INTRODUÇÃO AO LARAVEL (PARTE 2)

Douglas Nassif Roma Junior

/douglasjunior

/in/douglasjunior

douglasjunior.me

nassifrroma@gmail.com

Slides: https://git.io/vAd6S

AGENDA

- Controllers
  - MVC
  - Uso básico
  - Integrando com Middlewares
  - Resources
  - Injeção de dependências
  - Request
- Validação

CONTROLLERS

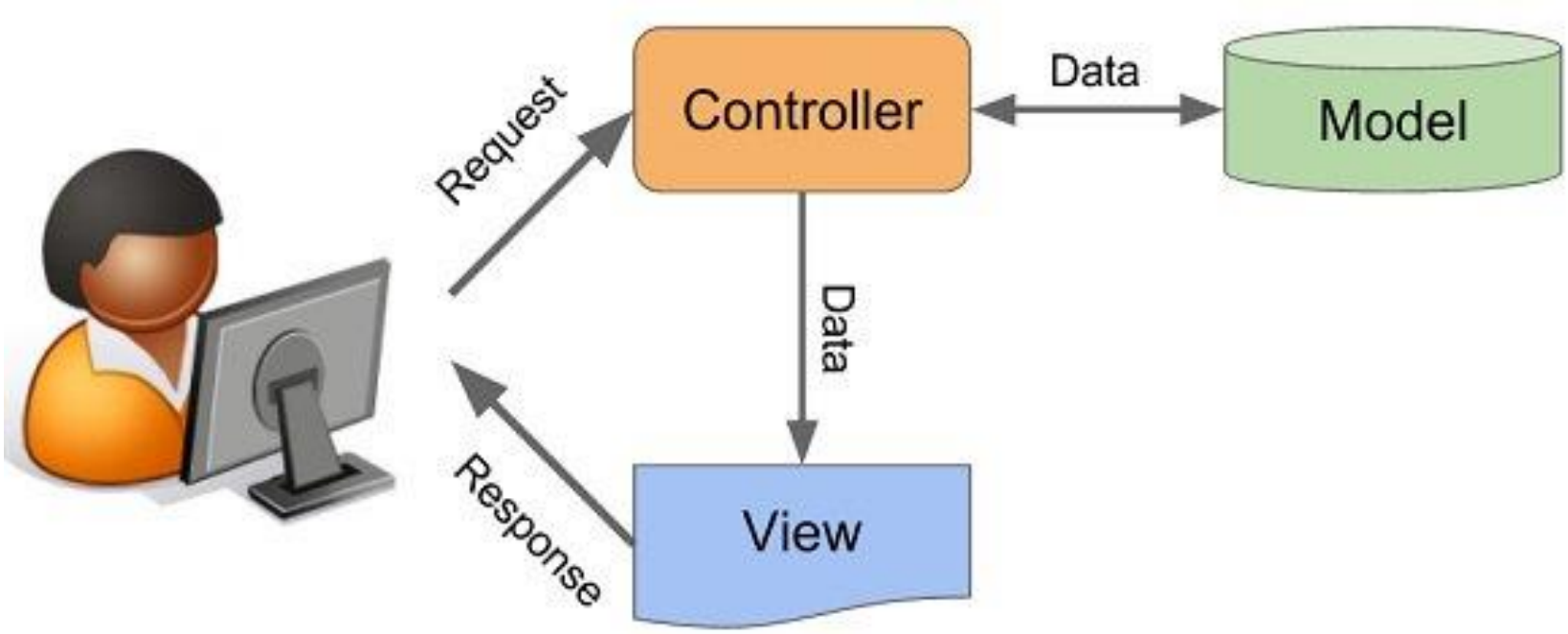
- Em vez de tratar todas as requisições das rotas em uma função de *callback* (Clousure), você pode desejar organizar este comportamento utilizando os **Controllers**.

- **Controllers** podem agrupar requisições relacionadas e tratar a lógica em uma única classe.

- No Laravel, os **Controllers** ficam localizados no diretório `app/Http/Controllers`

CONTROLLERS

- Model View Controller (MVC)



CONTROLLERS

- **Exemplo básico de um Controller:**
  - app/Http/Controllers/BemVindoController.php

```php
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;

class BemVindoController extends Controller
{
  public function saudacao($nome = '')
  {
    return view('bemvindo', ['nome' => $nome]);
  }
}
```

CONTROLLERS

- Definindo uma rota para o Controller:
  - `routes/web.php`

```php
<?php

Route::get('bemvindo/{nome?}', 'BemVindoController@saudacao');
```

CONTROLLERS

- Observe que todo *controller* deve herdar da classe base do **Controller** incluída no Laravel.

