Curso PHP

Do XLSX ao CMS (aula 10)

Model View Controller

MVC

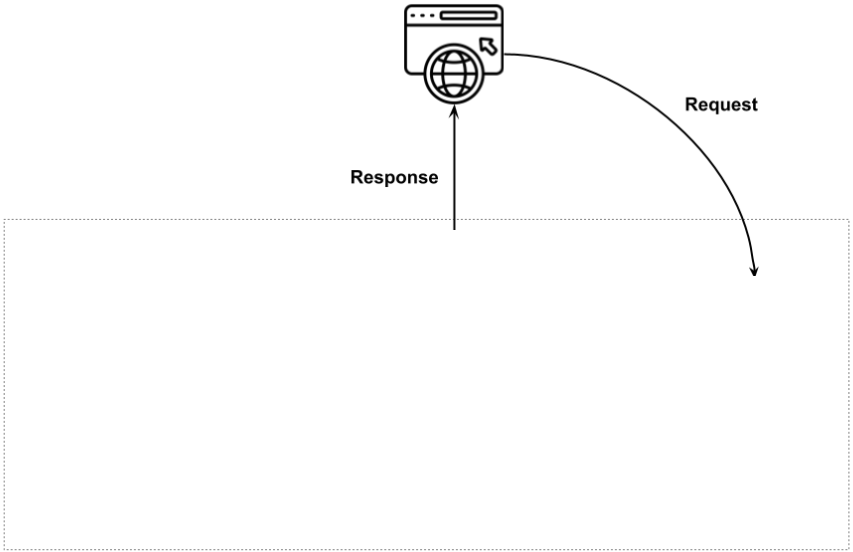
A Arquitetura Modelo-Visão-Controlador (MVC) é um padrão de design amplamente utilizado na construção de aplicativos de software.

Ele visa separar as preocupações de uma aplicação em três componentes principais: Modelo, Visão e Controlador.

Essa separação oferece uma série de benefícios, como modularidade, manutenibilidade e reutilização de código.

(Ângelo Souza)

Request lifecycle



O ciclo que requisição (input) começa quando o usuário envia uma requisição à aplicação.

Essa requisição será recebida pelo roteador, que irá buscar pela rota que atenda à URI.

Uma vez encontrada a rota, será resolvido as dependências (argumentos) do *callback*.

A resposta (output) será o resultado (processing) do *callback*.

HttpFoundation

No PHP, a requisição é representada por algumas variáveis globais (**$_GET**, **$_POST**, **$_FILES**, **$_COOKIE**, **$_SESSION**, ...) e a resposta é gerada por algumas funções (*echo*, *header()*, *setcookie()*, ...).

O componente **Symfony HttpFoundation** substitui essas variáveis e funções globais padrão do PHP por uma camada orientada a objetos.

(The HttpFoundation Component)

composer require symfony/http-foundation

Instala a dependência

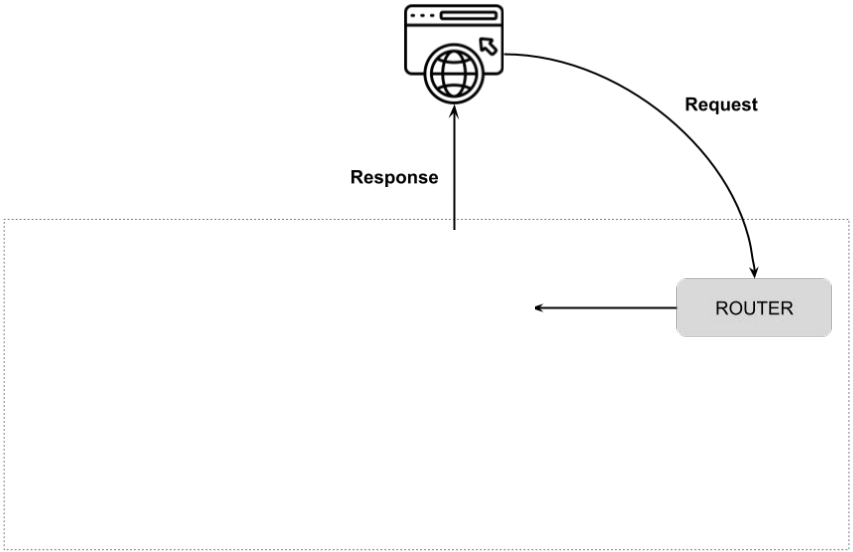
Rotas

Router

Um componente frequentemente negligenciado ou percebido como parte do controlador é o mecanismo de roteamento.

