



Desenvolvimento Web II

Framework Laravel 5

Controller / Model / Migration / Seeder
(Menu Principal – SETA)

Gil Eduardo de Andrade





Framework Laravel

Configuração Banco de Dados

- No framework Laravel as informações de configuração para conexão com o banco de dados encontram-se em dois arquivos:
 - *config/database.php*;
 - *.env*

(o arquivo *.env* possibilita que variáveis de configuração necessárias para o funcionamento da aplicação sejam carregadas globalmente)





Framework Laravel

Configuração Banco de Dados

(arquivo : "database.conf")

```
'mysql' => [  
    'driver' => 'mysql',  
    'host' => env('DB_HOST', '127.0.0.1'),  
    'port' => env('DB_PORT', '3306'),  
    'database' => env('DB_DATABASE', 'seta30'),  
    'username' => env('DB_USERNAME', 'root'),  
    'password' => env('DB_PASSWORD', 'Gil.Eduardo12'),  
    'unix_socket' => env('DB_SOCKET', ''),  
    'charset' => 'utf8mb4',  
    'collation' => 'utf8mb4_unicode_ci',  
    'prefix' => '',  
    'strict' => true,  
    'engine' => null,  
],
```

(arquivo : "database.conf")

```
DB_CONNECTION=mysql  
DB_HOST=127.0.0.1  
DB_PORT=3306  
DB_DATABASE=seta30  
DB_USERNAME=root  
DB_PASSWORD=Gil.Eduardo12
```





Framework Laravel

Criando Classes de Modelo (php artisan)

(Localização: “/app”)

- As classes de modelo criadas via “php artisan” herdam as funcionalidades de uma super classe chamada ***Model***;
- A superclasse ***Model*** já possui métodos pré-definidos para manipular dados do banco, e trabalha em conjunto com um framework denominado ***Eloquent***;





Framework Laravel

Criando Classes de Modelo (php artisan)

(Localização: “/app”)

- O ***Eloquent*** possui rotinas que possibilitam acessar e modificar os dados do banco sem que haja a necessidade de utilizar *queries* SQL, ou seja, o acesso ao banco de dados torna-se transparente para o desenvolvedor;
- Ele trabalha, por exemplo, de maneira análoga ao Hibernate no Java;





Framework Laravel

Criando Classes de Modelo (php artisan)

(Localização: “/app”)

- O **Eloquent** é um Framework “ORM - Mapeamento Objeto-Relacional”, técnica onde o programador não precisa saber comandos da linguagem SQL, já que é possível utilizar uma interface de programação simplificada disponível no framework e que garante toda a persistência dos dados;





Framework Laravel

Criando Classes de Modelo (php artisan)

(Localização: “/app”)

- Existem, basicamente, duas maneiras de criarmos e vincularmos classes de modelo utilizando o *artisan*:
 - 1) vinculando a classe *model* com a tabela do banco via atributo *\$table*;
 - 2) vinculando a classe *model* com a tabela do banco via *migration*;



Framework Laravel

Criando Classes de Modelo (Atributo *\$table*)

(Considerando a tabela *tb_aluno* já foi criada no MySQL)

```
Terminal
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
[g1l3du4rd0@localhost aula]$ php artisan make:model AlunoModel
Model created successfully.
[g1l3du4rd0@localhost aula]$
```

```
AlunoModel.php
1 <?php
2
3 namespace App;
4
5 use Illuminate\Database\Eloquent\Model;
6
7 class AlunoModel extends Model {
8
9     protected $table = 'alunos';
10 }
```

O atributo *\$table = 'alunos'* especifica que a classe de Modelo criada, "*AlunoModel*", está vinculada a tabela "*alunos*" do banco de dados. Esse atributo é definido pelo desenvolvedor após a criação da classe.



