PHP
fwdays

Deep dive into Symfony4 internals

Tobias Nyholm, @tobiasnyholm

Why?

Tobias Nyholm

- Full stack unicorn on Happyr.com
- Certified Symfony developer
- Symfony core member
- PHP-Stockholm
- Open source

Open source

Assert

Neo4j

FriendsOfApi/boilerplate

Guzzle

Buzz

Backup-manager/symfony

nyholm/effective-interest-rate

Puli

Swap

LinkedIn API client

NSA

php-http/discovery

PHP-cache

PHP-Translation

PSR7

CacheBundle

league/geotools

Mailgun

KNP Github API

happyr/normal-distribution-bundle

HTTPlug

SymfonyBundleTest

Stampie

php-http/httplug-bundle

MailgunBundle

php-http/multipart-stream

SimpleBus integrations

PHP-Geocoder

BazingaGeocoderBundle

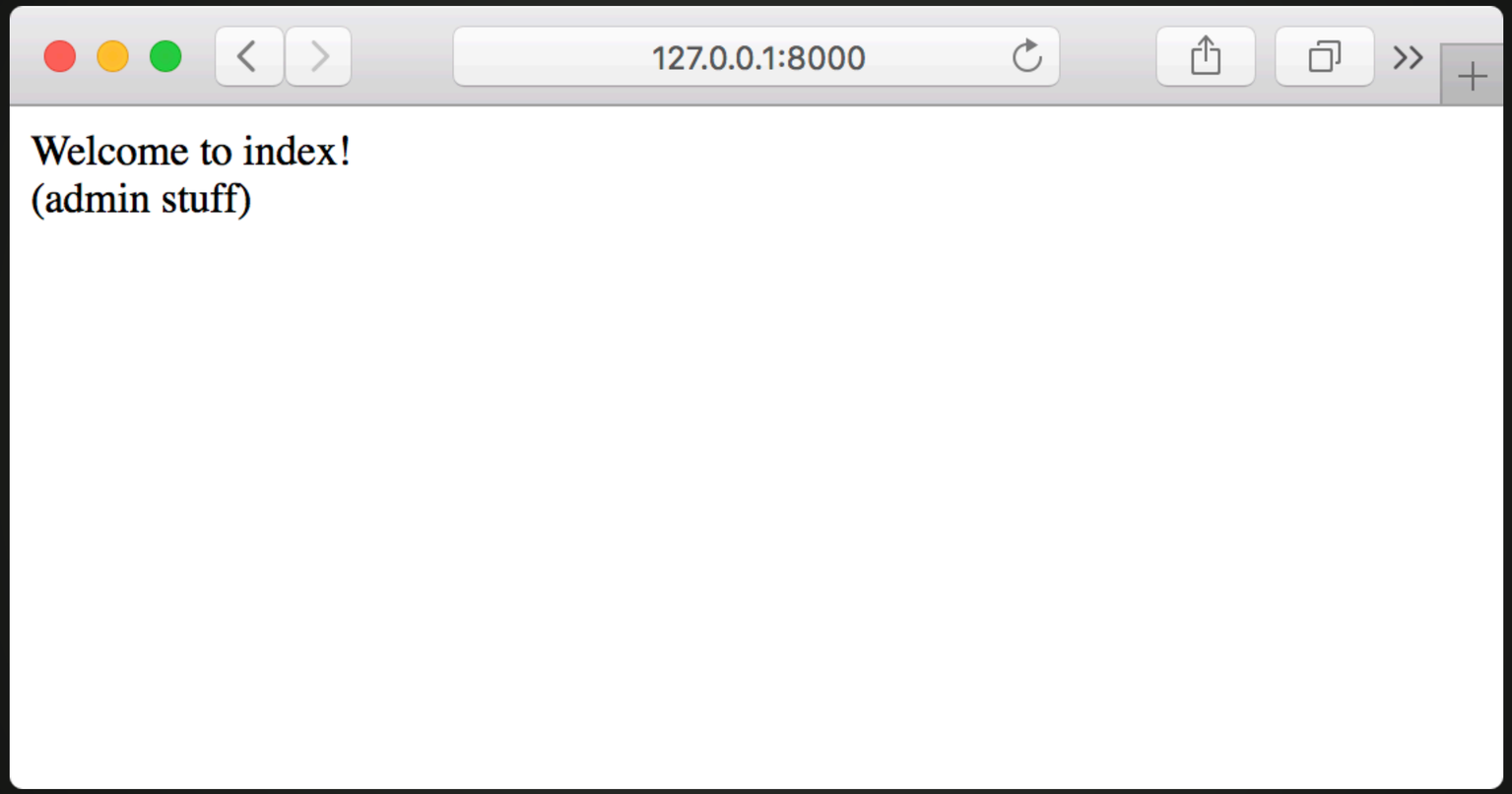
PSR HTTP clients

Building your own framework

Just a bunch of files

```php
<?php // index.php

if (isset($_GET['page']) && $_GET['page'] === 'foo') {
    echo "Foo page <br>";
} else {
    echo "Welcome to index! <br>";
}

if ($_SERVER['REMOTE_ADDR'] === '127.0.0.1') {
    echo "(admin stuff)";
}
```

Just a bunch of files



Just a bunch of files

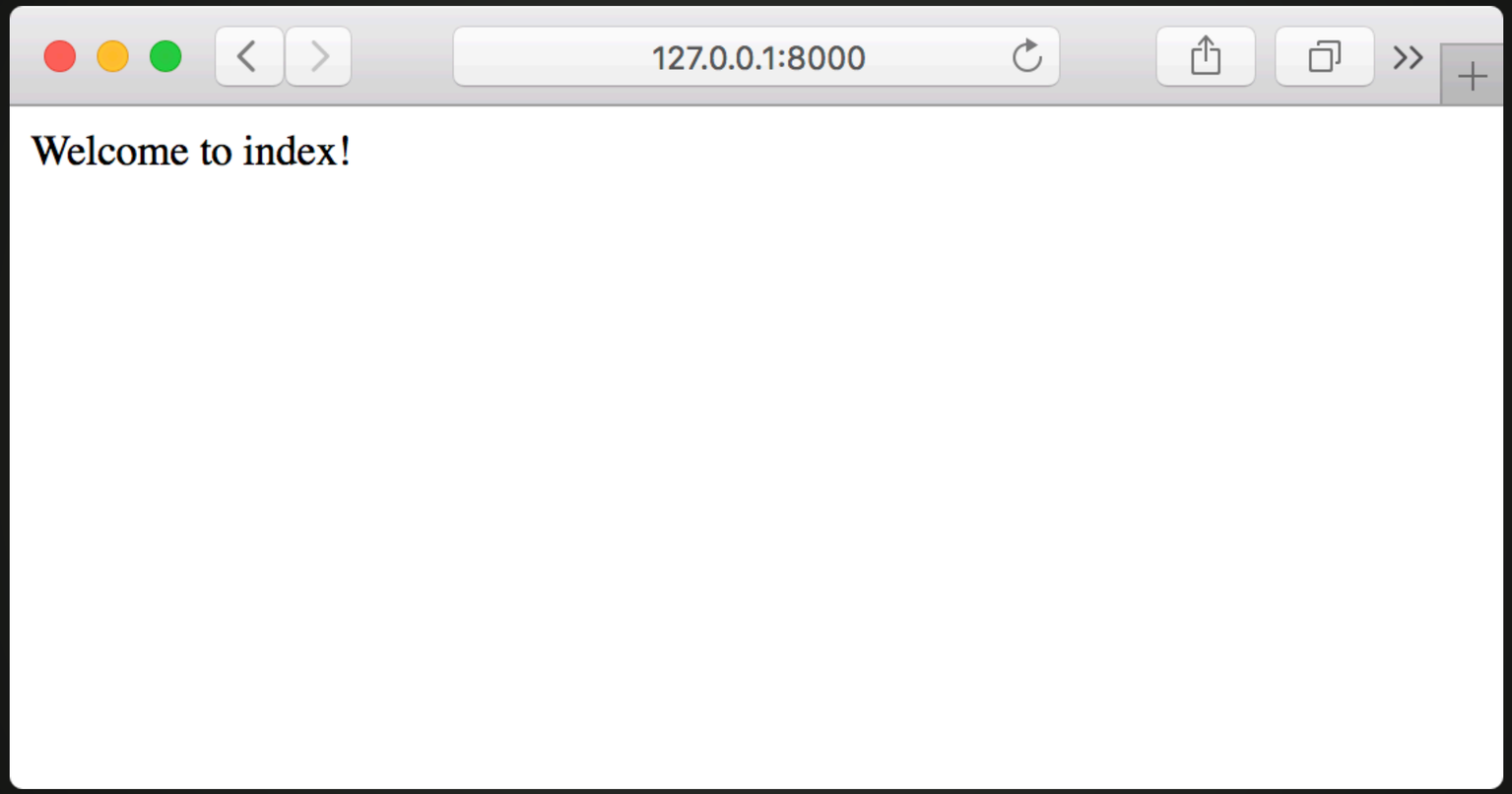
```php
<?php // index.php

if (isset($_GET['page']) && $_GET['page'] === 'foo') {
    echo "Foo page <br>";
} else {
    echo "Welcome to index! <br>";
}

if ($_SERVER['REMOTE_ADDR'] === '127.0.0.1') {
    echo "(admin stuff)";
}
```