É um dos componentes mais importantes do MVC, que inicialmente recebem a solicitação do cliente e alocam qual controlador vai lidar com a solicitação.

(Bilal Haidar)



Router

Nossas rotas residem no arquivo **src/routes.php**.

Para cada rota, usamos verbos HTTP que induzem ao tipo de requisição recebida.

```php
<?php

use App\Http\Router;

// página inicial
Router::redirect('/', '/agendamentos');

// agendamentos
Router::get('/agendamentos', fn () => 'Agendamentos');
```

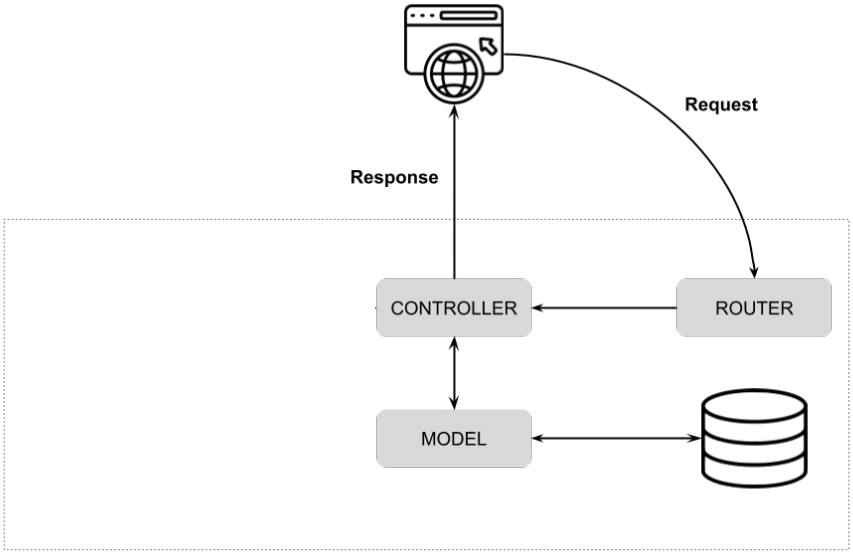
Controladores

Controller

O componente *controller* atua como um mediador entre os componentes *view* e *model*.

- Recebe uma solicitação do cliente para uma *view* específica.
- Coordena com o componente *model* para consultar dados (ou atualizar dados).
- Empacota a *view* junto com os dados relacionados em uma única resposta.

(Bilal Haidar)



Controllers

Controllers são (e devem ser) classes simples.

Esta classe não deve processar dados, ou seja, apenas fazem a intermediação entre o que foi recebido pela requisição e o que seja enviado como resposta.

```php
<?php

use App\Controllers\AgendamentoController;
use App\Http\Router;

// página inicial
Router::redirect('/', '/agendamentos');

// agendamentos
Router::get('/agendamentos', AgendamentoController::class);
```

```php
<?php

namespace App\Controllers;

class AgendamentoController
{
    public function __invoke()
    {
        return 'Agendamentos';
    }
}
```

Injeção de dependência

DI

Injeção de dependência (DI, Dependency Injection) é um design pattern utilizado para separar a criação de um objeto de suas dependências.

Quando estamos utilizando injeção de dependência não instanciamos nossas dependências dentro da classe que estamos trabalhando, mas sim injetamos como parâmetros.

(Matheus Honorato)

```php
<?php

use App\Controllers\AgendamentoController;
use App\Http\Router;

// página inicial
Router::redirect('/', '/agendamentos');

// agendamentos
Router::get('/agendamentos', [AgendamentoController::class, 'index']);
Router::get('/agendamentos/{agendamento}', [AgendamentoController::class, 'ver']);
```

```php
<?php

namespace App\Controllers;

use App\Models\Agendamento;

class AgendamentoController
{
    public function index()
    {
        return 'Agendamentos';
    }

    public function ver(Agendamento $agendamento)
    {
        return "Visualizando agendamento #{$agendamento->id}: {$agendamento->data}";
    }
}
```

Um momento de reflexão...

