MIDDLEWARE WEB APIS IN PHP 7.X

Jan Burkl
Solution Consulting Manager
Rogue Wave Software

php developer day 2017, Dresden, 22nd September 2017


PHP

PHP

- PHP: Hypertext Preprocessor
- The most popular server-side language: PHP is used by **82.6%** of all the websites (source: w3techs.com)
- Used by **Facebook**, **Wikipedia**, **Yahoo**, **Etsy**, **Flickr**, **Digg**, etc
- 22 years of usage, since 1995
- Full **OOP** support since PHP 5

PHP 7

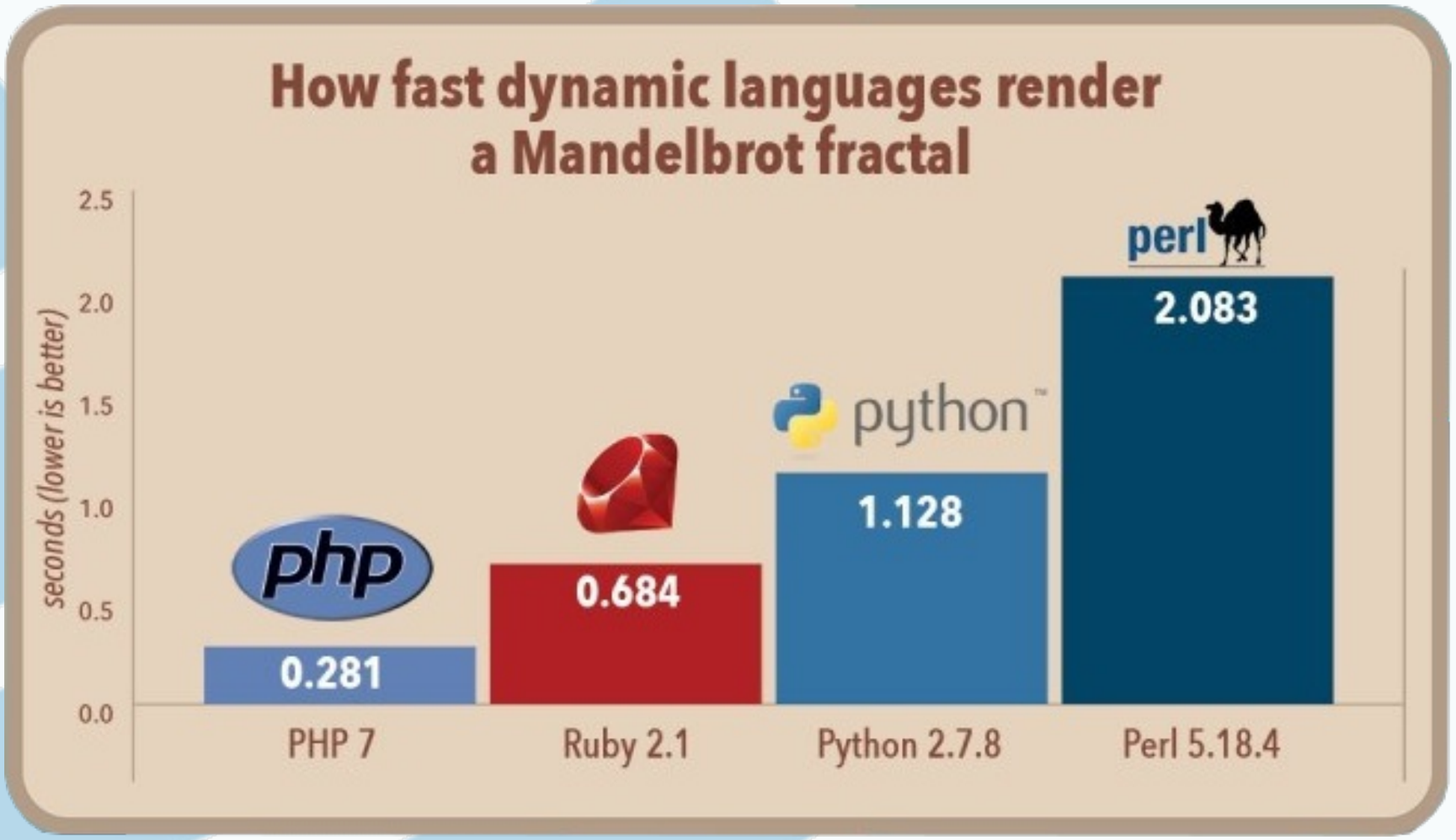
Released: 3 December 2015

Previous major was PHP 5, 13 July 2004

Skipped PHP 6: Unicode failure

Last release is 7.1.8 (3 Aug 2017)

PHP 7 PERFORMANCE



PHP 7 is also faster than Python 3!

