Curso PHP

Do XLSX ao CMS (aula 15)

Middlewares

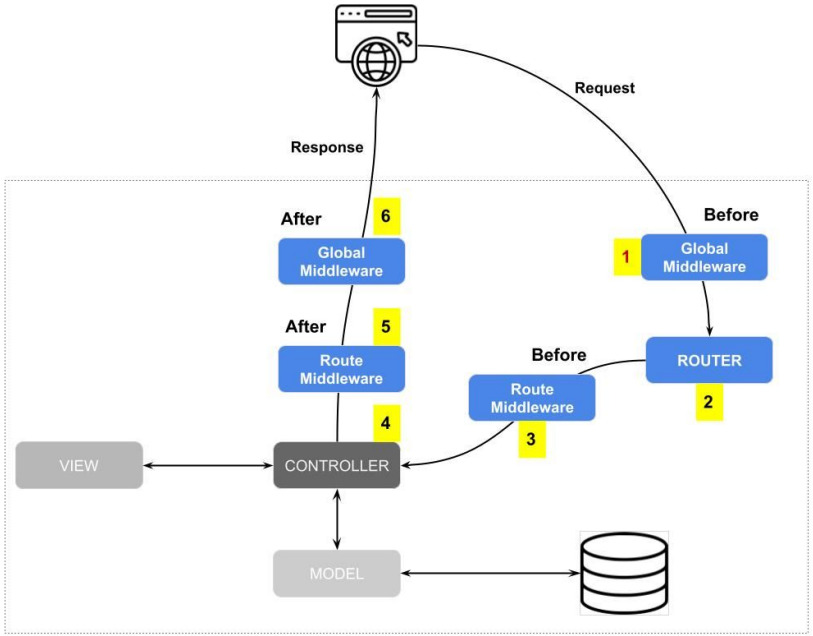
Middlewares

Middleware em PHP é uma camada de ações/*callables* que são embrulhados em torno de um pedaço de lógica central em um aplicativo.

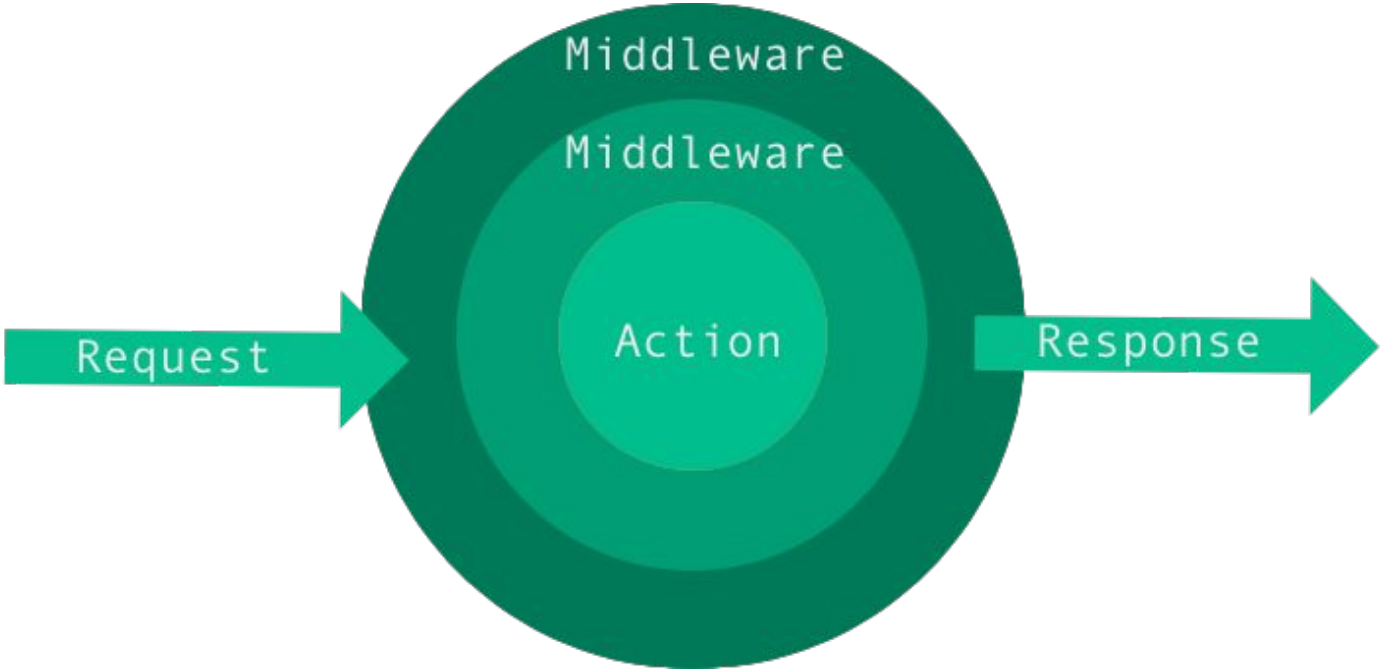
Esses middlewares fornecem a capacidade de alterar a entrada ou a saída dessa lógica.

Então eles "vivem" entre a entrada e a saída, bem no meio.

(Doeken)



Fluxo HTTP



Fluxo HTTP através de middlewares

A ordem interfere no progresso?

Ordem e progresso

Os middlewares trabalham em **ordem estritamente exclusiva**, isto é, eles devem seguir uma lógica, pois as modificações de um irá afetar no trabalho do outro.

Perceba que o middleware de **início de sessão** está antes da **verificação do token CSRF**, pois o segundo depende do primeiro.

Entre eles, está o middleware para configurar as **constantes da *view***, pois, em caso de erro, será retornada uma resposta e esta resposta precisa de tais contantes.

```php
// captura requisição
$request = Request::boot();

// despacha rota e captura resposta
$response = Router::dispatch($request, [
    StartSession::class,
    ConfigureView::class,
    ConfigurePaginator::class,
    EnableHttpMethodParameterOverride::class,
    ValidateCsrfToken::class,
]);

// envia resposta
$response->send();
```

Simples, mas eficazes.

Simples e eficaz

Use uma nomenclatura sugestiva.

Por exemplo, **StartSession**. Apenas passando olho nisso, já entendemos que irá inicializar a sessão, independente de ver a implementação.

Como dito antes, os middlewares são partes específicas de uma lógica complexa, por conseguinte, devem ser simples.

```php
namespace App\Middlewares;

class StartSession
{
    public function __invoke(Request $request): bool
    {
        $handler = new PdoSessionHandler(DB::connection()->getPdo());
        $storage = new NativeSessionStorage(handler: $handler);
        $session = new Session($storage);

        $request->setSession($session);

        return true;
    }
}
```

```php
namespace App\Middlewares;

class EnableHttpMethodParameterOverride
{
    public function __invoke(Request $request): bool
    {
        $request->enableHttpMethodParameterOverride();

        return true;
    }
}
```

Mais um exemplo de simplicidade e eficácia

```php
class ValidateCsrfToken
{
    public function __invoke(Request $request): bool|Response
    {
        if (in_array($request->getMethod(), [
            BaseRequest::METHOD_DELETE,
            BaseRequest::METHOD_PATCH,
            BaseRequest::METHOD_POST,
            BaseRequest::METHOD_PUT,
        ])) {
            return $request->verifyCsrfToken() ?: response('erro_400', status: Response::HTTP_BAD_REQUEST);
        }

        return true;
    }
}
```

Resposta imediata

Área pública e privada será o tema de nossa próxima aventura.