HTTP objects

HTTP objects

```php
<?php // index.php

use Nyholm\Psr7\Factory\ServerRequestFactory;
use Nyholm\Psr7\Response;

require __DIR__.'/vendor/autoload.php';

$request = (new ServerRequestFactory())->createServerRequestFromGlobals();

$query = $request->getQueryParams();
if (isset($query['page']) && $query['page'] === 'foo') {
    $response = new Response(200, [], 'Foo page');
} else {
    $response = new Response(200, [], 'Welcome to index!');
}

// Send response
echo $response->getBody();
```

HTTP objects



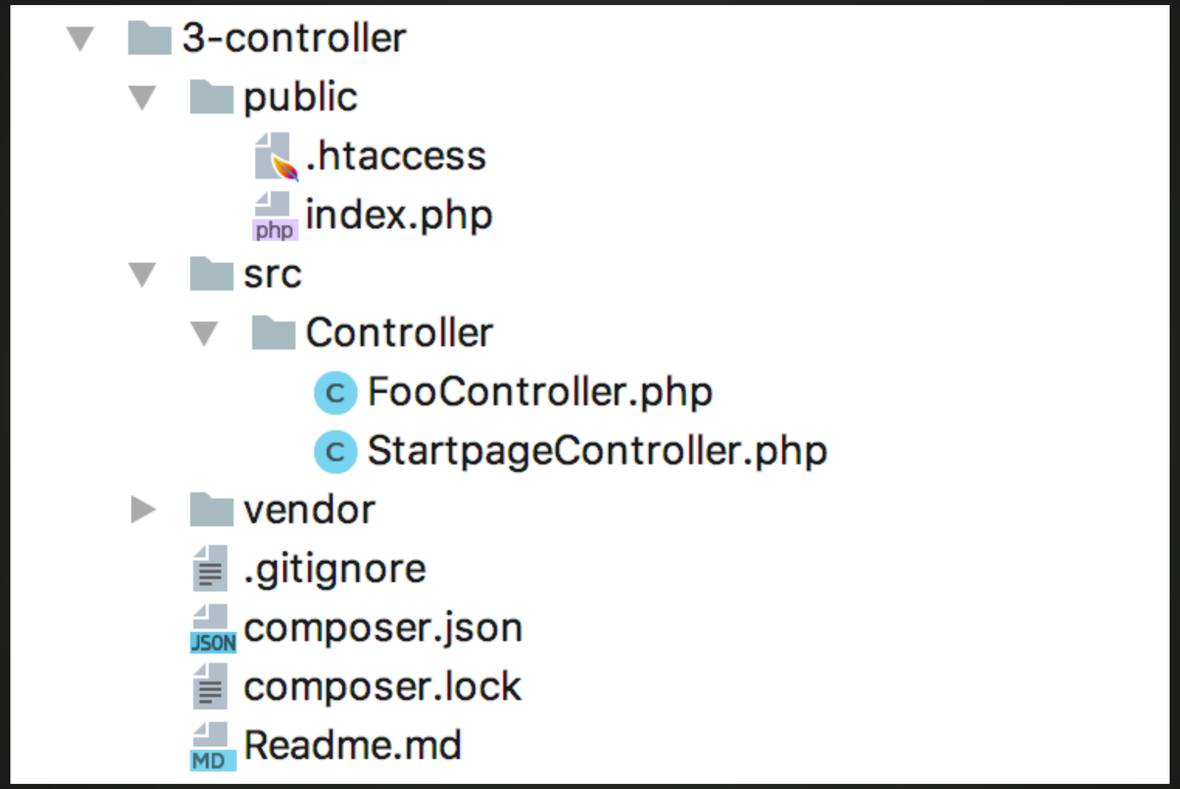
HTTP objects

HTTP objects

```php
<?php // index.php

use Nyholm\Psr7\Factory\ServerRequestFactory;
use Nyholm\Psr7\Response;

require __DIR__.'/vendor/autoload.php';

$request = (new ServerRequestFactory())->createServerRequestFromGlobals();

$query = $request->getQueryParams();
if (isset($query['page']) && $query['page'] === 'foo') {
    $response = new Response(200, [], 'Foo page');
} else {
    $response = new Response(200, [], 'Welcome to index!');
}

// Send response
echo $response->getBody();
```

Controllers

Controllers



Controllers

```
DirectoryIndex index.php

<IfModule mod_rewrite.c>
    RewriteEngine On

        RewriteCond %{REQUEST_URI}::$1 ^(/.+)/(.*)::\2$
        RewriteRule ^(.*) - [E=BASE:%1]

        RewriteCond %{ENV:REDIRECT_STATUS} ^$
        RewriteRule ^index\.php(/(.*)|$) %{ENV:BASE}/$2 [R=301,L]

        # If the requested filename exists, simply serve it.
        # We only want to let Apache serve files and not directories.
        RewriteCond %{REQUEST_FILENAME} -f
        RewriteRule .? - [L]

        # Prod
        RewriteRule .? %{ENV:BASE}/index.php [L]
</IfModule>
```

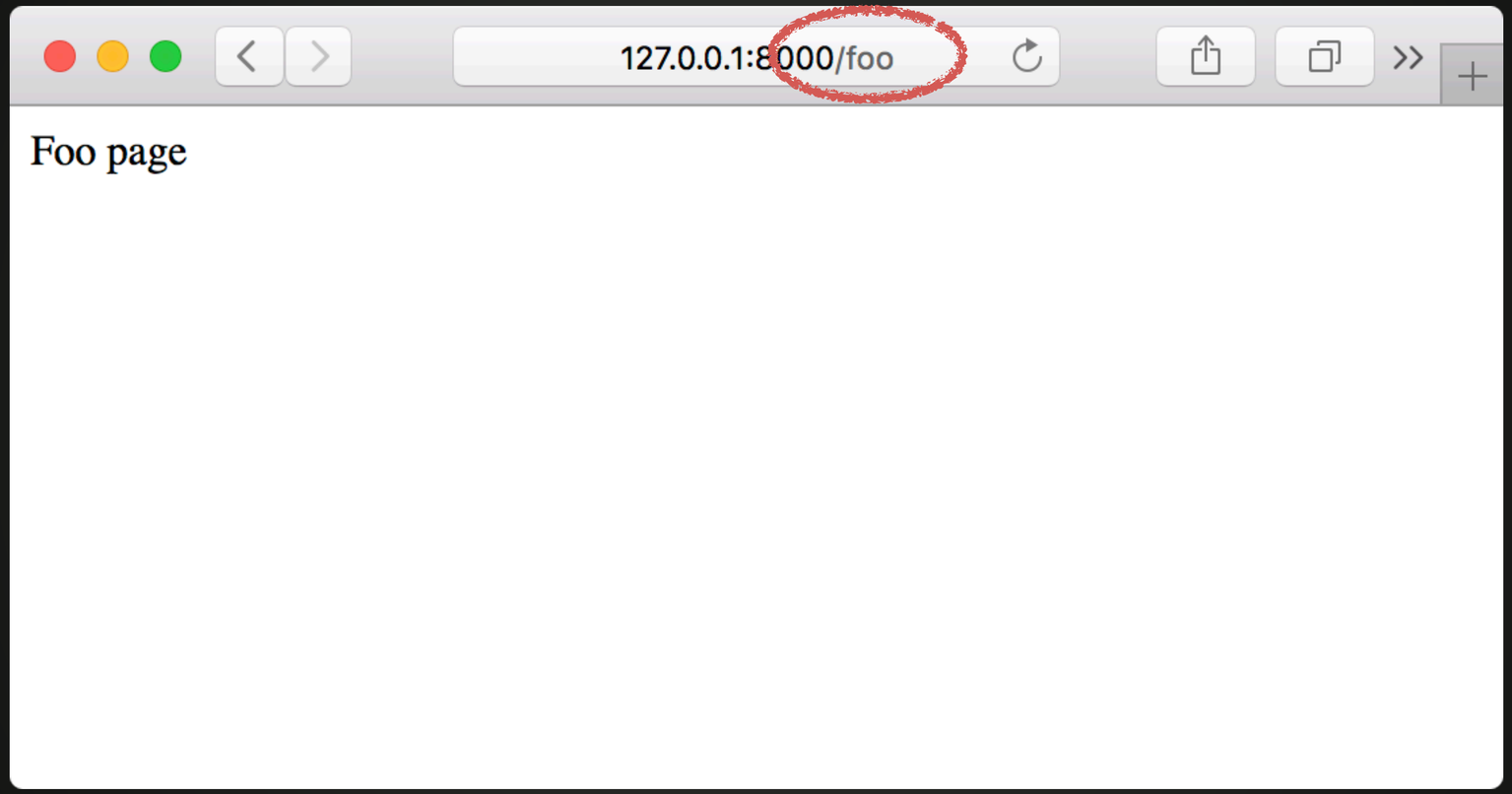
Controllers

```php
<?php // index.php

use Nyholm\Psr7\Factory\ServerRequestFactory;
use Nyholm\Psr7\Response;

require __DIR__.'/../vendor/autoload.php';

$request = (new ServerRequestFactory())->createServerRequestFromGlobals();

$uri = $request->getUri()->getPath();
if ($uri === '/') {
    $response = (new \App\Controller\StartpageController())->run($request);
} elseif ($uri === '/foo') {
    $response = (new \App\Controller\FooController())->run($request);
} else {
    $response = new Response(404, [], 'Not found');
}

// Send response
echo $response->getBody();
```