BENCHMARK

```php
$a = [];
for ($i = 0; $i < 1000000; $i++) {
  $a[$i] = ["hello"];
}
echo memory_get_usage(true);
```

	PHP 5.6	PHP 7
Memory Usage	428 MB	33 MB
Execution time	0.49 sec	0.06 sec

MOVING TO PHP 7

- **Badoo** saved one million dollars switching to PHP 7 (source)
- **Tumblr** reduced the latency and CPU load by half moving to PHP 7 (source)
- **Dailymotion** handles twice more traffic with same infrastructure switching to PHP 7 (source)

PHP 7 IS NOT ONLY FAST!

- Return and Scalar Type Declarations
- Improved Exception hierarchy
- Many fatal errors converted to Exceptions
- Secure random number generator
- Authenticated encryption AEAD (PHP 7.1+)
- Nullable types (PHP 7.1+)
- and more!

WEB APIS IN PHP 7

HTTP IN/OUT

EXAMPLE

Request:

```
GET /api/version
```

Response:

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 17
Content-Type: application/json

{
 "version": "1.0"
}
```

MIDDLEWARE

A function that gets a request and generates a response

```
use Psr\Http\Message\ServerRequestInterface as Request;
use Interop\Http\ServerMiddleware\DelegateInterface;

function (Request $request, DelegateInterface $next)
{
    // doing something with $request...
    // for instance calling the delegate middleware $next
    $response = $next->process($request);
    // manipulate the $response
    return $response;
}
```

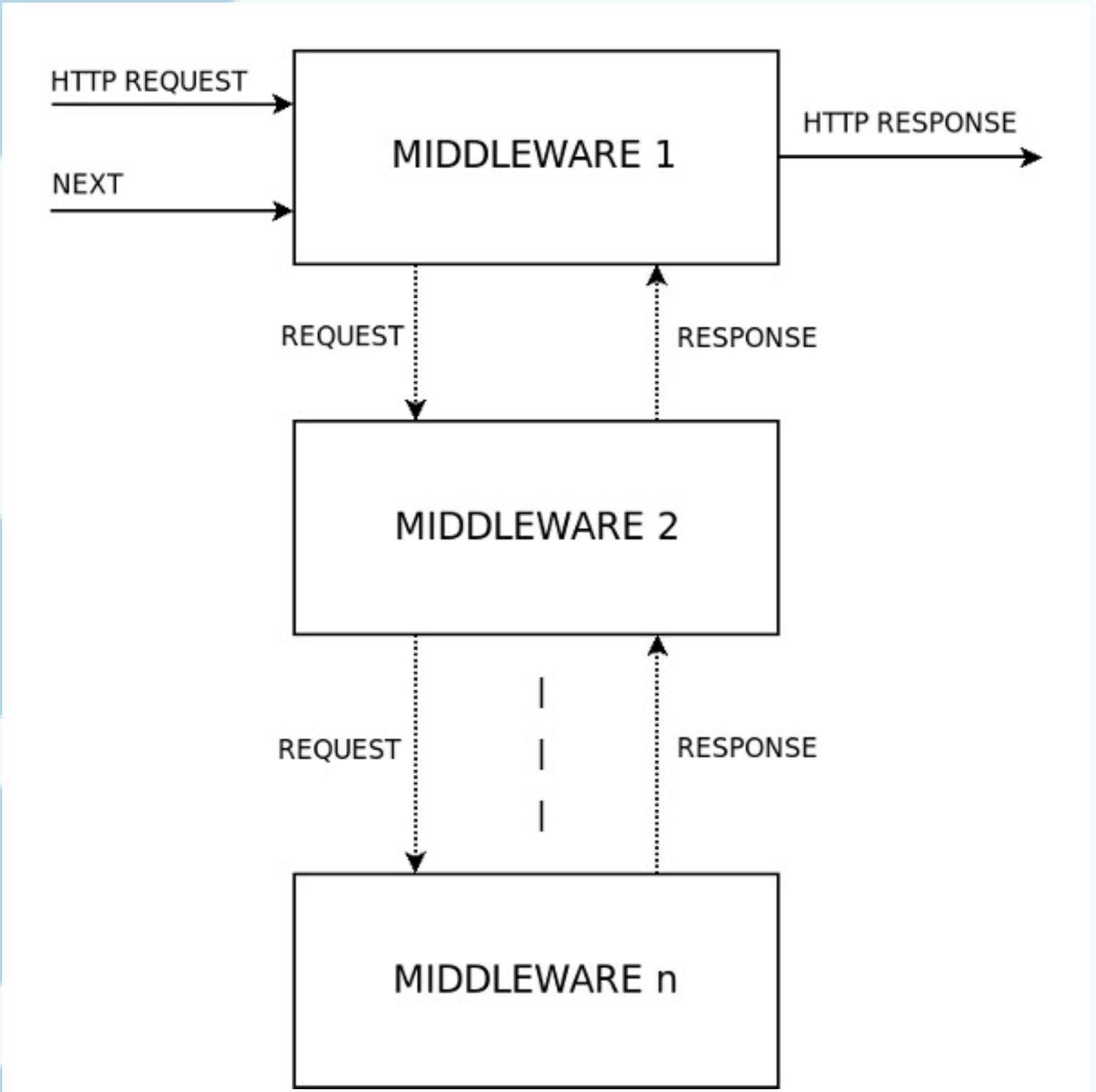
DELEGATEINTERFACE