- A classe base fornece alguns métodos convenientes, como o método `middleware()`, que pode ser usado para conectar *middleware* às ações do *controller*.

```
public function __construct()
{
    $this->middleware('checar.idade');
}
```

CONTROLLERS

- **Controllers** também podem ser associados aos **Middlewares** através da definição das rotas.

```
Route::get('bemvindo/{nome?}', 'BemVindoController@saudacao’)
    ->middleware('checar.idade');
```

CONTROLLERS

- Para operações básicas de CRUD, você pode querer criar um **Resource Controller**.

- Este tipo de *controller* pode ser criado facilmente utilizando o **Artisan**:

```
php artisan make:controller ProdutoController --resource
```

- **Este comando irá gerar um novo controller em** `app/Http/Controllers/ProdutoController.php`

CONTROLLERS

- **Resource Controllers** podem ser associados à rotas através de uma única linha.

```
Route::resource('produto', 'ProdutoController');
```

- Esta única linha irá criar múltiplas rotas para tratar uma variedade de ações do recurso.

CONTROLLERS

- O controller gerado também irá conter métodos para cada uma das ações.
- Incluindo comentários sobre os verbos HTTP e URIs que eles manipulam.
- Você também pode conferir todas as suas rotas através do comando:

```
php artisan route:list
```

Verb	URI	Action	Route Name
GET	/produto	index	produto.index
GET	/produto/create	create	produto.create
POST	/produto	store	produto.store
GET	/produto/{id}	show	produto.show
GET	/produto/{id}/edit	edit	produto.edit
PUT/PATCH	/produto/{id}	update	produto.update
DELETE	/produto/{id}	destroy	produto.destroy

CONTROLLERS

- Quando declarar rotas que serão consumidas como API, normalmente não precisamos das rotas que exibem templates HTML, como por exemplo `create` ou `edit`.

- Para isso você pode declarar a rota da seguinte forma:

```
Route::apiResource('produto', 'ProdutoController');
```

E para gerar um controller sem os métodos `create` ou `edit`, você pode usar:

```
php artisan make:controller ProdutoController --api
```

CONTROLLERS

- Por padrão o Laravel irá criar os verbos da URI em inglês, como `create` e `edit`. Porém, você pode traduzi-los no arquivo `app/Providers/AppServiceProvider.php`

```php
use Illuminate\Support\Facades\Route;

...

public function boot()
{
  Route::resourceVerbs([
    'create' => 'cadastrar',
    'edit' => 'editar',
  ]);
}
```

CONTROLLERS

- O Laravel também conta com um recurso chamado **Service Container**, responsável por realizar injeção de dependência nos *controllers*.

- Deste modo, você pode declarar as dependências de seu *controller* como parâmetro de entrada no construtor, e o Laravel irá cuidar do resto para você.

CONTROLLERS

- Exemplo de injeção de dependência.

```
class UserController extends Controller {

  protected $taskCtrl;

  public function __construct(TaskController $taskCtrl) {
    $this->taskCtrl = $taskCtrl;
  }

  public function show($id) {
    return view(
       'user.profile’,
       [‘tasks' => $this->taskCtrl->getTasks()]
    );
  }
}
```

CONTROLLERS

- A injeção de dependência também pode ser realizada nos métodos dos *controllers*, como por exemplo:

```
public function saudacao(Request $request, $nome = '')
{
  var_dump($request->headers->get('cookie'));
  ...
}
```

VALIDAÇÃO

- O Laravel possui um sistema de validação baseado em regras. Com ele você pode criar suas próprias regras, ou utilizar uma das regras existentes.

- Por padrão, se você utilizar o método `$request->validate()`, será realizado um redirecionamento automático caso as regras de validação não sejam satisfeitas.

- Quando utilizado como API, é mais interessante gerenciar a resposta de validação manualmente, para isso existe a classe `Illuminate\Support\Facades\Validator`

VALIDAÇÃO

- Exemplo de uso da validação:

```
use Illuminate\Support\Facades\Validator;

...

$validator = Validator::make($request->all(), [
  'titulo' => 'required|max:50’,
  'descricao' => 'required’
], [
  'titulo.required' => 'O título é obrigatório’,
  'titulo.max' => 'O título deve ter no máximo 50 caracteres’,
  'descricao.required' => 'A descrição é obrigatória.',
]);

if ($validator->fails()) {
  return response()->json($validator->errors(), 412);
}
```

VALIDAÇÃO

- Para criar uma regra de validação customizada, basta utilizando o comando:

```
php artisan make:rule CPF
```

- Um arquivo contendo a regra de validação será criado em `app\Rules\CPF.php`

VALIDAÇÃO

- Para utilizar uma validação customizada:

```php
use App\Rules\CPF;

...

$validator = Validator::make($request->all(), [
  'cpf' => ['required', new CPF()],
], [
  'cpf.required' => 'O CPF é obrigatório.',
]);

if ($validator->fails()) {
  return response()->json($validator->errors(), 412);
}
```

DÚVIDAS?

Douglas Nassif Roma Junior

/douglasjunior

/in/douglasjunior

douglasjunior.me

nassifrroma@gmail.com

Slides: https://git.io/vAd6S