PHP MVC

DIO
CEZAR
GO

O QUE É?

- MVC é um padrão de projeto amplamente difundido em ambientes *back-end*;
- foi importado dos conceitos aplicados a *Desktop*;
- o importante do MVC é entender além do que os frameworks podem oferecer;
  - quase sempre um padrão já definido e seguido, sem saber: como, onde ou porque;
- para isso é importante revisar as camadas do MVC;

O QUE É?

- além disso, o MVC é uma discussão na comunidade de desenvolvimento web;
- fazendo comparações:
  - *Controller* → responsável pelo roteamento;
  - *Model* → responsável pela regra de negócios e validação dos dados;
  - e a *View*?
    - é o HTML que acessa o PHP pelo JavaScript?
    - é o PHP que escreve o HTML já formatado?
- conclusão → nada definido!

O QUE É MODEL?

- *model* é onde fica a lógica da aplicação. só isso.
- precisa disparar um e-mail?
- validar um formulário?
- enviar ou receber dados do banco de dados?
- ...não importa!
- a *model* deve saber como executar toda a regra de negócios da sua aplicação;
- mas importante! a *model* não precisa saber QUANDO deve ser feito, nem como MOSTRAR os dados;

O QUE É VIEW?

- *view* recebe dados, e só!
- a *view* não é só HTML;
- ela também é qualquer tipo de retorno de dados como: PDF, JSON, XML etc...
- sempre que se precisa de um retorno de dados para o navegador, é responsabilidade da *view*;
- a *view* precisa saber renderizar os dados corretamente, mas não precisa saber como OBTÊ-LOS, ou QUANDO renderiza-los;

O QUE É CONTROLLER?

- o *controller* diz quando as coisas devem acontecer, e nada mais!
- é utilizado para intermediar a *model* e a *view*;
- por exemplo, para pegar os dados da model (guardados em um banco) e exibí-los na *view* (em uma página HTML);
- também é responsabilidade do *controller* cuidar das requisições (*request* e *response*) e isso também inclui os famosos *middlewares*;
    - *middlewares* são operações a serem realizadas pré ou pós a chamada de um *controller*;

ESTRUTURA PADRÃO

- Na raiz do diretório do seu projeto crie estes 5 arquivos (e diretórios):
  - `src/App/Mvc/Controller.php`
  - `src/App/Mvc/Model.php`
  - `src/App/Mvc/View.php`
  - `composer.json`
  - `index.php`

UTILIZAREMOS O PSR-4 E COMPOSER

- o PSR-4 é uma especificação para *autoloading* de classes;
- deve-se então criar um arquivo composer.json com a seguinte estrutura e rodar o comando: `composer install`

COMPOSER.JSON

```
{
   "autoload": {
      "psr-4": {
         "App\\": "src/App"
      }
   }
}
```

EXEMPLO MODEL

MODEL.PHP

```php
<?php
   namespace App\Mvc;
   class Model
   {
      public function getText($str = 'Olá mundo!')
      {
         return $str;
      }
   }
?>
```

EXEMPLO VIEW

VIEW.PHP

```php
<?php
   namespace App\Mvc;
   class View
   {
      public function render($str)
      {
         echo $str;
      }
   }
?>
```

EXEMPLO CONTROLLER

CONTROLLER.PHP

```php
<?php
   namespace App\Mvc;
   class Controller
   {
      public function index()
      {
         $model = new Model;
         $view = new View;
         $view->render($model->getText());
      }
   }
?>
```

EXEMPLO INDEX

INDEX.PHP

```php
<?php
      require 'vendor/autoload.php';
      $controller = new App\Mvc\Controller();
      $controller->index();
?>
```

MATERIAIS COMPLEMENTARES

- https://goo.gl/UNxzwr