

# A first look at Zend Framework 3

Rob Allen ~ @akrabat ~ October 2015

What did ZF2 give us?

What's wrong with ZF2?

The PHP world has  
changed since 2012

So what's the ZF3 story?

# The ZF3 story

- Components
- ZF2 => ZF3 means MVC improvements!
- PSR-7, Interoperability & Middleware

# PHP 5.5

# Components



# Components

- Separate repositories

# Components

- Separate repositories
- Separate evolution

# Components

- Separate repositories
- Separate evolution
- Documentation in repository

# Components

- Separate repositories
- Separate evolution
- Documentation in repository
- All issues in the right place on GitHub

# Components

- Separate repositories
- Separate evolution
- Documentation in repository
- All issues in the right place on GitHub
- More maintainers

# ZF MVC framework

# MVC improvements

- ZF2 is now a meta package

# MVC improvements

- ZF2 is now a meta package
- Managed BC breaks



# MVC improvements

- ZF2 is now a meta package
- Managed BC breaks
- Key component changes
  - ServiceManager
  - EventManager
  - Hydrators

# PSR-7, Interoperability & Middleware

# It's all about HTTP

## Request:

```
{METHOD} {URI} HTTP/1.1  
Header: value1,value2  
Another-Header: value
```

Message body

## Response:

```
HTTP/1.1 {STATUS_CODE} {REASON_PHRASE}  
Header: value
```

Message body

# Current PHP

## Request:

- `$_SERVER`, `$_GET`, `$_POST`, `$_COOKIE`, `$_FILES`
- `apache_request_headers()`
- `php://input`

## Response:

- `header()`
- `echo` (& `ob_*` family)

# PSR-7

It's just some interfaces

- RequestInterface (& ServerRequestInterface)
- ResponseInterface
- UriInterface
- UploadedFileInterface

A quick aside

# Key feature 1: Immutability

Request, Response, Uri & UploadFile are *immutable*

```
$uri = new Uri('https://api.joind.in/v2.1/events');  
$uri2 = $uri->withQuery('?filter=upcoming');
```

```
$request = (new Request())  
    ->withMethod('GET')  
    ->withUri($uri2)  
    ->withHeader('Accept', 'application/json')  
    ->withHeader('Authorization', 'Bearer 0873418d');
```

# Key feature 2: Streams

Message bodies are *streams*

```
$body = new Stream();  
$body->write('<p>Hello');  
$body->write('World</p>');  
  
$response = (new Response())  
    ->withStatus(200, 'OK')  
    ->withHeader('Content-Type', 'application/header')  
    ->withBody($body);
```



# Diactoros

ZF's PSR-7 implementation

# Diactoros

- Complete PSR-7 implementation

# Diactoros

- Complete PSR-7 implementation
- Specialised Responses: JSON, Empty & Redirect

# Diactoros

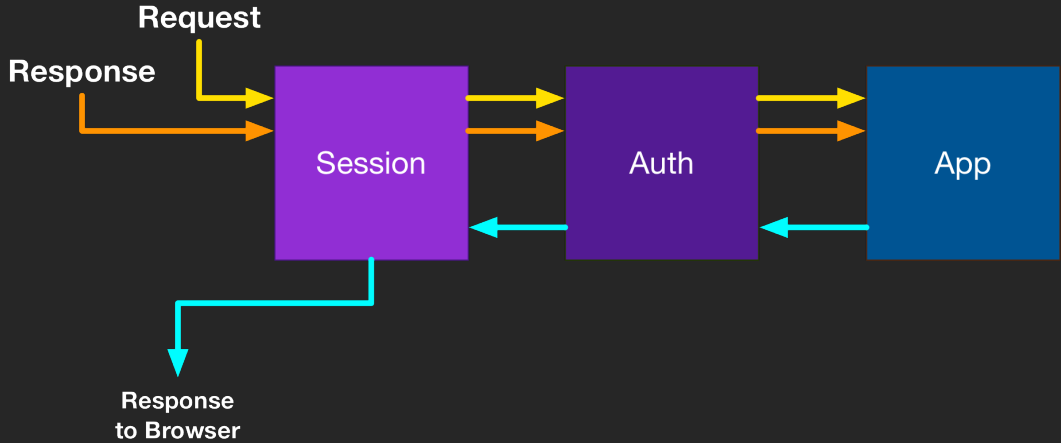
- Complete PSR-7 implementation
- Specialised Responses: JSON, Empty & Redirect
- Used by Symfony for their PSR-7 bridge

# Diactoros

- Complete PSR-7 implementation
- Specialised Responses: JSON, Empty & Redirect
- Used by Symfony for their PSR-7 bridge
- zend-psr7bridge: ZF3's PSR-7 to Zend\Htt p bridge

# Middleware

# Middleware



# Middleware

```
function ($request, $response, callable $next = null)
{
    // do something before

    // call through to next middleware
    if ($next) {
        $response = $next($request, $response);
    }

    // do something with $response after

    return $response;
}
```



# Writing middleware

Pattern:

- Optionally modify the received request and response
- Optionally invoke the next middleware
  - Optionally modify the returned response
- Return the response to the previous middleware.