```php
namespace Interop\Http\ServerMiddleware;

use Psr\Http\Message\ResponseInterface;
use Psr\Http\Message\ServerRequestInterface;

interface DelegateInterface
{
    /**
     * @return ResponseInterface;
     */
    public function process(ServerRequestInterface $reque
}
```

DelegateInterface is part of PSR-15 HTTP Middleware proposal


HTTP REQUEST
NEXT
MIDDLEWARE 1
HTTP RESPONSE
REQUEST
RESPONSE
MIDDLEWARE 2
REQUEST
RESPONSE
MIDDLEWARE n

EXPRESSIVE 2.0

The PHP framework for Middleware applications

- PSR-7 HTTP Message support (using zend-diactoros)
- Support of *lambda middleware* (PSR-15) and *double pass* ($request, $response, $next)
- Piping workflow (using zend-stratigility)
- Features: routing, dependency injection, templating, error handling
- Last release 2.0.3, 28th March 2017

INSTALLATION

You can install Expressive 2.0 using composer:

```
composer create-project \
  zendframework/zend-expressive-skeleton api
```

Choose the default options during the installation

DEFAULT

The skeleton has 2 URL as example: **/** and **/api/ping**

The routes are registered in **/config/routes.php**

The middleware actions are stored in **/src/App/Action**

ROUTES

```
$app->get('/', App\Action\HomePageAction::class, 'home')
$app->get('/api/ping', App\Action\PingAction::class, 'api.pi
```

/config/routes.php

API MIDDLEWARE

```php
namespace App\Action;

use Interop\Http\ServerMiddleware\DelegateInterface;
use Interop\Http\ServerMiddleware\MiddlewareInterface;
use Zend\Diactoros\Response\JsonResponse;
use Psr\Http\Message\ServerRequestInterface;

class PingAction implements MiddlewareInterface
{
    public function process(
        ServerRequestInterface $request,
        DelegateInterface $delegate
```

/src/App/Action/PingAction.php

PIPELINE WORKFLOW

```
$app->pipe(ErrorHandler::class);
$app->pipe(ServerUrlMiddleware::class);

$app->pipeRoutingMiddleware();

$app->pipe(ImplicitHeadMiddleware::class);
$app->pipe(ImplicitOptionsMiddleware::class);
$app->pipe(UrlHelperMiddleware::class);

$app->pipeDispatchMiddleware();
$app->pipe(NotFoundHandler::class);
```

/config/pipeline.php

SERVICE CONTAINER

```php
use Zend\ServiceManager\Config;
use Zend\ServiceManager\ServiceManager;

$config = require __DIR__ . '/config.php';
$container = new ServiceManager();
$config = new Config($config['dependencies']);
$config->configureServiceManager($container);
$container->setService('config', $config);

return $container;
```

/config/container.php

THE EXPRESSIVE APP

```php
chdir(dirname(__DIR__));
require 'vendor/autoload.php';

call_user_func(function () {
    $container = require 'config/container.php';
    $app = $container->get(\Zend\Expressive\Application
    require 'config/pipeline.php';
    require 'config/routes.php';

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

/public/index.php

ROUTE A REST API

```
$app->route('/api/users[/{user-id}]', [
    Authentication\AuthenticationMiddleware::class,
    Authorization\AuthorizationMiddleware::class,
    Api\Action\UserAction::class
], ['GET', 'POST', 'PATCH', 'DELETE'], 'api.users');

// or route each HTTP method
$app->get('/api/users[/{user-id}]', ..., 'api.users.get');
$app->post('/api/users', ..., 'api.users.post');
$app->patch('/api/users/{user-id}', ..., 'api.users.patch');
$app->delete('/api/users/{user-id}', ..., 'api.users.delete');
```

REST DISPATCH TRAIT

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

trait RestDispatchTrait
{
    public function process(
        ServerRequestInterface $request,
        DelegateInterface $delegate
    ) {
        $method = strtolower($request->getMethod());
        if (method_exists($this, $method)) {
            return $this->$method($request);
```

REST MIDDLEWARE

```php
class UserAction implements MiddlewareInterface
{
    use RestDispatchTrait;

    public function get(ServerRequestInterface $request)
    {
        $id = $request->getAttribute('user-id', false);
        $data = (false === $id) ? /* all users */ : /* user id *
        return new JsonResponse($data);
    }

    public function post(ServerRequestInterface $request)
```

Api\Action\UserAction.php

DANKE SCHÖN

Slides: http://5square.github.io/talks/

More info: https://framework.zend.com/blog

Contact me: jan.burkl [at] roguewave.com

Follow me: @janatzend

Credits: @ezimuel