Controllers

```php
<?php

declare(strict_types=1);

namespace App\Controller;

use Nyholm\Psr7\Response;
use Psr\Http\Message\RequestInterface;

class StartpageController
{
    public function run(RequestInterface $request)
    {
        return new Response(200, [], 'Welcome to index!');
    }
}
```

Controllers



Controllers

```php
<?php // index.php

use Nyholm\Psr7\Factory\ServerRequestFactory;
use Nyholm\Psr7\Response;

require __DIR__.'/../vendor/autoload.php';

$request = (new ServerRequestFactory())->createServerRequestFromGlobals();

$uri = $request->getUri()->getPath();
if ($uri === '/') {
    $response = (new \App\Controller\StartpageController())->run($request);
} elseif ($uri === '/foo') {
    $response = (new \App\Controller\FooController())->run($request);
} else {
    $response = new Response(404, [], 'Not found');
}

// Send response
echo $response->getBody();
```

Event loop

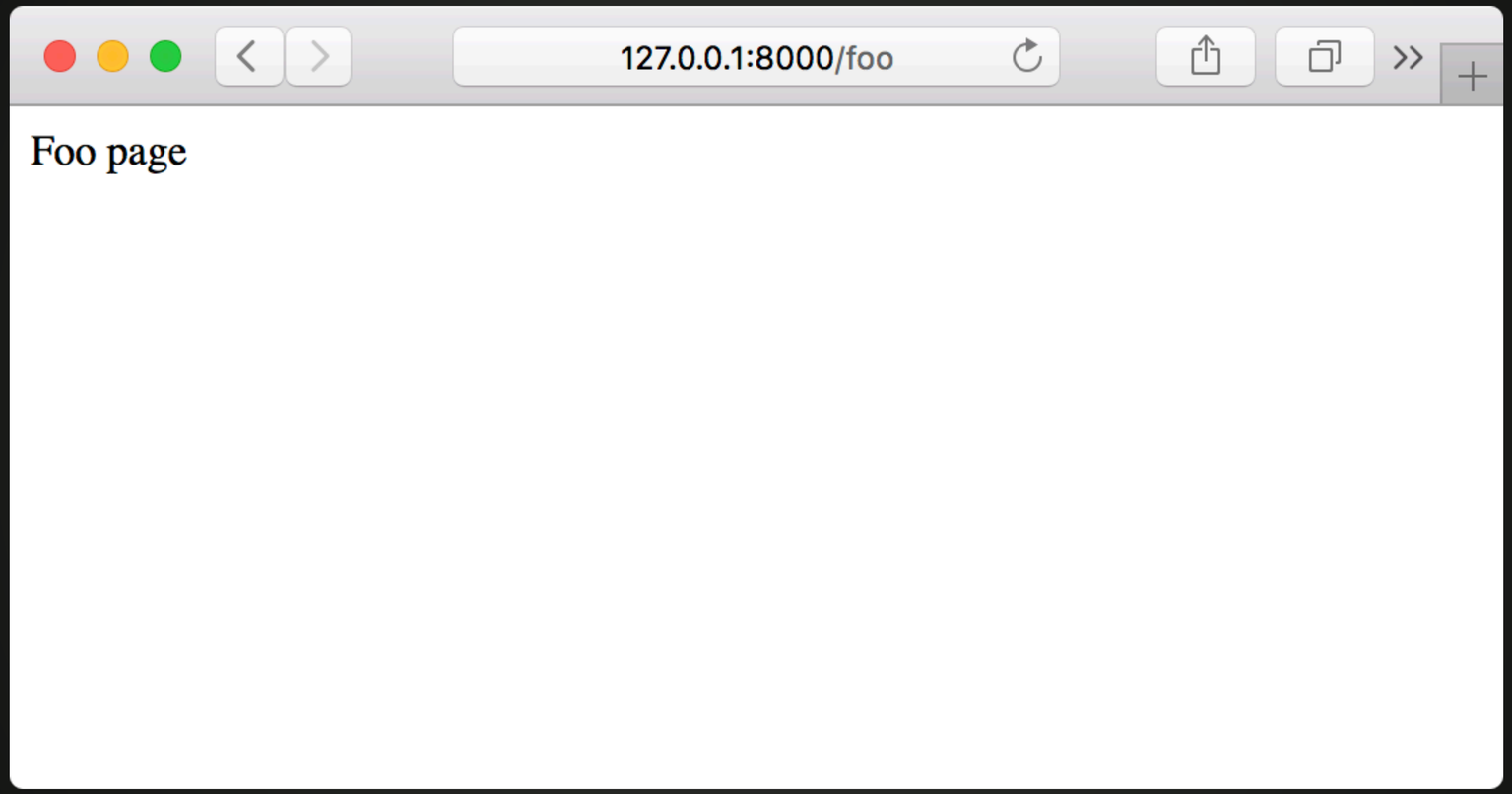
Event loop

```php
<?php // index.php

use Nyholm\Psr7\Factory\ServerRequestFactory;
use Nyholm\Psr7\Response;

require __DIR__.'/../vendor/autoload.php';

$request = (new ServerRequestFactory())->createServerRequestFromGlobals();
$response = new Response();

$middlewares[] = new \App\Middleware\Router();

$runner = (new \Relay\RelayBuilder())->newInstance($middlewares);
$response = $runner($request, $response);

// Send response
echo $response->getBody();
```

@tobiasnyholm

```php
namespace App\Middleware;

use Psr\Http\Message\ResponseInterface as Response;
use Psr\Http\Message\ServerRequestInterface as Request;

class Router implements MiddlewareInterface
{
    public function __invoke(Request $request, Response $response, callable $next)
    {
        $uri = $request->getUri()->getPath();

        switch ($uri) {
            case '/':
                $response = (new \App\Controller\StartpageController())->run($request);
                break;
            case '/foo':
                $response = (new \App\Controller\FooController())->run($request);
                break;
            default:
                $response = $response->withStatus(404);
                $response->getBody()->write('Not Found');
                break;
        }

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

Event loop



Event loop

```php
<?php // index.php

use Nyholm\Psr7\Factory\ServerRequestFactory;
use Nyholm\Psr7\Response;

require __DIR__.'/../vendor/autoload.php';

$request = (new ServerRequestFactory())->createServerRequestFromGlobals();
$response = new Response();

$middlewares[] = new \App\Middleware\Router();

$runner = (new \Relay\RelayBuilder())->newInstance($middlewares);
$response = $runner($request, $response);

