

Frameworks em PHP

Elton Minetto

@eminetto

<http://eltonminetto.net>

eminetto@gmail.com

Perguntas

- O que é um framework?
 - Por que usar?
- Quais são as vantagens e desvantagens?
- Quais são as diferenças entre os frameworks?
 - Quais são os principais frameworks?
 - Como escolher um framework?

Lançamento rápido, projetos menores, sites

**+ liberdade
+ configuração
- regras**

**- liberdade
- configuração
+ regras**

Projetos complexos, maior longevidade

Lançamento rápido, projetos menores, sites

Laravel

Cake

Yii

+ liberdade
+ configuração
- regras

- liberdade
- configuração
+ regras

Projetos complexos, maior longevidade

Lançamento rápido, projetos menores, sites

Laravel

Cake

Yii

+ liberdade
+ configuração
- regras

- liberdade
- configuração
+ regras

Symfony 3

Symfony 2

Symfony 1

ZF3

ZF2

ZF1

Projetos complexos, maior longevidade

Lançamento rápido, projetos menores, sites

Laravel

Cake

Yii

+ liberdade
+ configuração
- regras

- liberdade
- configuração
+ regras



Slim

Lumen

Silex

Expressive

Symfony 3

Symfony 2

Symfony 1

ZF3

ZF2

ZF1

Projetos complexos, maior longevidade

PSR-7

***Http Is The Foundation Of
The Web***

Um cliente manda uma request

```
$method = $request->getMethod();  
$accept = $request->getHeader('Accept');  
$path = $request->getUri()->getPath();  
$controller = $request->getAttribute('controller');
```

O servidor retorna uma response

```
$response->getBody()->write('Hello world!');  
$response = $response  
    ->withStatus(200, 'OK')  
    ->withHeader('Content-Type', 'text/plain');
```

