**Hướng dẫn sử dụng Laravel 5.1 cơ bản**

1. **Cài đặt laravel:**

Trước hết cài đặt composer theo link:

<https://getcomposer.org/download/>

Sau đó tải laravel + tạo project theo hướng dẫn theo link bên dưới:

composer create-project laravel/laravel --prefer-dist

<http://laravel.com/docs/5.1#installation>

1. **Cài đặt debugBar**

B1: Cài đặt:

Mở cmd và cd vào thư mục laravel

composer require barryvdh/laravel-debugbar

B2: Khai báo trong config/app.php:

Providers:

Barryvdh\Debugbar\ServiceProvider::class,

Aliases:

'Debugbar' => Barryvdh\Debugbar\Facade::class,

B3: Công khai cấu hình debugBar:

php artisan vendor:publish

OK Mở url: <http://localhost/laravel> để xem kết quả

1. **Tạo controller**

php artisan make:controller WorkerController

Tạo route link url tới controller

Mở file app/http/route.php và thêm dòng sau:

Route::get('worker', [

'as' => 'worker-list', 'uses' => 'WorkerController@index'

]);

Mở url: <http://localhost/laravel/public/worker> để xem kết quả

1. **Tạo file migrate table và model liên kết với table**

Mặc định laravel quy định sử dụng tên model là số ít và tên bảng là số nhiều.

Ví dụ model: User.php và table là users

Khóa chính mặc định sẽ là id tự tăng

Khóa ngoại bao gồm tên bảng liên kết và id ví dụ: user\_id

Để tạo model và file migrate gõ lệnh sau:

php artisan make:model Worker –-migration

Sau khi gõ lệnh thì 1 file migrate sẽ được tạo trong thư mục database/migrations/ và file model Worker.php trong thư mục app

Mở file \*\*\*\_create\_workers\_table và sửa lại đoạn code cho phù hợp với thiết kế table:

Schema::create('workers', function (Blueprint $table) {

$table->increments('id');

$table->string('name');

$table->date('birth\_date');

$table->enum('gender', ['male', 'female']);

$table->text('description');

$table->timestamps();

});

Thực thi migrate để tạo bảng dữ liệu:

Trước khi migrate tạo database laravel-test sau đấy mở file .svn ở thư mục gốc và sửa kết nối đến server:

DB\_HOST=localhost

DB\_DATABASE=laravel-test

DB\_USERNAME=root

DB\_PASSWORD=admin

Thực hiện lệnh migrate:

php artisan migrate

Bảng workers sẽ được tạo trên database

Tạo file WorkersTableSeeder để tạo dữ liệu test cho bảng wokers

php artisan make:seeder WorkersTableSeeder

Để tiện cho việc tạo dữ liệu test nhanh cài thêm thư viện Faker

composer require fzaninotto/faker --dev

Mở file database/seeds/WorkersTableSeeder thêm đoạn code sau vào hàm run

public function run()

{

for ($i = 1; $i <= 50; $i++) {

$faker = \Faker\Factory::create('vi\_VN');

$worker = new \App\Worker();

$worker->name = $faker->name;

$worker->birth\_date = $faker->date();

$worker->gender = $faker->randomElement(['male', 'female']);

$worker->description = $faker->text;

$worker->save();

}

}

Mở file database/seeds/DatabaseSeeder

Thêm dòng sau vào hàm run:

$this->call(WorkersTableSeeder::class);

Thực hiện tạo dữ liệu test:

php artisan db:seed

1. **Sử dụng model orm truy vấn tabase**

Mở file app/http/controllers/WorkerController.php

Để truy vấn dữ liệu:

$workers = \App\Worker::all();

foreach ($workers as $worker) {

echo($worker->name) . '<br>';

}

dd('ok');

Mở url: <http://localhost/laravel/public/worker> để xem kết quả

Tham khảo cách truy vấn sử dụng Orm của laravel: <http://laravel.com/docs/5.1/eloquent>

Để tạo bản ghi mới để lấy dữ liệu ví dụ:

$worker = new \App\Worker();

$worker->name = 'Mạnh Phạm';

$worker->birth\_date = '1991-01-01';

$worker->gender = 'male';

$worker->description = 'no description';

$worker->save();

dd('ok');

Để sửa một bản ghi id=50

$worker = \App\Worker::find(51);

$worker->name = 'Mạnh Phạm';

$worker->birth\_date = '1998-01-01';

$worker->gender = 'female';