Framework Laravel

Criando Classes de Modelo (*Migration*)

(Além da classe de modelo, um arquivo de migração é criado em */database/migrations*)

```
Terminal
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
[g1l3du4rd0@localhost aula]$ php artisan make:model CursoModel -m
Model created successfully.
Created Migration: 2018_03_19_225200_create_curso_models_table
[g1l3du4rd0@localhost aula]$
```

```
public function up()
{
    Schema::create('curso_models', function (Blueprint $table) {
        $table->increments('id');
        $table->string('nome');
        $table->string('abreviatura');
    });
}
```

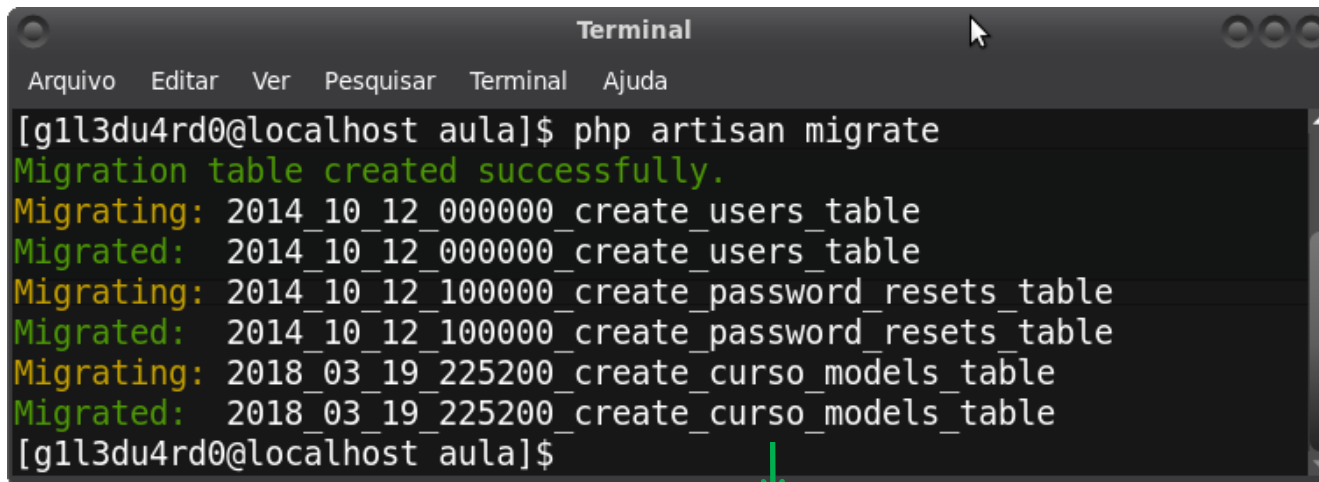
No arquivo de migração criado, especificamos os campos da tabela que ficará vinculada a classe de modelo criada. Essa *migration* será utilizada para gerar uma nova tabela no banco.



Framework Laravel

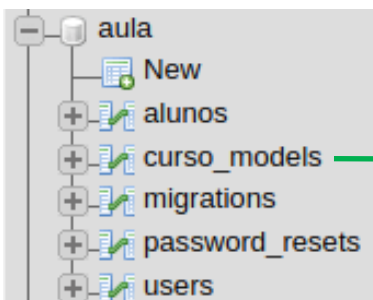
Criando Tabela no Banco via Migration

(Considerando que os arquivos de configuração para o banco de dados estão corretos)



```
Terminal
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda

[g1l3du4rd0@localhost aula]$ php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table
Migrating: 2018_03_19_225200_create_curso_models_table
Migrated: 2018_03_19_225200_create_curso_models_table
[g1l3du4rd0@localhost aula]$
```



Após executar o comando ***php artisan migrate*** a tabela é criada no banco. Observe que mais algumas *migrations* também já são geradas durante a criação do projeto Laravel, elas são utilizadas pelo framework para autenticação dos usuários. Caso você queira desfazer uma migração pode usar o comando ***php artisan migrate:rollback***.



Framework Laravel

Povoando as Tabelas via Seeder

(Localização: `"/database/seeds/DatabaseSeeder.php"`)

- o recurso **"seeder"**, do Laravel, permite inserir dados no banco durante a criação do projeto;
- Essa funcionalidade permite que o desenvolvedor teste a aplicação durante o seu desenvolvimento;





Framework Laravel

Povoando as Tabelas via Seeder

(Editando o arquivo: *"/database/seeds/DatabaseSeeder.php"*)

```
public function run() {  
  
    DB::insert('INSERT INTO alunos(nome, curso, turma) VALUES(?, ?, ?)',  
        array('João Paulo França', 'TADS', 'TADS18'));  
    DB::insert('INSERT INTO alunos(nome, curso, turma) VALUES(?, ?, ?)',  
        array('Mariana Ramos Albernaz', 'INFO', 'INFO12'));  
    DB::insert('INSERT INTO curso_models(nome, abreviatura) VALUES(?, ?)',  
        array('ANÁLISE E DESENVOLVIMENTO DE SISTEMAS', 'TDAS'));  
    DB::insert('INSERT INTO curso_models(nome, abreviatura) VALUES(?, ?)',  
        array('EMI EM INFORMÁTICA', 'INFO'));  
}
```



Dentro do arquivo *DatabaseSeeder.php* – método *run()* – definimos os dados que devem ser inseridos nas tabelas do banco de dados. Utilizamos para isso *queries SQL*.