// Send response
echo $response->getBody();
```

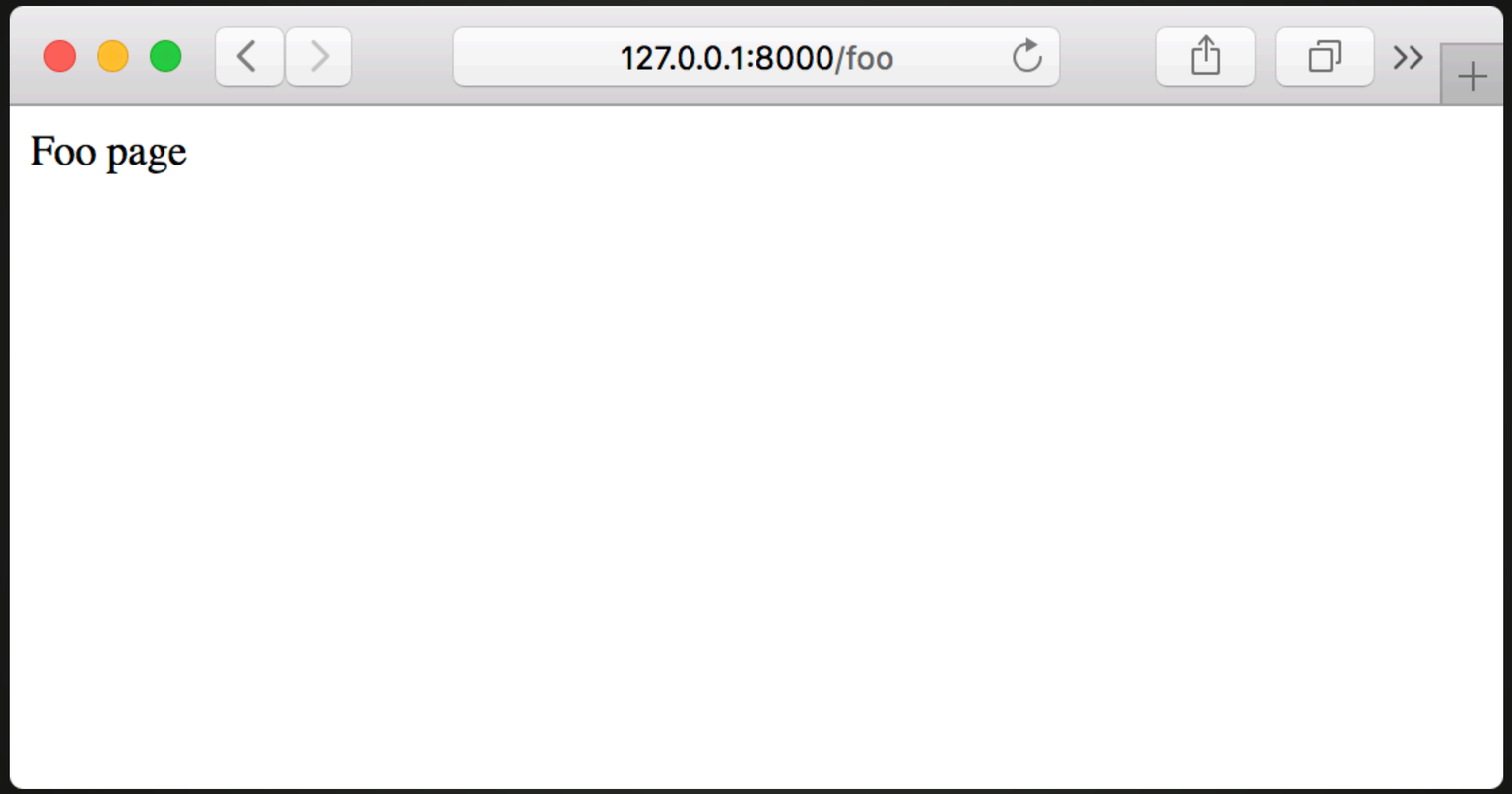
Cache

```php
<?php // Cache.php

namespace App\Middleware;

use Cache\Adapter\Apcu\ApcuCachePool;
use Psr\Cache\CacheItemPoolInterface;
use Psr\Http\Message\ResponseInterface as Response;
use Psr\Http\Message\ServerRequestInterface as Request;

class Cache implements MiddlewareInterface
{
    /*** @var CacheItemPoolInterface */
    private $cache;

    /** @var int */
    private $ttl;

    public function __construct()
    {
        $this->cache = new ApcuCachePool();
        $this->ttl = 300;
    }

    public function __invoke(Request $request, Response $response, callable $next)
    {
        $uri = $request->getUri();
        $cacheKey = 'url'.sha1($uri->getPath().'?'.$uri->getQuery());
        $cacheItem = $this->cache->getItem($cacheKey);
```

Cache

```php
<?php // index.php

use Nyholm\Psr7\Factory\ServerRequestFactory;
use Nyholm\Psr7\Response;

require __DIR__.'/../vendor/autoload.php';

$request = (new ServerRequestFactory())->createServerRequestFromGlobals();
$response = new Response();

$middlewares[] = new \App\Middleware\Cache();
$middlewares[] = new \App\Middleware\Router();

$runner = (new \Relay\RelayBuilder())->newInstance($middlewares);
$response = $runner($request, $response);

// Send response
echo $response->getBody();
```

Cache



Cache

```php
<?php // index.php

use Nyholm\Psr7\Factory\ServerRequestFactory;
use Nyholm\Psr7\Response;

require __DIR__.'/../vendor/autoload.php';

$request = (new ServerRequestFactory())->createServerRequestFromGlobals();
$response = new Response();

$middlewares[] = new \App\Middleware\Cache();
$middlewares[] = new \App\Middleware\Router();

$runner = (new \Relay\RelayBuilder())->newInstance($middlewares);
$response = $runner($request, $response);

// Send response
echo $response->getBody();
```

Container

Container

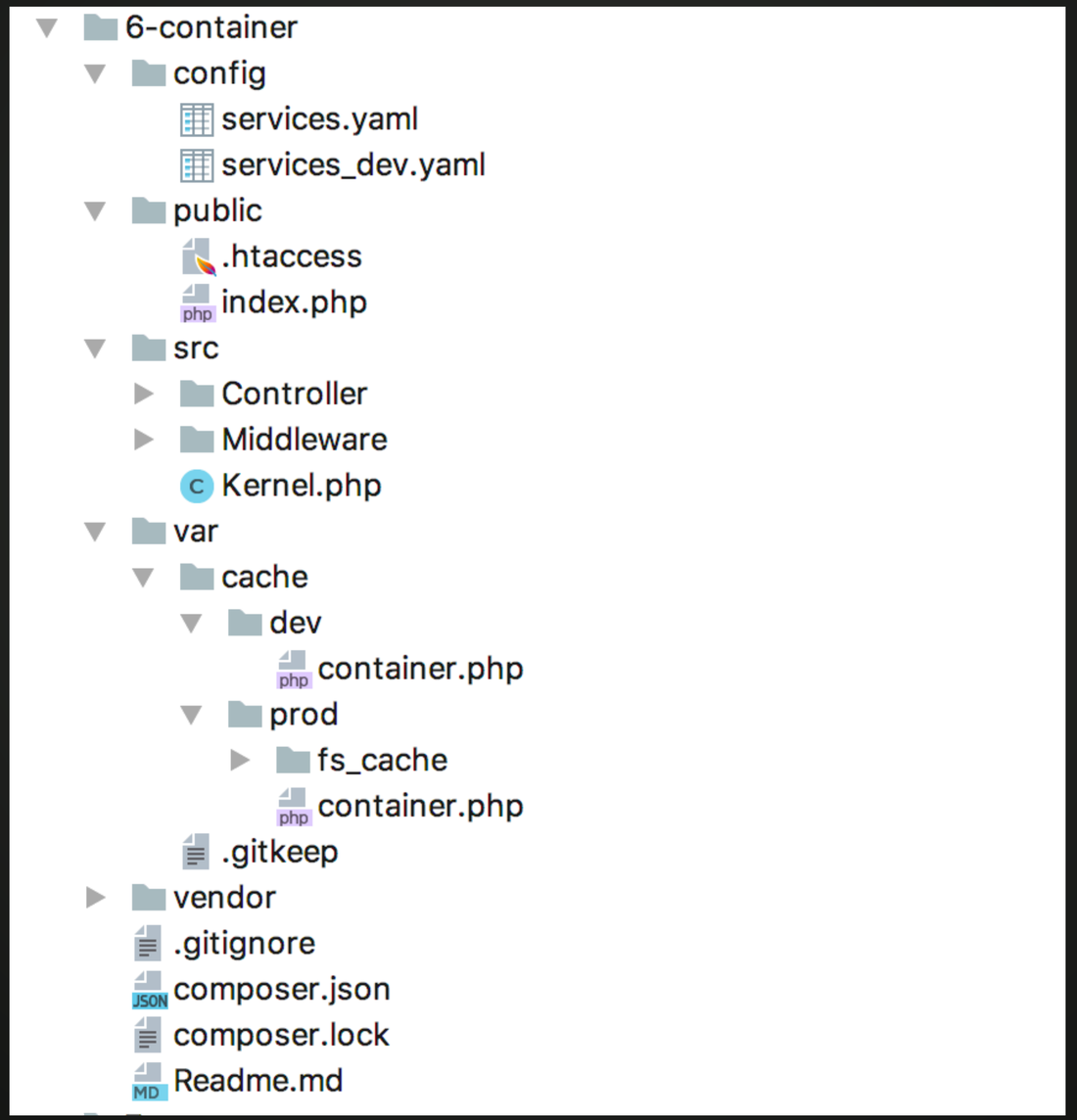
```php
<?php // index.php

use Nyholm\Psr7\Factory\ServerRequestFactory;
use Nyholm\Psr7\Response;

require __DIR__.'/../vendor/autoload.php';

$request = (new ServerRequestFactory())->createServerRequestFromGlobals();
$response = new Response();

$middlewares[] = new \App\Middleware\Cache();
$middlewares[] = new \App\Middleware\Router();

$runner = (new \Relay\RelayBuilder())->newInstance($middlewares);
$response = $runner($request, $response);

// Send response
echo $response->getBody();
```

Container

```php
<?php // index.php

use Nyholm\Psr7\Factory\ServerRequestFactory;
use Nyholm\Psr7\Response;

require __DIR__.'/../vendor/autoload.php';

$request = (new ServerRequestFactory())->createServerRequestFromGlobals();

$kernel = new \App\Kernel('dev', true);
$response = $kernel->handle($request);