# Stratigility

ZF's Middleware implementation

# Stratigility

- Dispatches a stack of middleware

# Stratigility

- Dispatches a stack of middleware
- Middleware format:
  - Any callable
  - `Zend\Stratigility\MiddlewareInterface`

```
public function __invoke(  
    ServerRequestInterface $request,  
    ResponseInterface $response,  
    callable $out = null  
) : ResponseInterface;
```

# ErrorMiddleware

Pass error as third parameter to \$next:

```
return $next($request, $response, $error);
```

# ErrorMiddleware

Pass error as third parameter to \$next:

```
return $next($request, $response, $error);
```

Handle like this:

```
function ($error,  
    ServerRequestInterface $request,  
    ResponseInterface $response,  
    callable $out  
);
```

or `Zend\Stratigility\ErrorMiddlewareInterface`

# Path segregation:

```
use Zend\Stratigility\MiddlewarePipe();
$app = new MiddlewarePipe();
$app->pipe($mw1); // always evaluate
$app->pipe('/blog', $blogMw); // only if path matches
$app->pipe('/contact', $contactMw);
$app->pipe($outputMw);

$server = Server::createServer($app, ...);
$server->listen();
```

# Nesting Middleware

Compose middleware together based on path:

```
$blog = new MiddlewarePipe();  
$blog->pipe('/post', $postMw);  
$blog->pipe('/feed', $rssMw);  
$blog->pipe('/', $listMw);
```

```
$app = new MiddlewarePipe();  
$app->pipe('/blog', $blog);
```



# Middleware wrappers

```
$app->pipe('/', $homepage);           // Static HTML
$app->pipe('/customer', $zf2Middleware); // ZF2
$app->pipe('/products', $zf1Middleware); // ZF1
$app->pipe('/api', $apigility);        // Apigility
$app->pipe('/user', $userMiddleware);   // 3rd party
```

What about routing?  
(& DI container, etc...)

# Expressive

## ZF's micro framework

# Expressive

- Builds on Diactoros & Stratigility - Middleware!

# Expressive

- Builds on Diactoros & Stratigility - Middleware!
- Choose your own components
  - Router

# Expressive

- Builds on Diactoros & Stratigility - Middleware!
- Choose your own components
  - Router
  - DI container

# Expressive

- Builds on Diactoros & Stratigility - Middleware!
- Choose your own components
  - Router
  - DI container
  - Templating

# Expressive

- Builds on Diactoros & Stratigility - Middleware!
- Choose your own components
  - Router
  - DI container
  - Templating
- Error handling



# Agnostic

Router:

- Aura.Router, FastRoute or Zend\Mvc\Router

DI Container:

- Pimple, Aura.Di, Zend\ServiceManager or container-interop

Template:

- Plates, Twig or Zend\View

# Hello world

```
use Zend\Expressive\AppFactory;

$app = AppFactory::create();

$app->get(
    '/hello/{name}',
    function ($request, $response, $next) {
        $name = htmlentities($request->getAttribute('name'));
        $response->write("<p>Hello, $name!</p>");
        return $next($request, $response);
    }
);

$app->run();
```

# Middleware pipes

```
$app->pipe($sessionMiddleware);  
$app->pipe($authMiddleware);  
$app->get('/', $homepageMiddleware);  
  
$app->run();
```

# Named routes

3rd parameter:

```
$app->get('/books/{id}', $getBookAction, 'book');
```

Build URI:

```
$url = $router->generateUri('edit', ['id' => 1]);
```

# Views

Templates are instances of  
Zend\Expressive\Template\TemplateInterface

```
$html = $templates->render('book/detail', [  
    'layout' => 'master',  
    'book' => $bookEntity,  
]);  
  
return new HtmlResponse($html);
```

# Why Expressive?

- Performance

# Why Expressive?

- Performance
- Developer experience

# Why Expressive?

- Performance
- Developer experience
- Reusable middleware



This is the ZF3 era

# The ZF3 era

- Separate components
- ZF2 MVC with performance improvements
- Stratigility PSR-7 middleware foundation
- Expressive micro framework

# Questions?

<https://joind.in/15429>

Rob Allen - <http://akrabat.com> - @akrabat

# Thank you!

<https://joind.in/15429>

Rob Allen - <http://akrabat.com> - @akrabat