Framework Laravel

Povoando as Tabelas via Seeder

(Executando o comando: *"php artisan db:seed"*)

```
Terminal
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda

[g1l3du4rd0@localhost aula]$ php artisan db:seed
[g1l3du4rd0@localhost aula]$
```

+ Opções

				id	nome	curso	turma
<input type="checkbox"/>		Edita		Copiar		Apagar	1 João Paulo França TADS TADS18
<input type="checkbox"/>		Edita		Copiar		Apagar	2 Mariana Ramos Albernaz INFO INFO12

Após a execução do comando os dados são inseridos nas respectivas tabelas.

+ Opções

<div><div><div></div><div></div><div></div></div><div></div></div>							id	nome	abreviatura
<input type="checkbox"/>		Edita		Copiar		Apagar	1	ANÁLISE E DESENVOLVIMENTO DE SISTEMAS	TDAS
<input type="checkbox"/>		Edita		Copiar		Apagar	2	EMI EM INFORMÁTICA	INFO



Framework Laravel

Selecionando Dados do Banco – Eloquent

```
CursoController.php x
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use App\CursoModel;
7
8 class CursoController extends Controller {
9
10     public function listar() {
11
12         $cursos = CursoModel::orderBy('nome')->get();
13         return view('cursos')->with('cursos', $cursos);
14     }
```

Especifica que a classe de **Controle** fará uso dos métodos disponíveis na classe de **Modelo**.

Efetua um **SELECT** (método **get()** do **Eloquent**) na tabela vinculada a classe **CursoModel** ordenando os dados de acordo com o campo **nome**.

Invoca a **view** dos cursos passando como parâmetro (método **with()**) o **array de dados** (\$cursos) obtido pelo método **get()** do **Eloquent**.



Framework Laravel

Exibindo os Dados na View com *Blade*

```
<tbody>
@foreach ($cursos as $dados)
    <tr>
        <td>{{ $dados->id }}</td>
        <td>{{ $dados->nome }}</td>
        <td>{{ $dados->abreviatura }}</td>
        <td>
            <a href="{{ action('CursoController@editar', $dados->id) }}">Editar</span></a>
            &nbsp;
            <a href="{{ action('CursoController@confirmar', $dados->id) }}">Remover</span></a>
        </td>
    </tr>
@endforeach
</tbody>
```

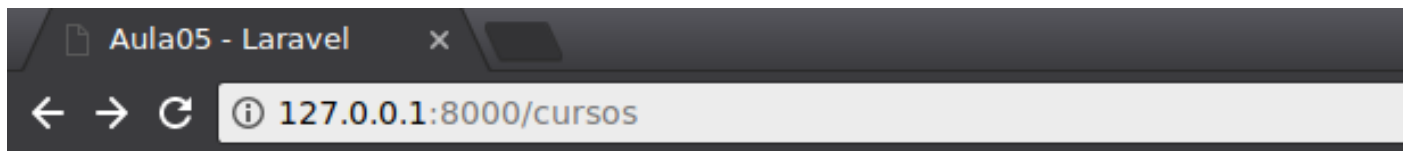


O **blade** possui algumas funções de programação conhecidas como, por exemplo, o laço de repetição **foreach** que possibilita percorrer o array **\$cursos** enviado anteriormente pela classe de controle. Observe também, que o **blade** possui uma função **action()** que permite invocar e passar parâmetro para um método de uma classe.



Framework Laravel

Resultado das codificações:



[Home](#)

BD / Migration / Seeder / Model / Controller



Cursos Cadastrados

[Cadastrar Novo Curso](#)

ID	NOME DO CURSO	ABREVIATURA	EVENTOS
1	ANÁLISE E DESENVOLVIMENTO DE SISTEMAS	TDAS	Editar Remover
2	EMI EM INFORMÁTICA	INFO	Editar Remover

©2018 » Gil Eduardo de Andrade





Framework Laravel

Selecionando Dados do Banco – *Eloquent*

```
AlunoController.php x
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use App\AlunoModel;
7
8  class AlunoController extends Controller {
9
10     public function listar() {
11
12         $alunos = AlunoModel::orderBy('nome')->get();
13         return view('alunos')->with('alunos', $alunos);
14     }
```

Mesmo procedimento adotado anteriormente, mas agora para classe de Modelo do aluno, que foi criada sem a utilização do recurso de migração. Isso permite visualizar que independente da forma como a classe de modelo foi criada, a leitura dos dados da tabela ao qual ela está vinculada acontece da mesma maneira.





