN HỌC SINH</h3>
    <div class="form-group row">
        <label class="col-3">Họ tên học sinh</label>
        <div class="col-9">
            <input value="{{old('hoten')}}" type="text" class="form-control" name="hoten">
        </div>
    </div>
</div>
```

```
<div class="form-group row">
  <label class="col-3">Tuổi</label>
  <div class="col-9">
    <input value="{{old('tuoi')}}" type="text" class="form-control" name="tuoi">
  </div>
</div>
<div class="form-group row">
  <label class="col-3">Ngày sinh</label>
  <div class="col-9">
    <input value="{{old('ngaysinh')}}" type="text" class="form-
control" name="ngaysinh">
  </div>
</div>
<div class="form-group row">
  <div class="col-12">
    <button type="submit" class="btn btn-primary w-25">Lưu thông tin</button>
  </div>
</div>
</form>
</div>
```

- Định nghĩa action cho form :

```
action="{{route('hs_store')}}"
```

- Thêm field csrf cho form: Trong form thêm code: **@csrf**
- Test <http://localhost:8000/hs> , submit thử sẽ phải thấy text **Code xử lý lưu thông tin học sinh**

NHẬP THÔNG TIN HỌC SINH

Họ tên học sinh

Tuổi

Ngày sinh

Lưu thông tin

4. Hiện các thông báo lỗi trong form

- Code trước tag form trong view nhaphs:

```
@if ($errors->any())
  <div class="alert alert-danger">
    <ul>
      @foreach ($errors->all() as $error)
        <li>{{ $error }}</li>
      @endforeach
    </ul>
  </div>
@endif
```

5. Code kiểm tra trong hàm validate() của đối tượng request

Yêu cầu: Phải kiểm tra dữ liệu nhập trong form theo các quy tắc:

- Họ tên: phải nhập, độ dài từ 3 đến 20 ký tự
- Tuổi: bắt buộc nhập, là số nguyên, giá trị phải nhập từ 16 đến 100
- Ngày sinh: phải nhập kiểu theo định dạng ngày

Trong action **hs_store** ở trên, code:

```
public function hs_store(Request $request) {  
    $request->validate([  
        'hoten' => ['required', 'min:3', 'max:20'],  
        'tuoi' => 'required|integer|min:16|max:100',  
        'ngaysinh' => ['required', 'date'],  
    ]  
    );  
    echo "Code xử lý";  
}
```

Test: <http://localhost:8000/hs> . Submit phải thấy báo lỗi

- The hoten field is required.
- The tuoi field is required.
- The ngaysinh field is required.

NHẬP THÔNG TIN HỌC SINH

Họ tên học sinh

Tuổi

Ngày sinh

Lưu thông tin

Bài 2: Validate dùng request class

1. Tạo controller, action và view

- Tạo controller có tên SvController và 2 action

```
public function sv(){
    return view("nhapsv");
}
function sv_store(RuleNhapSV $request){
    echo "Code xử lý lưu thông tin sinh viên";
}
```

- Thêm lệnh use request class ở đầu controller

```
use App\Http\Requests\RuleNhapSV;
```

2. Tạo route

Tạo route dẫn vào 2 action vừa định nghĩa

```
Route::get("sv",[App\Http\Controllers\SvController::class,'sv']);
Route::post("sv",[App\Http\Controllers\SvController::class,'sv_store']) -
>name('sv_store');
```

3. Tạo view

- Tạo file views/nhapsv.blade.php như sau:

```
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/b
ootstrap.min.css">
<div class="col-6 m-auto">
<form method="post" class="p-3 border border-primary">
    <div class="form-group row">
        <label class="col-3">Mã SV</label>
        <div class="col-9">
            <input value="{{old('masv')}}" type="text" class="form-control" name="masv">
        </div>
    </div>
    <div class="form-group row">
        <label class="col-3">Họ tên</label>
        <div class="col-9">
            <input value="{{old('hoten')}}" type="text" class="form-control" name="hoten">
        </div>
    </div>
    <div class="form-group row">
        <label class="col-3">Tuổi</label>
        <div class="col-9">
            <input value="{{old('tuoi')}}" type="text" class="form-control" name="tuoi">
        </div>
    </div>
    <div class="form-group row">
        <label class="col-3">Ngày sinh</label>
        <div class="col-9">
            <input value="{{old('ngaysinh')}}" type="text" class="form-
control" name="ngaysinh">
        </div>
    </div>
```