PHP Reflection

Trabalhar com reflexão no PHP é possível graças as classes mágicas de Reflection do PHP. Essas classes estão disponíveis no *core* da linguagem desde a versão 5, então não é necessário fazer nenhuma instalação. Existem algumas classes de Reflection no PHP, sendo que cada uma depende de onde você vai aplicar:

- ReflectionClass: classes
- ReflectionFunction: funções
- ReflectionMethod: métodos (funções em uma classe)
- ReflectionParameter: parâmetros (de métodos ou funções)

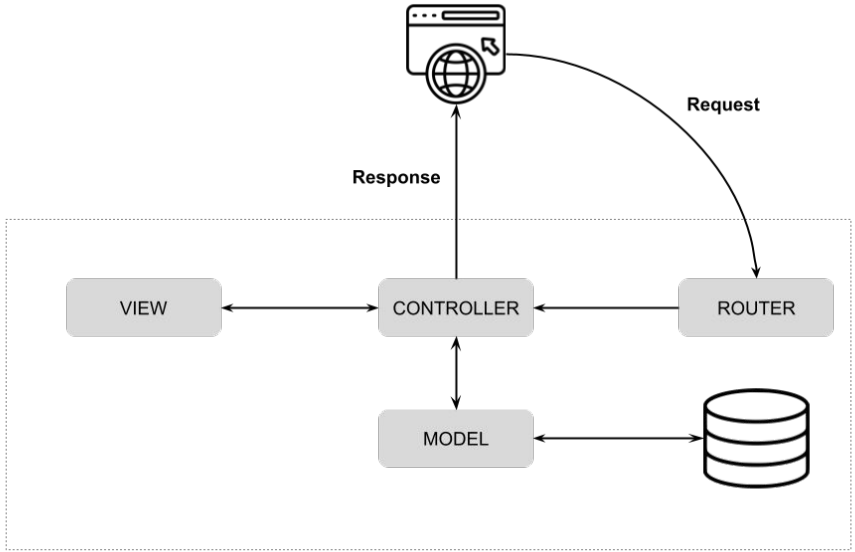
(Marcos Felipe)

Visões

View

- Gera e renderiza a interface do usuário (UI) do aplicativo.
- É composto por HTML/CSS e possivelmente JavaScript.
- Recebe os dados do *controller*, que recebeu os dados do *model*.
- Mescla os dados com a estrutura HTML para gerar a interface do usuário.

(Bilal Haidar)



View

Para interpretar nossas *views*, usaremos o *template engine* **Twig**.

Ao usar um *template engine*, temos a vantagem de escrever um código próximo ao HTML com funcionalidades PHP, mas sem escrever PHP de fato neste arquivo.

Isso nos permite criar layouts, incluir arquivos e reaproveitar trechos e tantos outros recursos que o PHP nos permite por ser uma linguagem dinâmica.

composer require twig/twig

Instala a dependência

Twig

Como dito antes, o arquivo de *view* usa uma sintaxe próxima ao HTML, mas com particularidades chamadas de delimitadores.

Existem dois tipos de delimitadores:

- {% ... %}
- {{ ... }}

O primeiro é usado para executar instruções como loops, o último produz o resultado de uma expressão.

(Twig Documentation)

Views: criação e uso

```php
<?php

namespace App\Controllers;

use App\Http\View;
use App\Models\Agendamento;

class AgendamentoController
{
    public function index()
    {
        return View::render('agendamentos/index', [
            'agendamentos' => Agendamento::query()->get(),
        ]);
    }

    public function ver(Agendamento $agendamento)
    {
        return View::render('agendamentos/ver', [
            'agendamento' => $agendamento,
        ]);
    }
}
```

```twig
<h1>Agendamentos</h1>
<ul>
    {% for agendamento in agendamentos %}
        <li>
            <a href="/agendamentos/{{ agendamento.id }}">#{{ agendamento.id }}</a>: {{ agendamento.data }}
        </li>
    {% endfor %}
</ul>
```

```twig
<h1>Agendamento #{{ agendamento.id }}</h1>
<h3><a href="/agendamentos">&lsaquo; Voltar</a></h3>
<p>
    <strong>Agendamento #{{ agendamento.id }}</strong><br>
    Data: {{ agendamento.data|date('d/m/Y') }}
</p>
```

No próximo episódio: criando “telas”.