Framework Laravel

Exibindo os Dados na View com *Blade*

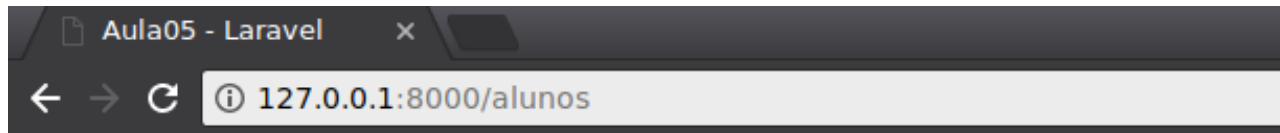
```
<tbody>
@foreach ($alunos as $dados)
    <tr>
        <td>{{ $dados->id }}</td>
        <td>{{ $dados->nome }}</td>
        <td>{{ $dados->curso }}</td>
        <td>
            <a href="{{ action('AlunoController@editar', $dados->id) }}">Editar</span></a>
            &nbsp;
            <a href="{{ action('AlunoController@confirmar', $dados->id) }}">Remover</span></a>
        </td>
    </tr>
@endforeach
</tbody>
```





Framework Laravel

Resultado das codificações:



[Home](#)

BD / Migration / Seeder / Model / Controller



Alunos Cadastrados

[Cadastrar Novo Aluno](#)

ID	NOME DO ALUNO	CURSO	EVENTOS
1	João Paulo França	TADS	Editar Remover
2	Mariana Ramos Albernaz	INFO	Editar Remover

©2018 » Gil Eduardo de Andrade





Framework Laravel

Cadastrando Dados – Formulário no Laravel

- A utilização de formulários HTML no Laravel exige que um `<input type='hidden'>` contendo um ***token*** de segurança seja declarado;
- Esse procedimento evita que um usuário mal intencionado crie seu próprio formulário e tente enviá-lo via ***POST*** para uma ***URL*** de entrada do projeto;





Framework Laravel

Cadastro: Novo Curso – Formulário HTML

(Arquivo: *CursoCadastrar.blade.php*)

```
<form action="{ { action('CursoController@salvar', 0) } }" method="POST">
  <input type="hidden" name="_token" value="{ { csrf_token() } }">

  <label>Nome: </label>
  <input type="text" name="nome" class="form-control">
  <br>
  <label>Abreviatura: </label>
  <input type="text" name="abreviatura" class="form-control">
  <br>
  <button type="submit" class="btn btn-success btn-block">Salvar</button>
</form>
```



Observe que, obrigatoriamente, o formulário Laravel deve possuir uma entrada de dado do tipo *hidden* com o nome *token* e valor obtido da função *blade* "*csrf_token()*". Se essa premissa não for atendida a aplicação apresentará um erro.





Framework Laravel

Cadastro: Novo Curso - Detalhes

(Arquivos: *CursoController.php* e *CursoModelo.php*)

```
use Request;
use App\CursoModel;

class CursoController extends Controller {

    public function cadastrar() {
        return view('cursoCadastrar');
    }
}
```

Para que seja possível trabalhar com a classe ***Request*** de maneira estática (sem instanciar um objeto), podemos alterar a sua inclusão de ***"Illuminate\Http\Request"*** para apenas ***"Request"***.

A classe ***Request***, dentro do Laravel, manipula os dados vinculados aos formulários HTML. Sendo assim, podemos utilizar objetos dessa classe para recuperar os dados inseridos pelos usuários nos campos do formulário que está sendo manipulado.





Framework Laravel

Cadastro: Novo Curso - Detalhes

(Arquivos: *CursoModelo.php*)

```
use Illuminate\Database\Eloquent\Model;

class CursoModel extends Model {

    public $timestamps = false;
}
```

Por padrão, o Laravel trabalha com dois campos do tipo **timestamp** nas tabelas do banco. Eles são utilizados para armazenar o momento em que o registro da tabela foi criado e atualizado pela aplicação. Sendo assim sempre que uma inserção ou atualização é efetuada, por padrão ele altera os valores desses campos.