// Send response
echo $response->getBody();
```

```php
<?php // Kernel.php

namespace App;

use Nyholm\Psr7\Response;
use Psr\Http\Message\RequestInterface;
use Psr\Http\Message\ResponseInterface;
use Symfony\Component\Config\Exception\FileLocatorFileNotFoundException;
use Symfony\Component\Config\FileLocator;
use Symfony\Component\DependencyInjection\Container;
use Symfony\Component\DependencyInjection\ContainerBuilder;
use Symfony\Component\DependencyInjection\Dumper\PhpDumper;
use Symfony\Component\DependencyInjection\Loader\YamlFileLoader;
class Kernel
{
    private $booted = false;
    private $debug;
    private $environment;

    /** @var Container */
    private $container;

    public function __construct(string $env, bool $debug = false)
    {
        $this->debug = $debug;
        $this->environment = $env;
    }

    /**
```

Container


6-container
config
services.yaml
services_dev.yaml
public
.htaccess
index.php
src
Controller
Middleware
Kernel.php
var
cache
dev
container.php
prod
fs_cache
container.php
.gitkeep
vendor
.gitignore
composer.json
composer.lock
Readme.md

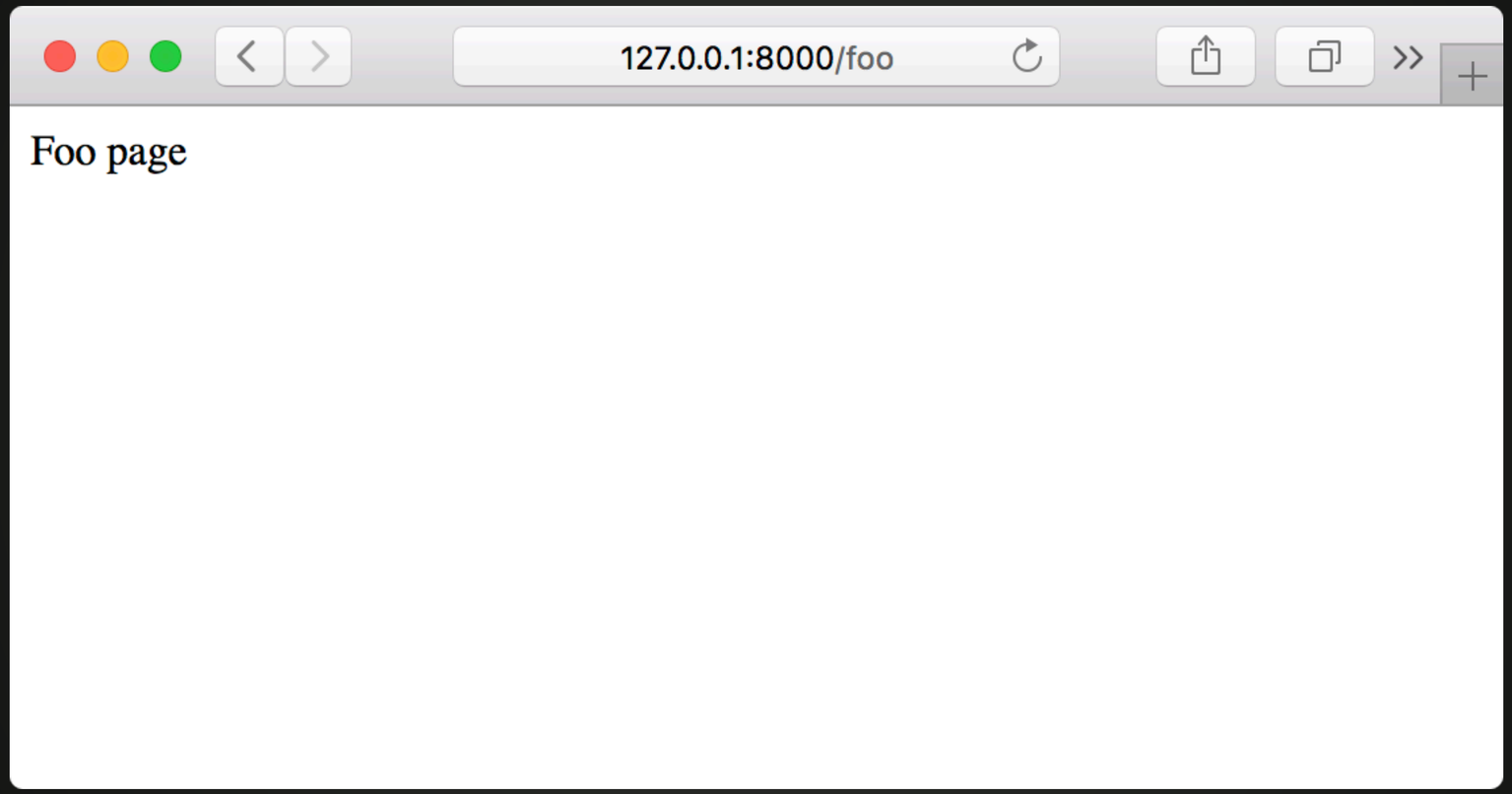
Container

```yaml
# services.yaml
services:
  cache:
    alias: cache.filesystem

  cache.void:
    class: Cache\Adapter\Void\VoidCachePool

  cache.filesystem:
    class: Cache\Adapter\Filesystem\FilesystemCachePool
    arguments: ['@flysystem.filesystem']

  flysystem.filesystem:
    class: League\Flysystem\Filesystem
    arguments: ['@flysystem.local_adapter']

  flysystem.local_adapter:
    class: League\Flysystem\Adapter\Local
    arguments: ['%kernel.project_dir%/var/cache/%kernel.environment%/fs_cache']

  middleware.cache:
    class: App\Middleware\Cache
    arguments: ['@cache']
    public: true

  controller.startpage:
    class: App\Controller\StartpageController
    public: true

  controller.foo:
    class: App\Controller\FooController
    public: true
```

```yaml
# services_dev.yaml
services:
  cache:
    alias: cache.void
```

Container

Container



Router

@tobiasnyholm

```php
<?php

namespace App\Middleware;

use Psr\Http\Message\ResponseInterface as Response;
use Psr\Http\Message\ServerRequestInterface as Request;

class Router implements MiddlewareInterface
{
    public function __invoke(Request $request, Response $response, callable $next)
    {
        $uri = $request->getUri()->getPath();

        switch ($uri) {
            case '/':
                $response = (new \App\Controller\StartpageController())->run($request);
                break;
            case '/foo':
                $response = (new \App\Controller\FooController())->run($request);
                break;
            default:
                $response = $response->withStatus(404);
                $response->getBody()->write('Not Found');
                break;
        }

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

```php
public function __invoke(Request $request, Response $response, callable $next)
{
    $uri = $request->getUri()->getPath();

    switch ($uri) {
        case '/':
            $response = (new \App\Controller\StartpageController())->run($request);
            break;
        case '/foo':
            $response = (new \App\Controller\FooController())->run($request);
            break;
        default:
            $response = $response->withStatus(404);
            $response->getBody()->write('Not Found');
            break;
    }

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

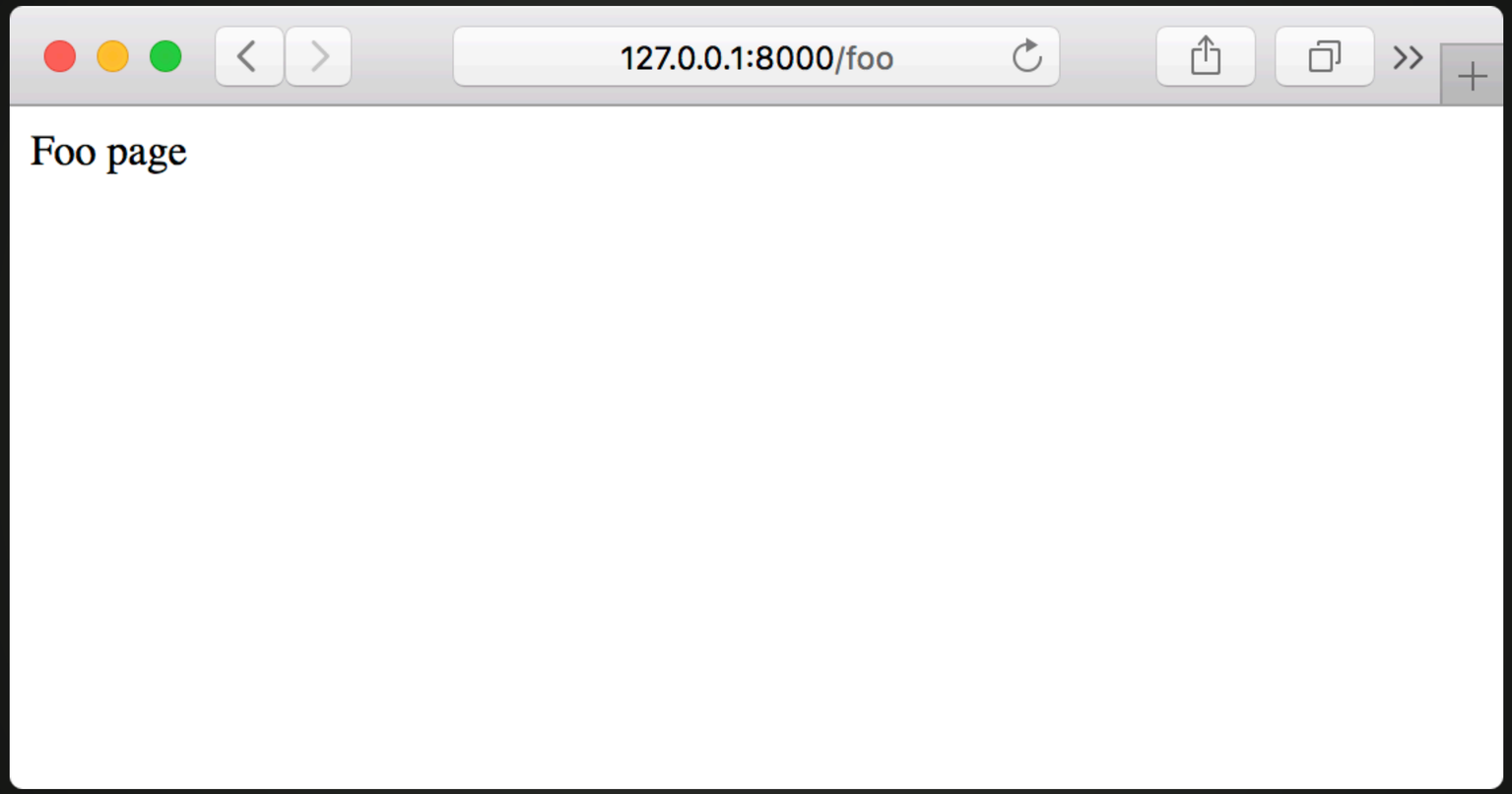
Router

/admin/account
/admin/password
/images/upload
/images/{id}/show
/images/{id}/delete
/foo
/