Middlewares

Between the request and response

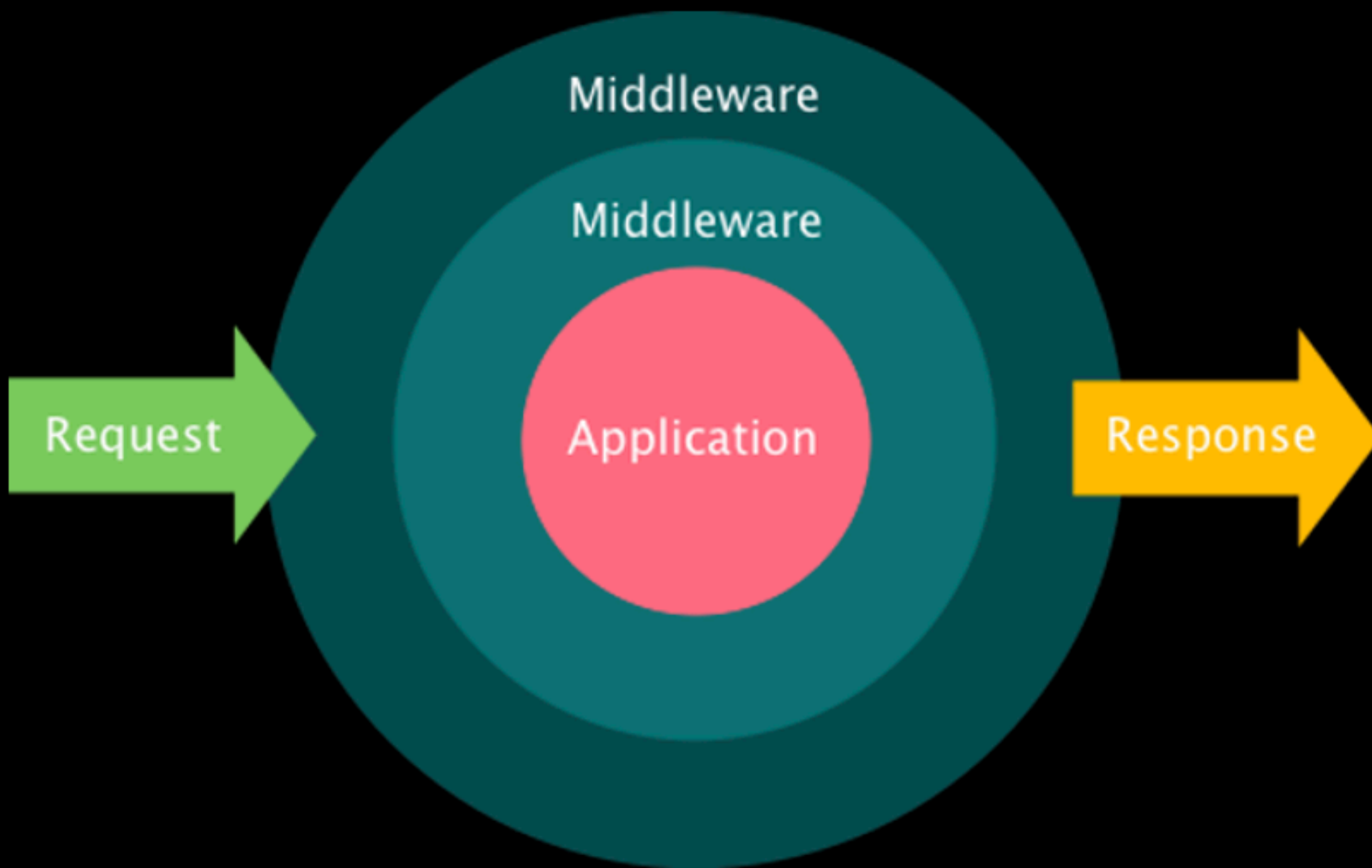
Enrico Zimuel

HTTP middleware is not for your Domain work. The middleware is a path in to, and out of, the core Domain.

Paul M. Jones

In other words, we'll no longer write Zend Framework or Symfony or Laravel or framework-flavor-of-the-day applications or modules/bundles/packages/what-have-you. We'll write middleware that solves a discrete problem, potentially using other third-party libraries, and then compose them into our applications — whether those are integrated into a framework or otherwise.

Matthew O'phhinney



```
class Auth implements MiddlewareInterface
{
    public function __invoke(Request $request, Response $response, callable $out = null)
    {
        if(! $request->hasHeader('authorization')){
            return $response->withStatus(401);
        }

        if (!$this->isValid($request)) {
            return $response->withStatus(403);
        }

        return $out($request, $response);
    }
}
```

```
$api = AppFactory::create();  
$api->get('/', function ($request, $response, $next) {  
    $response->getBody()->write('Hello, beers of world!');  
    return $response;  
});  
$api->get('/doc', function ($request, $response, $next) {  
    $response->getBody()->write('Hello, beers of world!');  
    return $response;  
});
```

```
$app = AppFactory::create();  
$app->pipe(new Auth());  
$app->pipe($api);  
$app->pipe(new Format());
```

```
$app->run();
```


PSR-15

PSR-15 HTTP Middleware describes a common standard for HTTP middleware components using HTTP Messages defined by PSR-7.

Currently, PSR-15 is a PHP Standards Recommendation Proposal of the Framework Interoperability Group (FIG).

```
use Interop\Http\ServerMiddleware\DelegateInterface;
use Interop\Http\ServerMiddleware\MiddlewareInterface;
use Psr\Http\Message\ServerRequestInterface;

class Authentication implements MiddlewareInterface
{
    public function process(ServerRequestInterface $request, DelegateInterface $delegate)
    {
        //do some logic

        return $delegate->process($request, $delegate);
    }
}
```

Exemplo

RestBeer

API about 🍺🍺

Multiple response formats (JSON, HTML)

`http://restbeer.com/brands`

`http://restbeer.com/styles`

`http://restbeer.com/beer/1`

Zend Expressive 2

Expressive allows you to write PSR-7 middleware applications for the web. It is a simple micro-framework built on top of Stratigility, providing:

Dynamic routing

Dependency injection via container-interop

Templating

Error Handling

<http://devzone.zend.com/6615/announcing-expressive/>

Arquitectura

Router

- FastRoute
- Zend Route
- Aura.Router

Container

- Aura.DI
- Pimple
- Zend Service Manager

Template Engine

→ Twig

→ Plates

→ Zend View

Instalando

```
curl -s http://getcomposer.org/installer | php
```

```
php composer.phar require zendframework/zend-expressive
```

```
php composer.phar require zendframework/zend-expressive-fastroute
```

```
php composer.phar require zendframework/zend-servicemanager
```

Hello World!

```
<?php
use Zend\Expressive\AppFactory;

require 'vendor/autoload.php';

$app = AppFactory::create();

$app->get('/', function ($request, $response, $next) {
    $response->getBody()->write('Hello, world!');
    return $response;
});

$app->pipeRoutingMiddleware();
$app->pipeDispatchMiddleware();
$app->run();
```

Testando


```
php -S localhost:8080
```

Mostrando as cervejas

```
$beers = array(  
    'brands' => ['Heineken', 'Guinness', 'Skol', 'Colorado'],  
    'styles' => ['Pilsen', 'Stout']  
);  
  
$app->get('/brands', function ($request, $response, $next) use ($beers) {  
    $response->getBody()->write(implode(',', $beers['brands']));  
  
    return $response;  
});  
  
$app->get('/styles', function ($request, $response, $next) use ($beers) {  
    $response->getBody()->write(implode(',', $beers['styles']));  
  
    return $response;  
});
```

**Mostrando uma em
específico**

```
$app->get('/beer/{id}', function ($request, $response, $next) use ($beers) {  
    $id = $request->getAttribute('id');  
    if (!isset($beers['brands'][$id])) {  
        return $response->withStatus(404);  
    }  
  
    $response->getBody()->write($beers['brands'][$id]);  
  
    return $response;  
});
```