Considerando que o objetivo dessa aula é aprender como funcionam as **migrations** em conjunto com as classes de modelo relacionadas a elas, durante o processo de migração os campos de **timestamp** foram retirados do arquivo de migração relativo a tabela `curso_models` criada. Isso indica que a nossa aplicação não trabalhará com os campos de **timestamp**, contudo quando essa opção é utilizada torna-se necessário explicitá-la dentro da classe modelo, através do atribuo `$timestamp`, pelo código **`public $timestamp = false;`**






Framework Laravel

Cadastro: Novo Curso – Método Controle

(Arquivo: *CursoController.php*)

```
public function salvar($id) {  
    // INSERT  
    if($id == 0) {  
        $objCursoModel = new CursoModel();  
        $objCursoModel->nome = mb_strtoupper(Request::input('nome'), 'UTF-8');  
        $objCursoModel->abreviatura = mb_strtoupper(Request::input('abreviatura'), 'UTF-8');  
        $objCursoModel->save();  
    }  
  
    return redirect()->action('CursoController@listar')->withInput();  
}
```



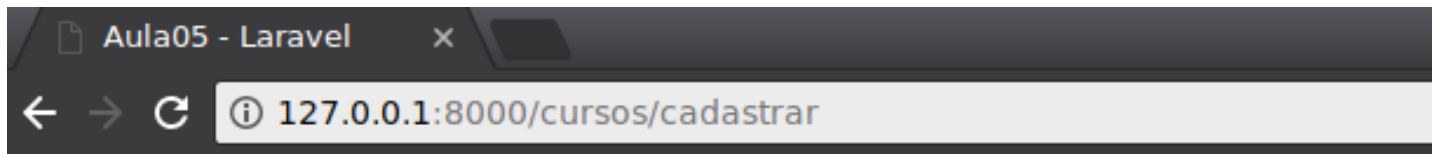
Um objeto da classe *CursoModel* é criado, visto que a mesma estende da superclasse *Model* (*Eloquent*). Sendo assim, ele já possui nativo todos os métodos necessário interagir com a base de dados. A classe *Request*, método *input()* é utilizada de maneira estática para recuperar os campos *nome* e *abreviatura* do formulário de cadastro. O método *save()* faz a inserção dos dados na tabela. A rotina *redirect()->action()* redireciona a página no final;



Framework Laravel

Resultado das codificações:

(Durante o cadastro)



[Home](#)

BD / Migration / Seeder / Model / Controller



Cadastrar Novo Curso

Nome:

Abreviatura:

Salvar

©2018 » Gil Eduardo de Andrade

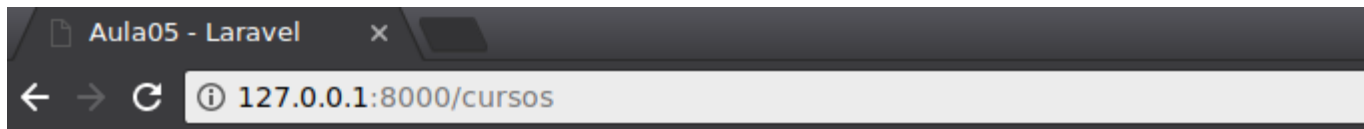




Framework Laravel

Resultado das codificações:

(Após o cadastro)



[Home](#)

BD / Migration / Seeder / Model / Controller



Cursos Cadastrados

[Cadastrar Novo Curso](#)

ID	NOME DO CURSO	ABREVIATURA	EVENTOS
1	ANÁLISE E DESENVOLVIMENTO DE SISTEMAS	TDAS	Editar Remover
2	EMI EM INFORMÁTICA	INFO	Editar Remover
4	PÓS-GRADUAÇÃO EM MATEMÁTICA COMPUTACIONAL	PGMC	Editar Remover

©2018 » Gil Eduardo de Andrade





Conceitos Iniciais

Exemplos Utilizados no Documento

http://www.gileduardo.com.br/ifpr/dwii/downloads/dwii_exdoc06.zip

Código-fonte da Aplicação SETA

http://www.gileduardo.com.br/ifpr/dwii/downloads/dwii_dica06.zip

Exercício sobre o Conteúdo

http://www.gileduardo.com.br/ifpr/dwii/downloads/dwii_pratica06.pdf