$worker->description = 'test ti hoy;

$worker->save();

dd('ok');

1. **Tạo view và đưa dữ liệu ra view**

Trên action index sau khi truy vấn dữ liệu thì trả mảng kết quả ra view

$workers = \App\Worker::limit(5)->get();

return view('worker.list')->with('workers', $workers);

Tạo file list.blade.php trong thư mục resources/worker/

@foreach($workers as $worker)

<div>{{ $worker-> birth\_date}} | {{ $worker->name }}</div>

@endforeach

Mở url: <http://localhost/laravel/public/worker> để xem kết quả

Laravel hỗ trợ phân trang rất tốt

Ở action index thay

$workers = \App\Worker::limit(5)->get();

Bằng

$workers = \App\Worker::paginate(5);

Sửa lại file view

@foreach($workers as $worker)

<div>{{ $worker->birth\_date }} | {{ $worker->name }}</div>

@endforeach

<div>{!! $workers !!}</div>

Mở url: <http://localhost/laravel/public/worker> để xem kết quả

Tham khảo thêm tài liệu về blade tại: <http://laravel.com/docs/5.1/blade>

1. **Tạo form và post dữ liệu**

Cài đặt form: mở cmd gõ lệnh:

composer require laravelcollective/html 5.1.\*

Mở file config/app.php bổ sung thêm:

'providers' => [

// ...

Collective\Html\HtmlServiceProvider::class,

// ...

],

…

'aliases' => [

// ...

'Form' => Collective\Html\FormFacade::class,

'Html' => Collective\Html\HtmlFacade::class,

// ...

],

Mở file app/http/routes.php thêm vào:

Route::get('worker/new', [

'as' => 'worker-new', 'uses' => 'WorkerController@create'

]);

Route::post('worker/add', [

'as' => 'worker-store', 'uses' => 'WorkerController@store'

]);

Mở file app/http/controllers/WorkerController.php thêm gọi view ở hàm create

public function create()

{

return view('worker.add');

}

Tạo file resources/views/worker/add.blade.php

{!! Form::open(['route' => 'worker-store']) !!}

<div class="row">

{!! Form::label('Name:') !!}

{!! Form::text('name') !!}

</div>

<div class="row">

{!! Form::label('Gender:') !!}

{!! Form::radio('gender', 'male', false) !!} Male

{!! Form::radio('gender', 'female', false) !!} Female

</div>

<div class="row">

{!! Form::label('Birth date:') !!}

{!! Form::date('birth\_date') !!}

</div>

<div class="row">

{!! Form::label('Description:') !!}

{!! Form::textarea('description') !!}

</div>

<hr>

<div class="row">

{!! Form::submit('Save') !!}

{!! Html::linkRoute('worker-list', 'Cancel') !!}

</div>

{!! Form::close() !!}

Tham khảo tài liệu về form tại: <http://laravelcollective.com/docs/5.1/html>

1. **Validate dữ liệu và lưu database**

Mở file WorkerController và sửa hàm store để lưu dữ liệu post vào db

public function store(Request $request)

{

$worker = new \App\Worker();

$worker->name = $request->get('name');

$worker->birth\_date = $request->get('birth\_date');

$worker->gender = $request->get('gender');

$worker->description = $request->get('description');

$worker->save();

return redirect()->route('worker-list');

}

Submit form new worker và xem kết quả

Validate form input:

Mở file model app/Worker.php và thêm thuộc tính static

public static $rules = [

'name' => 'required|max:50',

'gender' => 'required|in:male,female',

'birth\_date' => 'required|date\_format:Y-m-d',

];

Sửa lại hàm store ở WorkerController

public function store(Request $request)

{

/\*\*

\* if validate ok create new worker

\* if invalid data back to new worker form

\*/

$this->validate($request, \App\Worker::$rules);

$worker = new \App\Worker();

$worker->name = $request->get('name');

$worker->birth\_date = $request->get('birth\_date');

$worker->gender = $request->get('gender');

$worker->description = $request->get('description');

$worker->save();

\Session::flash('flash-message', 'Worker has been created successful!');

return redirect()->route('worker-list');

}

Thêm code html hiển thị lỗi validate trên view resources/views/worker/add.blade.php

@if (count($errors) > 0)

<div class="alert alert-danger">

<ul>

@foreach ($errors->all() as $error)

<li>{{ $error }}</li>

@endforeach

</ul>

</div>

@endif

Test các giá trị input, tham khảo validation tại: <http://laravel.com/docs/5.1/validation>

Thêm flash message khi thực hiện lưu thành công

Sau khi lưu thành công trang web tự động redirect sang trang list vậy muốn hiển thị message báo đã lưu thành công ở trang này thì làm thế nào?