Adicionando uma cerveja

```
$db = new PDO('sqlite:beers.db');
$app->post('/beer', function ($request, $response, $next) use ($db) {
    $db->exec(
        "create table if not exists
beer (id INTEGER PRIMARY KEY AUTOINCREMENT, name text not null, style text not null)"
    );

    $data = $request->getParsedBody();
    //@TODO: clean form data before insert into the database ;)
    $stmt = $db->prepare('insert into beer (name, style) values (:name, :style)');
    $stmt->bindParam(':name', $data['name']);
    $stmt->bindParam(':style', $data['style']);
    $stmt->execute();
    $data['id'] = $db->lastInsertId();

    $response->getBody()->write($data['id']);

    return $response->withStatus(201);
});
```

Alterando, excluindo e listando da base de dados

(exercício)

Formatando o resultado

📁 **phprs**

📁 **src**

📁 **RestBeer**

🐘 Auth.php

🐘 Format.php

📁 **vendor**

📁 **views**

🌿 content.twig

{ } composer.json

* **composer.lock**

* composer.phar

🐘 index.php

```
php composer.phar require zendframework/zend-expressive-twigrenderer
```

views/content.twig

```
{% for c in content %}  
    {{c}}<br>  
{% endfor %}
```

src/RestBeer/Format.php

```
<?php
```

```
namespace RestBeer;
```

```
use Zend\Stratigility\MiddlewareInterface;
```

```
use Psr\Http\Message\ServerRequestInterface as Request;
```

```
use Psr\Http\Message\ResponseInterface as Response;
```

```
use Zend\Expressive\Twig\TwigRenderer;
```

```
use Zend\Diactoros\Response\JsonResponse;
```

```
use Zend\Diactoros\Response\HtmlResponse;
```

```
class Format implements MiddlewareInterface
```

```
{
```

```
    public function __invoke(Request $request, Response $response, callable $out = null)
```

```
    {
```

```
        $content = explode(',', $response->getBody());
```

```
        $header = $request->getHeader('accept');
```

```
        $accept = null;
```

```
        if (isset($header[0])) {
```

```
            $accept = $header[0];
```

```
        }
```

```
        switch ($accept) {
```

```
            case 'application/json':
```

```
                return new JsonResponse($content, $response->getStatusCode());
```

```
                break;
```

```
            default:
```

```
                $twig = new TwigRenderer();
```

```
                $twig->addPath('views');
```

```
                $html = $twig->render('content.twig', ['content' => $content]);
```

```
                return new HtmlResponse($html);
```

```
                break;
```

```
        }
```

```
    }
```

```
}
```

index.php

```
//1 - alterar o loader
```

```
$loader = require __DIR__.' /vendor/autoload.php';  
$loader->add('RestBeer', __DIR__.' /src');
```

```
//2 - alterar o /brands
```

```
$app->get('/brands', function ($request, $response, $next) use ($beers) {  
    $response->getBody()->write(implode(',', $beers['brands']));  
  
    return $next($request, $response);  
});
```

```
//3 - alterar o pipe
```

```
$app->pipeRoutingMiddleware();  
$app->pipeDispatchMiddleware();  
$app->pipe(new \RestBeer\Format());  
$app->run();
```


Autenticando

src/RestBeer/Auth.php

```
<?php
namespace RestBeer;

use Zend\Stratigility\MiddlewareInterface;
use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Message\ResponseInterface as Response;

class Auth implements MiddlewareInterface
{
    public function __invoke(Request $request, Response $response, callable $out = null)
    {
        if(! $request->hasHeader('authorization')){
            return $response->withStatus(401);
        }

        if (!$this->isValid($request)) {
            return $response->withStatus(403);
        }

        return $out($request, $response);
    }

    private function isValid(Request $request)
    {
        $token = $request->getHeader('authorization');
        //validar o token de alguma forma...
        return true;
    }
}
```

index.php

```
//adicionar o pipe  
$app->pipe(new \RestBeer\Auth());  
$app->pipeRoutingMiddleware();  
$app->pipeDispatchMiddleware();  
$app->pipe(new \RestBeer\Format());  
$app->run();
```

O futuro?

We see ZF3 as a movement: **an end to framework silos**, by providing quality, commodity code that can be used everywhere and anywhere. An **end** to saying **"I'm a ZF developer,"** or **"I'm a Laravel developer"**, and a return to, **"I'm a PHP developer."**

Matthew O'phinnney

Trabalho

- Refatorar o index.php para separar em middlewares
- Implementar uma rota para fazer a autenticação e retornar um token
- Alterar o Auth.php para validar o novo token

Trabalho

- Fazer um fork do repositório <https://github.com/eminetto/frameworks-2017>
 - Implementar no seu fork
- Pode ser feito em grupo. Neste caso colocar os nomes em um arquivo README.me dentro do seu fork

Links

- <https://zendframework.github.io/zend-expressive/>
- <https://danizord.github.io/explorando-o-poder-dos-middlewares/#/>
- <http://eltonminetto.net/post/2017-04-06-iniciando-novo-projeto-decisoes-arquitetura/>