1. MVC

Já falamos da View:
Responsável pela parte visual do site do nosso projeto.
Utiliza o Blade para facilitar a criação das páginas HTML que vamos precisar.

Já falamos do Controller:
Responsável por decidir (ou controlar) o que precisa ser feito depois que o usuário acessou uma rota do projeto.
É ele quem vai, por exemplo, selecionar alguma informação do banco de dados e encaminhar para a visualização exibir para o usuário.

E para completar os 3 elementos (ou as 3 camadas) do MVC: hoje vamos falar do Model.

2. MODEL

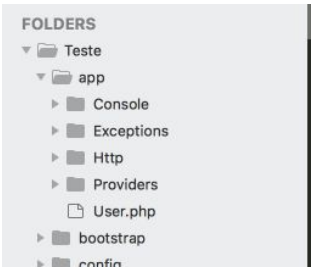
Resumindo de uma forma bastante simples o Model (ou o modelo) representa uma tabela do nosso banco de dados.
Trata-se de uma classe que vai permitir manipular um ou mais registros de uma determinada tabela do MySql (por exemplo, ou do banco de dados que estiver utilizando).

Da mesma forma como as Views do Laravel utilizam a biblioteca chamada Blade.
Os Models utilizam uma biblioteca conhecida como Eloquent (ORM).

Isso é bastante transparente para o programador.
Você não precisa se preocupar em transformar uma linha (ou registro) do banco de dados em uma classe do Eloquent.
O Laravel já trata de fazer isso automaticamente pra gente.

Ele vai fazer o papel do PDO que utilizamos no curso anterior.
Na verdade por debaixo dos panos o Laravel utiliza o PDO do banco de dados que está utilizando (no caso o MySql) de forma automática.

Todos os modelos ficam alí mesmo na pasta `app` do nosso projeto.



Neste exemplo o arquivo `User.php` é um modelo já criado.

3. Criando um Model

Para criar um modelo vamos utilizar um comando do artisan:

```
php artisan make:model Usuario
```

Para seguir as boas práticas, o modelo também deverá ter o nome iniciado com letra maiúscula e sempre no singular (não no plural).

Ao executar esse comando o modelo será criado dentro da pasta `app` do seu projeto Laravel.

4. Configurando o Model

O novo modelo criado não possui nenhum código. Ele vem mais ou menos assim:

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Usuario extends Model
{
    //
}
```

Então, a única coisa que precisamos fazer é dizer para o Laravel qual é a tabela do seu banco de dados que este modelo representa.
E também qual é o campo chave-primária desta tabela.

Para fazer isso basta criar 2 variáveis:

- `protected $table`
- `protected $primaryKey`

Assim:

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Usuario extends Model
{
    protected $table = 'usuario';

    protected $primaryKey = 'usuario_id';

}
```

Pronto!

Agora o Laravel já sabe qual é a tabela e a chave primária que este modelo deve utilizar.

5. Como fica o Controller?

Agora podemos utilizar esse novo modelo lá no seu controlador.

Para isso, antes de tudo precisamos informar o controlador que vamos utilizar esse modelo. Basta colocar no início do controller a linha de comando:

```
use App\Usuario;
```

Assim:

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Usuario;

class UsuarioController extends Controller
{
```

E para solicitar ao Laravel que vá até o banco de dados e selecione todos os registros da tabela, por exemplo todos os usuários, basta escrever seu código assim:

```
$usuario = Usuario::all();
```

O método `all()` serve exatamente para buscar todos os registros de uma tabela do seu banco de dados.

No controller ficaria mais ou menos assim:

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Usuario;

class UsuarioController extends Controller
{
    public function index()
    {
        $usuarios = Usuario::all();

        return view('exibir-todos-usuarios')->with('listaDeUsuarios', $usuarios);
    }
}
```

6. Outros Métodos do Model?

```
// Localiza todos os filmes cadastrados na tabela
$filmes = Filme::all();

// Localiza todas as informações do filme com id igual a 1
$meufilme = Filme::find(1);

// Obtém apenas o primeiro ou o último registro da tabela (ordenados pelo id)
$primeirofilme = Filme::first();
$ultimofilme = Filme::last();

// Gera uma pesquisa no banco de dados a partir do nome
$pesquisa = Filme::where( 'nome', '=', 'Divertida Mente')->get();

// É possível montar pesquisas encadeando condições de pesquisas ou de ordenação
$pesquisa = Filme::where( 'nome', '=', 'Divertida Mente')
                  ->where('rating', '>', '8')
                  ->get();
```