Sau khi save thành công lưu message vào flash-session:

$request->session()->flash('status', 'Task was successful!');

Mở view resources/views/worker/list.blade.php và them vào html hiển thị message:

@if(Session::has('flash-message'))

<div class="alert alert-success">{{ Session::get('flash-message') }}</div>

@endif

1. **Sử dụng quan hệ để truy vấn sql trên laravel Orm**
2. Quan hệ 1-n

Ví dụ mỗi worker có nhiều số điện thoại, 1 số chính và nhiều số phụ

Tạo thêm bảng phones và model Phone

php artisan make:model Phone --migration

Sửa file migrate

Schema::create('phones', function (Blueprint $table) {

$table->increments('id');

$table->integer('worker\_id')->unsigned();

$table->foreign('worker\_id')->references('id')->on('workers');

$table->string('number');

$table->boolean('main');

$table->timestamps();

});

Chạy migrate

php artisan migrate

Khai báo quan hệ 1-n trên model Worker

public function phones()

{

return $this->hasMany(Phone::class);

}

Khai báo quan hệ n-1 trên model Phone

public function worker()

{

return $this->belongsTo(Worker::class);

}

Sau khi khai báo thì có thể sử dụng hàm quan hệ như 1 thuộc tính của model

Ví dụ ở list worker muốn hiển thị thêm phone

@foreach($workers as $worker)

<div>

{{ $worker->birth\_date }} |

{{ $worker->name }} |

@if($worker->phones->count())

{{ $worker->phones->implode('number', ', ') }}

@else

No phone

@endif

</div>

@endforeach

Fake dữ liệu để test

php artisan make:seeder PhonesTableSeeder

Tạo dữ liệu random phone cho từng worker

public function run()

{

$workers = \App\Worker::all();

foreach ($workers as $worker) {

$p = rand(0, 4);

for ($i = 1; $i <= $p; ++$i) {

$phone = new \App\Phone();

$faker = \Faker\Factory::create();

$phone->number = $faker->phoneNumber;

$phone->main = $faker->randomElement([0, 1]);

$worker->phones()->save($phone);

}

}

}

Chạy seed và xem kết quả

php artisan db:seed --class=PhonesTableSeeder

chạy thử đã

Tối ưu query bảng liên kết()

Sử dụng cách trên thì số lượng query lớn => chậm

Sửa lại query ở controller

$workers = \App\Worker::with('phones')->paginate(5);

Số lượng query sẽ giảm đi

Sử dụng điều kiện đối với bảng phụ, ví dụ chỉ hiển thị số phone có main=1

$workers = \App\Worker::with([

'phones' => function ($q) {

$q->where('main', 1);

}

])->paginate(5);

Ví dụ khác: hiển thị số điện thoại main = 1 và nếu không có số nào là main thì hiện thị số đầu tiên (tự làm)

Sử dụng bảng phụ trong điều kiện query, ví dụ chỉ hiện danh sách những người có số phone

$workers = \App\Worker::with([

'phones' => function ($q) {

$q->where('main', 1);

}

])

->has('phones')

->paginate(5);

Bài tập: tìm những người có số điện thoại của Viettel? Tham khảo whereHas

<http://laravel.com/docs/5.1/eloquent-relationships>

1. Quan hệ nhiều – nhiều

Ví dụ: quan hệ workers và skills là quan hệ nhiều nhiều ta sẽ có bảng workers, skills và bảng liên kết worker\_skill

Tạo bảng skills và model Skill

php artisan make:model Skill –m

Sửa file migrate

Schema::create('skills', function (Blueprint $table) {

$table->increments('id');

$table->string('name', 50);

$table->text('description');

$table->enum('status', ['enable', 'disabled']);

$table->timestamps();

});

Tạo bảng liên kết: theo mặc định của laravel thì bảng liên kết phải đặt tên theo thứ tự a-b-c vì vậy bảng liên kết sẽ có tên skill\_worker, trên bảng này phải khai báo khóa ngoại liên kết tới 2 bảng skills và worker.