```

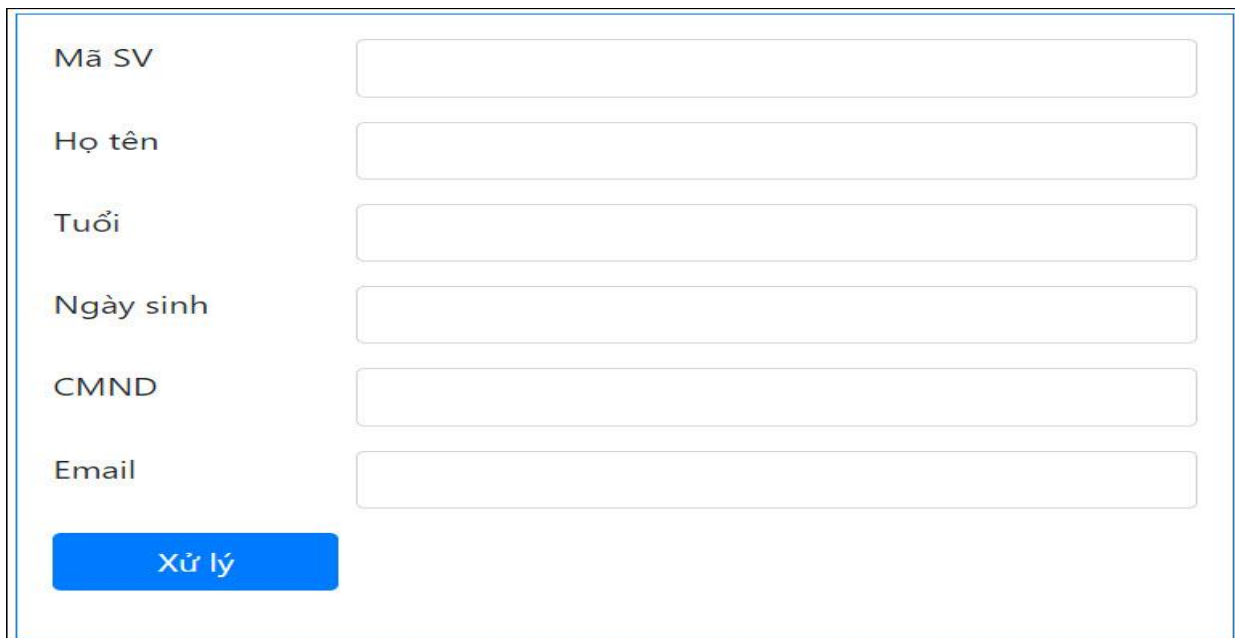
</div>
<div class="form-group row">
  <label class="col-3">CMND</label>
  <div class="col-9">
    <input value="{{old('cmd')}}" type="text" class="form-control" name="cmd">
  </div>
</div>
<div class="form-group row">
  <label class="col-3">Email</label>
  <div class="col-9">
    <input value="{{old('email')}}" type="text" class="form-control" name="email">
  </div>
</div>
<div class="form-group row">
  <div class="col-12">
    <button type="submit" class="btn btn-primary w-25">Xử lý</button>
  </div>
</div>
</form>
</div>

```

- Trong thuộc tính action của form, thêm code:

```
action="{{route('sv_store')}}"
```

- Thêm field csrf trong form: @csrf
- Xem thử: <http://localhost:8000/sv>



4. Hiện các thông báo lỗi trong form

Code trước tag form trong view nhapsv:

```

@if ($errors->any())
  <div class="alert alert-danger">

```

```

        <ul>
            @foreach ($errors->all() as $error)
                <li>{{ $error }}</li>
            @endforeach
        </ul>
    </div>
@endif
    
```

5. Tạo request class và code validate

- Tạo request tên RuleNhapSV như sau:

```
php artisan make:request RuleNhapSV
```

- Định nghĩa các rule cần kiểm tra:
Mở file app/Http/Requests/RuleNhapSV.php, viết các rules trong hàm rules

```

public function rules()
{
    return [
        'masv' => ['required', 'regex:/^PS\d{5}/'],
        'hoten' => ['required', 'min:3', 'max:20'],
        'tuoi' => 'required|integer|min:16|max:100',
        'ngaysinh' => ['required', 'regex:/\d{1,2}\/\d{1,2}\/\d{4}/'],
        'cmnd' => 'digits_between:9,10',
        'email' => 'email|ends_with:@fpt.edu.vn'
    ];
}
    
```

- Cho phép request class hoạt động: Sửa **false** trong hàm authorize() thành **true**
- Test

6. Hiện lỗi riêng cho từng field

- Code ngay sau tag input masv:

```

@error('masv')
    <span class="badge badge-danger">{{ $message }}</span>
@enderror
    
```

- Hiện lỗi riêng cho field hoten, field tuoi, cmnd, email: Thực hiện tương tự

7. Định nghĩa các chuỗi báo lỗi thân thiện

- Trong request class, định nghĩa hàm messages() như sau:

```
function messages(){
    return [
        'masv.required' => 'Mã SV chưa nhập',
        'masv.regex' => 'Mã SV không đúng định dạng',
        'hoten.required' => 'Họ tên sao không nhập ta',
        'hoten.min' => 'Họ tên sao ngắn quá vậy',
        'hoten.max' => 'Họ tên quá dài bồ ơi'
    ];
}
```

- Thêm các message cho ngaysinh, cmnd, tuoi, email: sinh viên tự thực hiện

8. Khai báo tên thân thiện cho các field

- Trong request class , định nghĩa hàm attributes() như sau:

```
function attributes(){
    return [
        'masv'=>'Mã sinh viên',
        'hoten'=>'Họ tên'
    ];
}
```

- Định nghĩa tên thân thiện cho cho các field ngaysinh, cmnd, tuoi, email
- Test:

Bài 3 Gửi mail với mailgun

1. Đăng ký mailgun ,

Đăng ký tài khoản trên <https://mailgun.com> . Xong vào Sending → Domains → API key để lấy Private API key

2. Khai báo cấu hình mailgun

Mở file .env khai báo 2 biến MAILGUN_DOMAIN và MAILGUN_SECRET

3. Cài gói Symfony Mailer

composer require symfony/mailgun-mailer symfony/http-client

4. Tạo view chứa nội dung mail :

views/guimail.blade.php

```
<b>Chào bạn</b>
<p>
    <i>Khỏe không bạn? Chúc an lành!
    Chúc thành công</i>
</p>
```

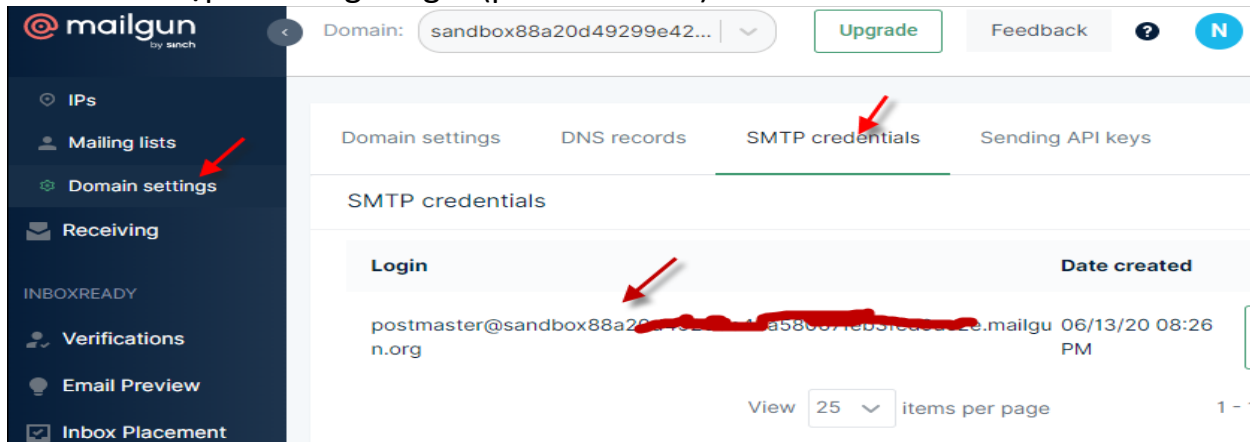
5. Tạo mailable class

```
php artisan make:mail GuiEmail
```

Mở file app/Mail/GuiMail.php, code trong hàm built để khai báo from, to, subject

```
public function build() {
    return $this
        ->from("diachin@guoigui.com", "Tên người gửi")
        ->to("diachi@nguoinhan.com")
        ->subject("Tiêu đề thư")
        ->attach(public_path() . "/hinh1.jpg") // đính kèm file
        ->view('guimail'); // nạp view nội dung mail
}
```

- from: nhập email người gửi (postmaster....)



- to: nhập địa chỉ người nhận (tùy ý)
- subject: nhập tiêu đề thư (tùy ý)
- attach: file đính kèm (tùy ý)

6. Thực hiện gửi mail:

routes/web.php

```
use App\Mail\GuiEmail;
Route::get("/guimail", function(){
    Mail::mailer('mailgun')->send(new GuiEmail());
});
```

*** Yêu cầu nộp bài:

SV nén file (hoặc share thư mục google drive) bao gồm các yêu cầu đã thực hiện trên, nộp LMS đúng thời gian quy định của giảng viên. KHÔNG NỘP BÀI COI NHƯ KHÔNG CÓ ĐIỂM.

--- Hết ---