```php
public function __invoke(Request $request, Response $response, callable $next)
    {
        $uri = $request->getUri()->getPath();

        if (0 === strpos($uri, '/admin')) {
            if (0 === strpos($uri, '/admin/account')) {
                $response = (new \App\Controller\ManagerController())->accountAction($request);
            }
            if (0 === strpos($uri, '/admin/password')) {
                $response = (new \App\Controller\ManagerController())->passwordAction($request);
            }
        } elseif (0 === strpos($uri, '/images')) {
            if (0 === strpos($uri, '/images/upload')) {
                $response = (new \App\Controller\ImageController())->uploadAction($request);
            }
            if (preg_match('#^/images/(?P<id>[^/]++)/show#sD', $uri, $matches)) {
                $response = (new \App\Controller\ImageController())->showAction($request, $matches
            }
            if (preg_match('#^/images/(?P<id>[^/]++)/delete#sD', $uri, $matches)) {
                $response = (new \App\Controller\ImageController())->deleteAction($request, $matche
            }
        } elseif ($uri === '/') {
            $response = (new \App\Controller\StartpageController())->run($request);
        } elseif ($uri === '/foo') {
            $response = (new \App\Controller\FooController())->run($request);
        } else {
            $response = $response->withStatus(404);
            $response->getBody()->write('Not Found');
        }
```

```php
public function __invoke(Request $request, Response $response, callable $next)
{
    $uri = $request->getUri()->getPath();

    $routes = array(
        '/foo' =>  'App\\Controller\\FooController::FooAction',
        '/images/upload' => 'App\\Controller\\ImageController::uploadAction',
        '/admin/account' => 'App\\Controller\\ManagerController::accountAction',
        '/admin/password' => 'App\\Controller\\ManagerController::passwordAction',
        // More static routes
        '/' => 'App\\Controller\\StartpageController::startpageAction',
    );

    if (isset($routes[$uri])) {
        $response = call_user_func($routes[$uri], $request);

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

    $regex =
        '{^(?'
            .'|/images/([^/]++)/(?'
                .'|show(*:31)'
                .'|delete(*:44)'
                // my dynamic and complex regex
            .')'
        .')$}sD';
```

Router



Security

Security

```php
<?php

namespace App\Middleware;

use Nyholm\Psr7\Response;
use Psr\Http\Message\ResponseInterface;
use Psr\Http\Message\ServerRequestInterface;

class Security implements MiddlewareInterface
{
    public function __invoke(ServerRequestInterface $request, ResponseInterface $response, callabl
    {
        $uri = $request->getUri()->getPath();
        $ip = $request->getServerParams()['REMOTE_ADDR'];

        if ($ip !== '127.0.0.1' && $uri === '/admin') {
            return new Response(403, [], 'Forbidden');
        }

        if ($uri === '/images/4711/delete') {
            if (true /* user is not "bob" */) {
                return new Response(403, [], 'Forbidden');
            }
        }

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

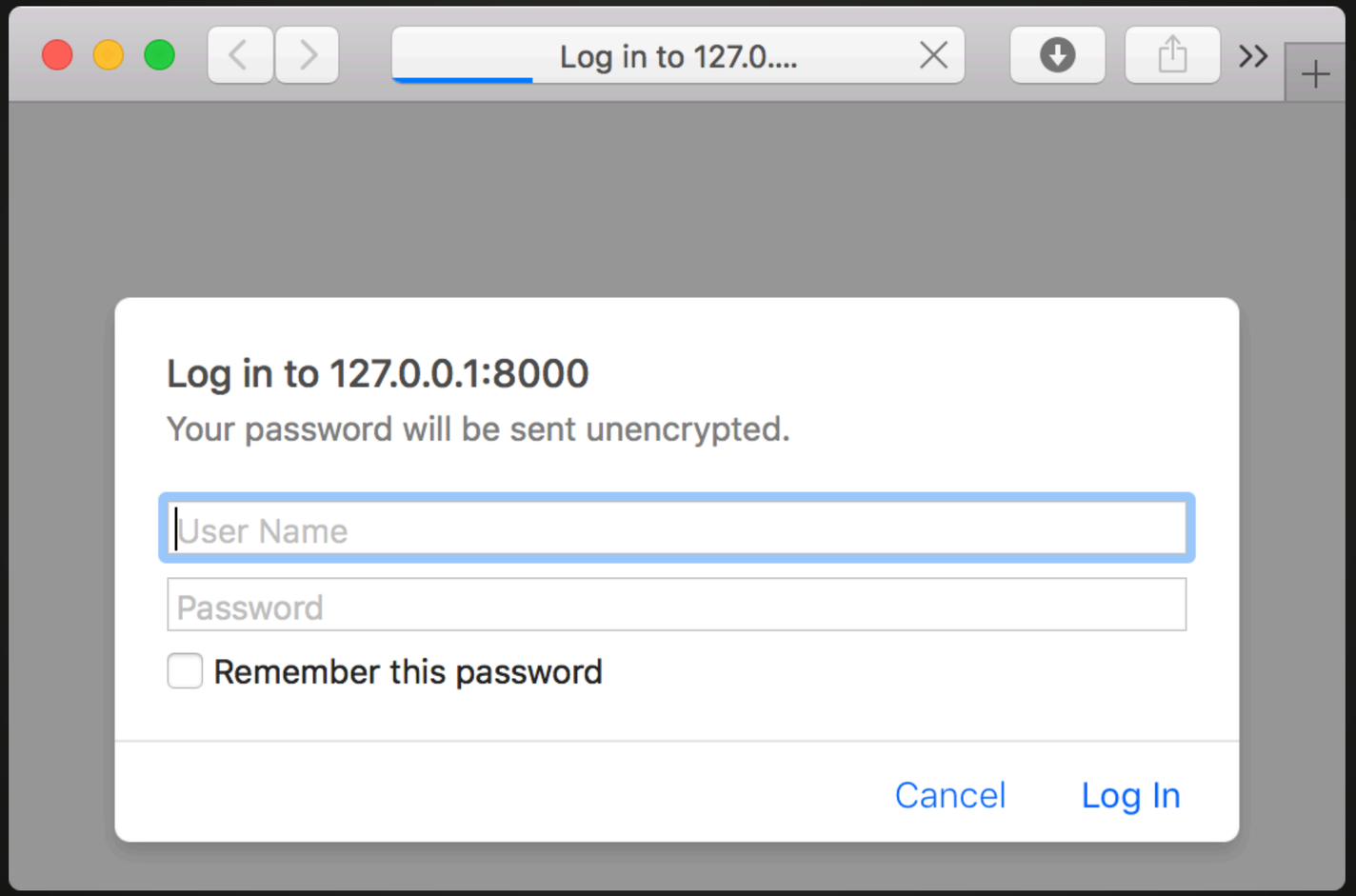
```php
<?php

namespace App\Middleware;

use App\Security\TokenStorage;
use Nyholm\Psr7\Response;
use Psr\Http\Message\ResponseInterface ;
use Psr\Http\Message\ServerRequestInterface;

class Authentication implements MiddlewareInterface
{
    private $tokenStorage;

    /**
     *
     * @param $tokenStorage
     */
    public function __construct(TokenStorage $tokenStorage)
    {
        $this->tokenStorage = $tokenStorage;
    }

    public function __invoke(ServerRequestInterface $request, ResponseInterface $response, callabl
    {
        $uri = $request->getUri()->getPath();
        $auth = $request->getServerParams()['PHP_AUTH_USER']??'';
        $pass = $request->getServerParams()['PHP_AUTH_PW']??'';
```