Để tiện cho việc tạo bảng liên kết sử dụng thư viện generate. Cài đặt generate :

composer require laracasts/generators –-dev

Khai báo provider tại file app\Providers\AppServiceProvider.php

public function register()

{

…………………

if ($this->app->environment() == 'local') {

$this->app->register('Laracasts\Generators\GeneratorsServiceProvider');

}

}

Chạy command để tạo file migration

php artisan make:migration:pivot workers skills

File migration \database\migrations\\*\*\*\_create\_skill\_worker\_pivot\_table.php sẽ được tạo đúng format

Chạy command để tạo table trên db

php artisan migrate

Khai báo quan hệ m-n tại model Worker

public function Skills()

{

return $this-> belongsToMany(Skill::class);

}

Khai báo quan hệ m-n tại model Skill

public function workers()

{

return $this->belongsToMany(Worker::class);

}

Tạo dữ liệu fake cho bảng skill

php artisan make:seeder SkillsTableSeeder

Sửa file seeder

public function run()

{

for($i=1; $i<20; $i++) {

$skill = new \App\Skill();

$faker = \Faker\Factory::create();

$skill->name = $faker->word;

$skill->status = 'enable';

$skill->save();

}

}

Cmd:

php artisan db:seed --class=SkillsTableSeeder

Tạo dữ liệu fake cho bảng skill\_worker

php artisan make:seeder SkillWorkerTableSeeder

….

public function run()

{

$workers = \App\Worker::all();

$skills = \App\Skill::all();

foreach ($workers as $worker) {

$randSkills = $skills->shuffle()->take(rand(0, 5))->all();

$worker->skills()->saveMany($randSkills);

}

}

Cmd:

php artisan db:seed --class=SkillWorkerTableSeeder

Test truy vấn dữ liệu quan hệ m-n

Tạo route

Route::get('worker/{id}', [

'as' => 'worker-view', 'uses' => 'WorkerController@show'

]);

Sửa file show() của WorkerController

public function show($id)

{

$worker = \App\Worker::find($id);

return view('worker.show', compact(‘worker’));

}

Tạo file view worker/show.blade.php

<table>

<tr>

<td>Name</td>

<td>{{ $worker->name }}</td>

</tr>

<tr>

<td>Gender</td>

<td>{{ $worker->gender }}</td>

</tr>

<tr>

<td>Birth date</td>

<td>{{ $worker->birth\_date }}</td>

</tr>

<tr>

<td>Phones</td>

<td>{{ $worker->phones->implode('number', ' | ') }}</td>

</tr>

<tr>

<td>Skills</td>

<td>{{ $worker->skills->implode('name', ' | ') }}</td>

</tr>

</table>

{!! Html::linkRoute('worker-list', 'Back') !!}

Mở url để test: <http://localhost/laravel/public/worker/1>

**\*\* Sử dụng hàm sync trong quan hệ m-n**

Thêm 2 route cho form skills và hàm update skills

Route::get('worker/{id}/skills', [

'as' => 'worker-skills', 'uses' => 'WorkerController@skills'

]);

Route::post('worker/{id}/update-skills', [

'as' => 'worker-update-skills', 'uses' => 'WorkerController@skillsUpdate'

]);

Thêm hàm skills vào WorkerController

public function skills($id)

{

$worker = \App\Worker::find($id);

$skills = \App\Skill::all();

return view('worker.skills')

->with('skills', $skills)

->with('worker', $worker);

}

Thêm file view: worker/skills.blade.php

{!! Form::open(['route' => ['worker-update-skills', $worker->id]]) !!}

<table>

<tr>

<td>Name</td>

<td>{{ $worker->name }}</td>

</tr>

<tr>

<td>Skills</td>

<td>

<ul>

@foreach($skills as $skill)

<?php $checked = $worker->skills->where('id', $skill->id)->count() > 0; ?>

<li> {!! Form::checkbox('skills[]', $skill->id, $checked) !!} {{ $skill->name }} </li>

@endforeach

</ul>

</td>

</tr>

</table>

{!! Form::submit('Save') !!}

{!! Form::close() !!}

Thêm link vào file show.blade.php

{!! Html::linkRoute('worker-skills', 'Skills', ['id' => $worker->id]) !!}

Mở url xem kết quả: <http://localhost/laravel/public/worker/1/skills>

Thêm hàm skillsUpdate() vào file WorkerController

public function SkillsUpdate($id, Request $request)

{

$worker = \App\Worker::find($id);

$worker->skills()->sync($request->get('skills'));

return redirect()->route('worker-view', $worker->id);

}

Mở form skills để test kết quả

Thêm validate, trường hợp nếu skills post lên nhưng không có trong db sẽ dẫn đến lỗi hệ thống vì vậy cần phải kiểm tra tất cả các skills post lên phải tồn tại

public function SkillsUpdate($id, Request $request)

{

$this->validate($request, ['skills' => 'array|exists:skills,id']);

$worker = \App\Worker::find($id);

$worker->skills()->sync($request->get('skills'));

return redirect()->route('worker-view', $worker->id);

}

Thêm html hiển thị message

@if (count($errors) > 0)

<div class="alert alert-danger">

<ul>

@foreach ($errors->all() as $error)

<li>{{ $error }}</li>

@endforeach

</ul>

</div>

@endif

Bài tập: Tạo link liên kết giữa các trang đã tạo

1. **Tạo layout và sử dụng extends view**

Tạo file resources/views/layouts/default.blade.php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- The above 3 meta tags \*must\* come first in the head; any other head content must come \*after\* these tags -->

<meta name="description" content="">

<meta name="author" content="">

<link rel="icon" href="favicon.ico">

<title>Starter Template</title>

<!-- Bootstrap core CSS -->

<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap.min.css" rel="stylesheet">

<!-- Custom styles for this template -->

<!-- <link href="starter-template.css" rel="stylesheet"> -->

<!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->

<!--[if lt IE 9]>

<script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"></script>

<script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>

<![endif]-->

</head>

<body>

<nav class="navbar navbar-default">

<div class="container">

<div class="navbar-header">

<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand" href="#">Project name</a>

</div>

<div id="navbar" class="collapse navbar-collapse">

<ul class="nav navbar-nav">

<li class="active"><a href="#">Home</a></li>

<li><a href="#about">About</a></li>

<li><a href="#contact">Contact</a></li>

</ul>

</div><!--/.nav-collapse -->

</div>

</nav>

<div class="container">

@yield('header')

@yield('content')

</div><!-- /.container -->

<!-- Bootstrap core JavaScript

================================================== -->

<!-- Placed at the end of the document so the pages load faster -->

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/js/bootstrap.min.js"></script>

<!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

<!-- <script src="../../assets/js/ie10-viewport-bug-workaround.js"></script> -->

</body>

</html>

Tiếp theo sửa lại các file views

Ví dụ file resources/views/worker/list.blade.php

@extends('layouts/default)

@section('header')

<div class="page-header clearfix">

<h1>

<i class="glyphicon glyphicon-align-justify"></i> Workers

<a class="btn btn-success pull-right" href="{{ route('worker-new') }}"><i class="glyphicon glyphicon-plus"></i> New Worker</a>

</h1>

@if(Session::has('flash-message'))

<div class="alert alert-success">{{ Session::get('flash-message') }}</div>

@endif

</div>

@endsection

@section('content')

<div class="row">

<div class="col-md-12">

@if($workers->count())

<table class="table table-condensed table-striped">

<thead>

<tr>

<th>Id</th>

<th>Name</th>

<th>Birth date</th>

<th>Gender</th>

<th>Phones</th>

<th class="text-right">Options</th>

</tr>

</thead>

<tbody>

@foreach($workers as $worker)

<tr>

<td>{{$worker->id}}</td>

<td>{{$worker->name}}</td>

<td>{{$worker->birth\_date}}</td>

<td>{{$worker->gender}}</td>

<td>

@if($worker->phones->count())

{{ $worker->phones->implode('number', ', ') }}

@else

No phone

@endif

</td>

<td class="text-right">

<a class="btn btn-xs btn-primary" href="{{ route('workers-show', $worker->id) }}"><i class="glyphicon glyphicon-eye-open"></i> View</a>

<a class="btn btn-xs btn-warning" href="{{ route('workers-edit', $worker->id) }}"><i class="glyphicon glyphicon-edit"></i> Edit</a>

<form action="{{ route('workers-delete', $worker->id) }}" method="POST" style="display: inline;" onsubmit="return confirm('Delete? Are you sure?')">

<input type="hidden" name="\_method" value="DELETE">

<input type="hidden" name="\_token" value="{{ csrf\_token() }}">

<button type="submit" class="btn btn-xs btn-danger"><i class="glyphicon glyphicon-trash"></i> Delete</button>

</form>

</td>

</tr>

@endforeach

</tbody>

</table>

{!! $workers !!}

@else

<h3 class="text-center alert alert-info">Empty!</h3>

@endif

</div>

</div>

@endsection

Bài tập sửa những views còn lại