Security

```php
class Kernel
{
    // …
    /**
     * Handle a Request and turn it in to a response.
     */
    public function handle(RequestInterface $request): ResponseInterface
    {
        $this->boot();

        $middlewares[] = $this->container->get(‘middleware.auth');
        $middlewares[] = $this->container->get('middleware.cache');
        $middlewares[] = new \App\Middleware\Router();

        $runner = (new \Relay\RelayBuilder())->newInstance($middlewares);

        return $runner($request, new Response());
    }
    // …
}
```

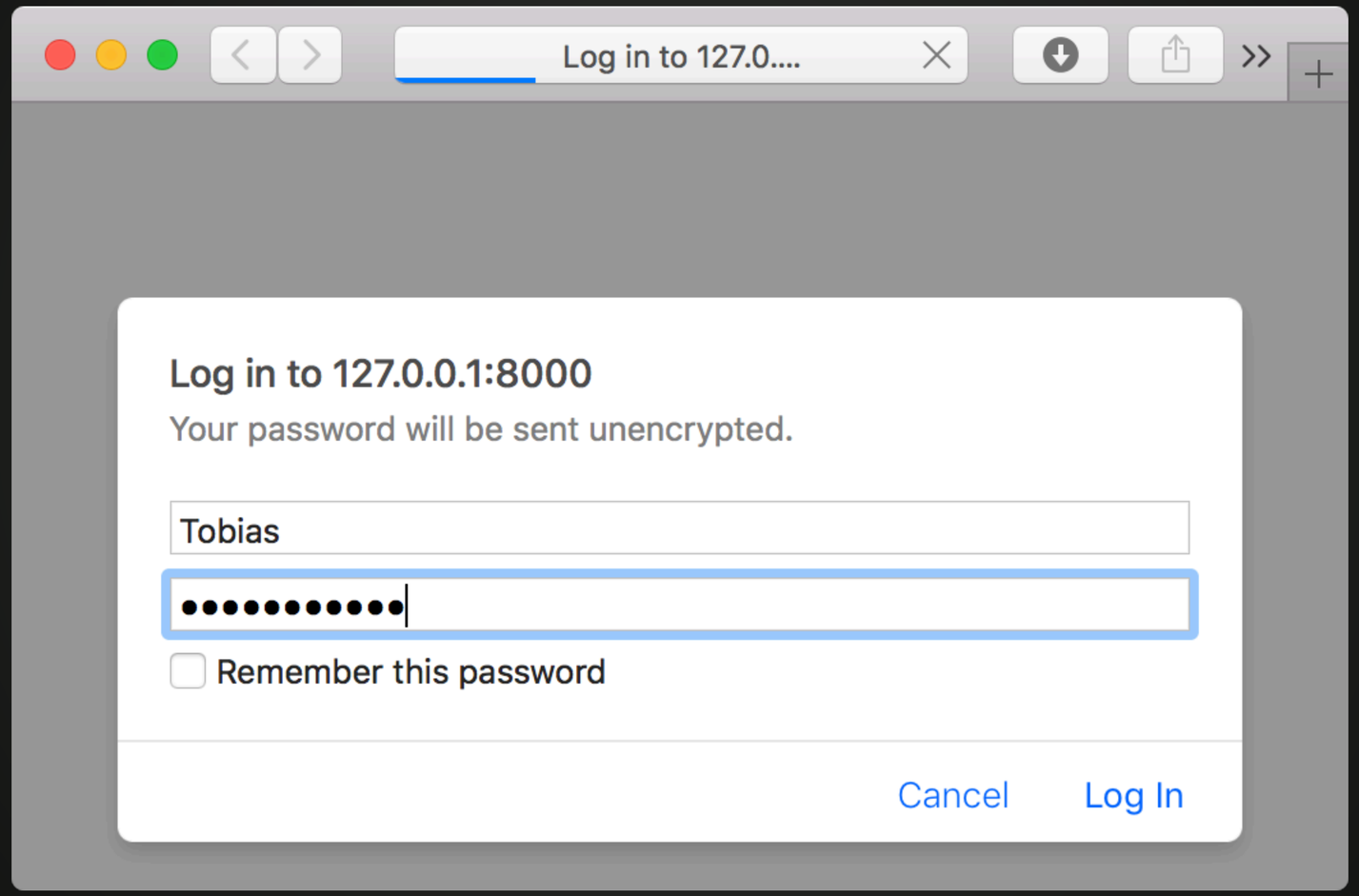
@tobiasnyholm

Security

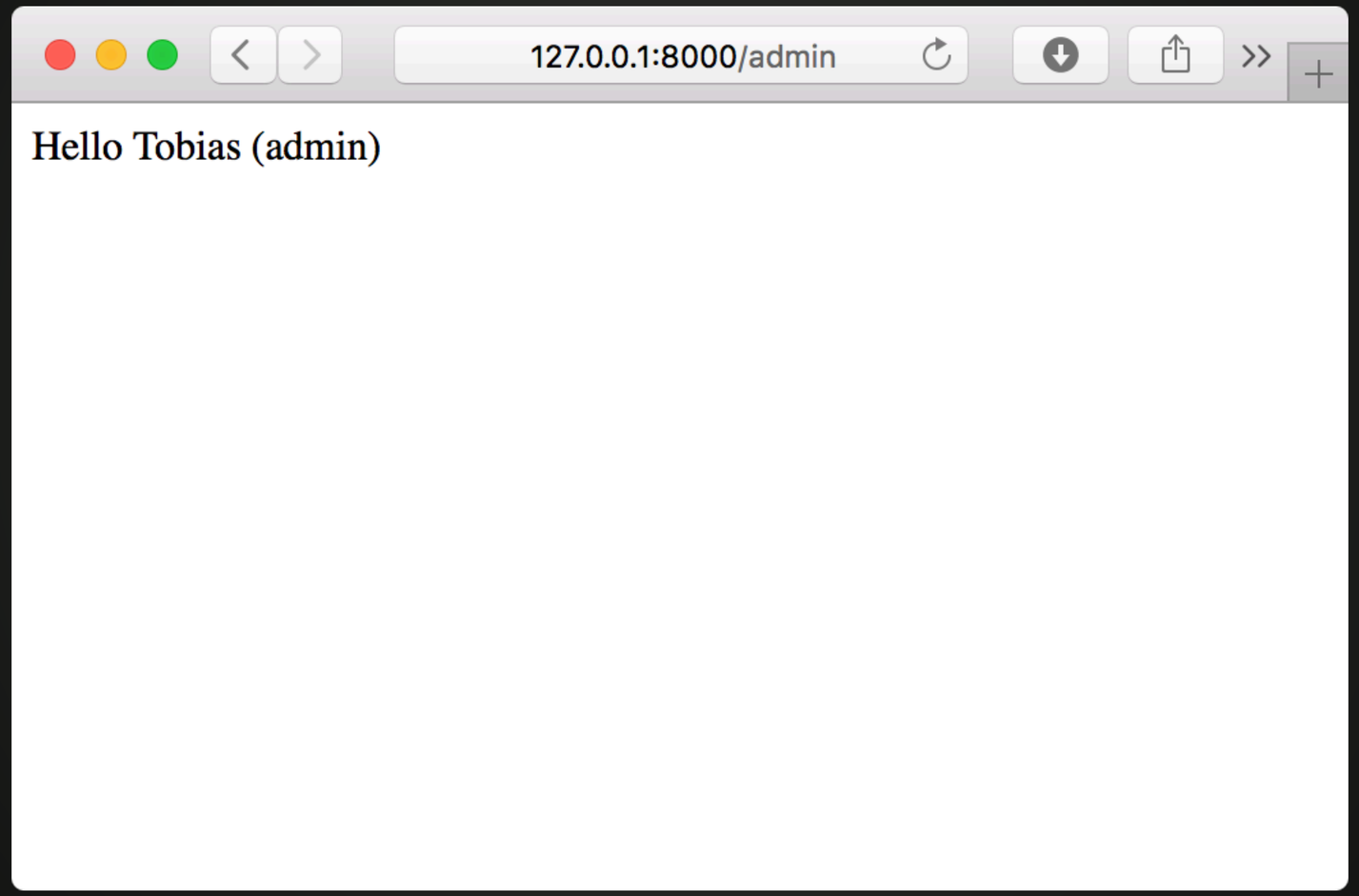


@tobiasnyholm

Security



Security



Security

```php
<?php

declare(strict_types=1);

namespace App\Controller;

use App\Security\TokenStorage;
use Nyholm\Psr7\Response;
use Psr\Http\Message\RequestInterface;

class AdminController
{
    private $tokenStorage;

    public function __construct(TokenStorage $tokenStorage)
    {
        $this->tokenStorage = $tokenStorage;
    }

    public function run(RequestInterface $request)
    {
        return new Response(200, [],
           sprintf('Hello %s (admin)', $this->tokenStorage->getLastToken()['username']));
    }
}
```

Security

Authentication vs Authorisation

@tobiasnyholm

```php
class SecurityVoters implements MiddlewareInterface
{
    /** @var  VoterInterface[] */
    private $voters;
    public function __construct(array $voters)
    {
        $this->voters = $voters;
    }
    public function __invoke(ServerRequestInterface $request, ResponseInterface $response, callable
    {
        $deny = 0;
        foreach ($this->voters as $voter) {
            $result = $voter->vote($request);
            switch ($result) {
                case VoterInterface::ACCESS_GRANTED:
                    return $next($request, $response);
                case VoterInterface::ACCESS_DENIED:
                    ++$deny;
                    break;
                default:
                    break;
            }
        }

        if ($deny > 0) {
            return new Response(403, [], 'Forbidden');
        }

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

@tobiasnyholm

Security

```php
declare(strict_types=1);

namespace App\Security\Voter;

use App\Security\TokenStorage;
use Psr\Http\Message\ServerRequestInterface;

class AdminVoter implements VoterInterface
{
    private $tokenStorage;
    public function __construct(TokenStorage$tokenStorage)
    {
        $this->tokenStorage = $tokenStorage;
    }

    public function vote(ServerRequestInterface $request)
    {
        $uri = $request->getUri()->getPath();
        $token = $this->tokenStorage->getLastToken();

        if ($token['username'] !== 'Tobias' && $uri === '/admin') {
            return VoterInterface::ACCESS_DENIED;
        }

        return VoterInterface::ACCESS_ABSTAIN;
    }
}
```

Security

```yaml
middleware.auth:
  class: App\Middleware\Authentication
  arguments: ['@security.token_storage']
  public: true

middleware.security:
  class: App\Middleware\SecurityVoters
  public: true
  arguments:
    - ['@security.voter.admin', '@security.voter.image']

security.token_storage:
  class: App\Security\TokenStorage

security.voter.admin:
  class: App\Security\Voter\AdminVoter
  arguments: ['@security.token_storage']

security.voter.image:
  class: App\Security\Voter\ImageVoter

controller.admin:
  class: App\Controller\AdminController
  arguments: ['@security.token_storage']
  public: true

controller.startpage:
  class: App\Controller\StartpageController
  public: true
```

Security

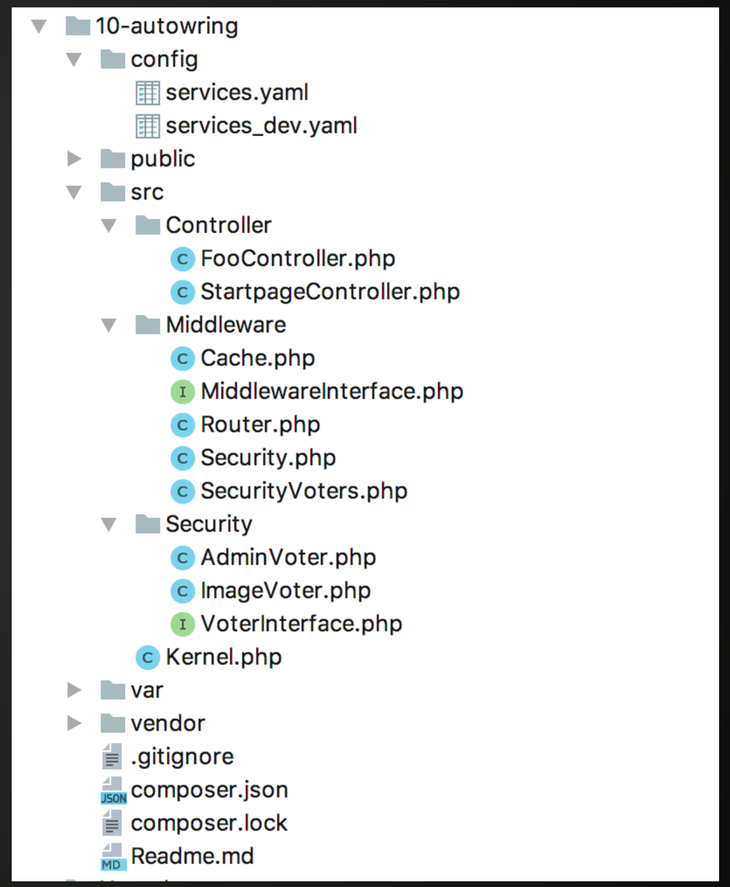
```php
class Kernel
{
    // …
    /**
     * Handle a Request and turn it in to a response.
     */
    public function handle(RequestInterface $request): ResponseInterface
    {
        $this->boot();

        $middlewares[] = $this->container->get(‘middleware.auth');
        $middlewares[] = $this->container->get('middleware.security');
        $middlewares[] = $this->container->get('middleware.cache');
        $middlewares[] = new \App\Middleware\Router();

        $runner = (new \Relay\RelayBuilder())->newInstance($middlewares);

        return $runner($request, new Response());
    }
    // …
}
```

Security

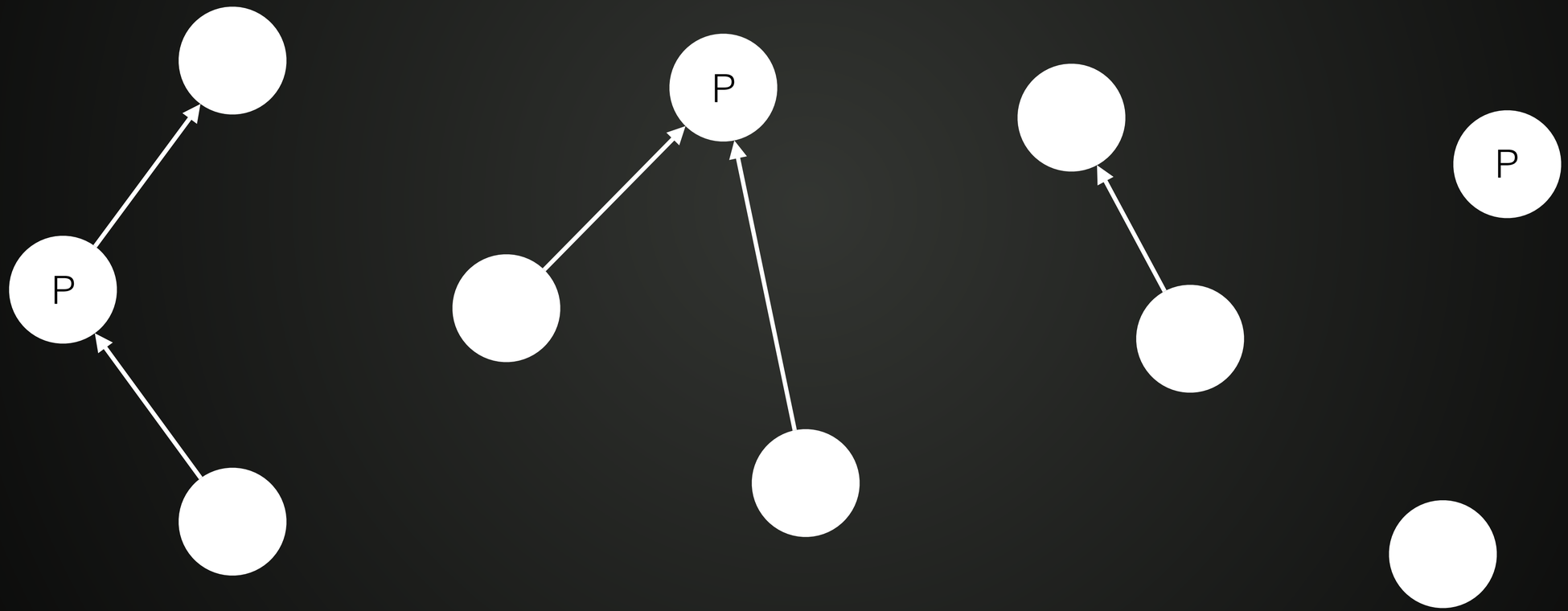
Autowiring

Autowiring

```yaml
# services.yaml
services:
  _defaults:
    autowire: true
    public: false

  App\:
    resource: '../src/*'
    exclude: '../src/{Entity,Tests,Kernel.php}'

  App\Controller\:
    resource: '../src/Controller'
    public: true
    tags: ['controller.service_arguments']

  App\Security\:
    resource: '../src/Security'
    tags: ['security.voter']
```



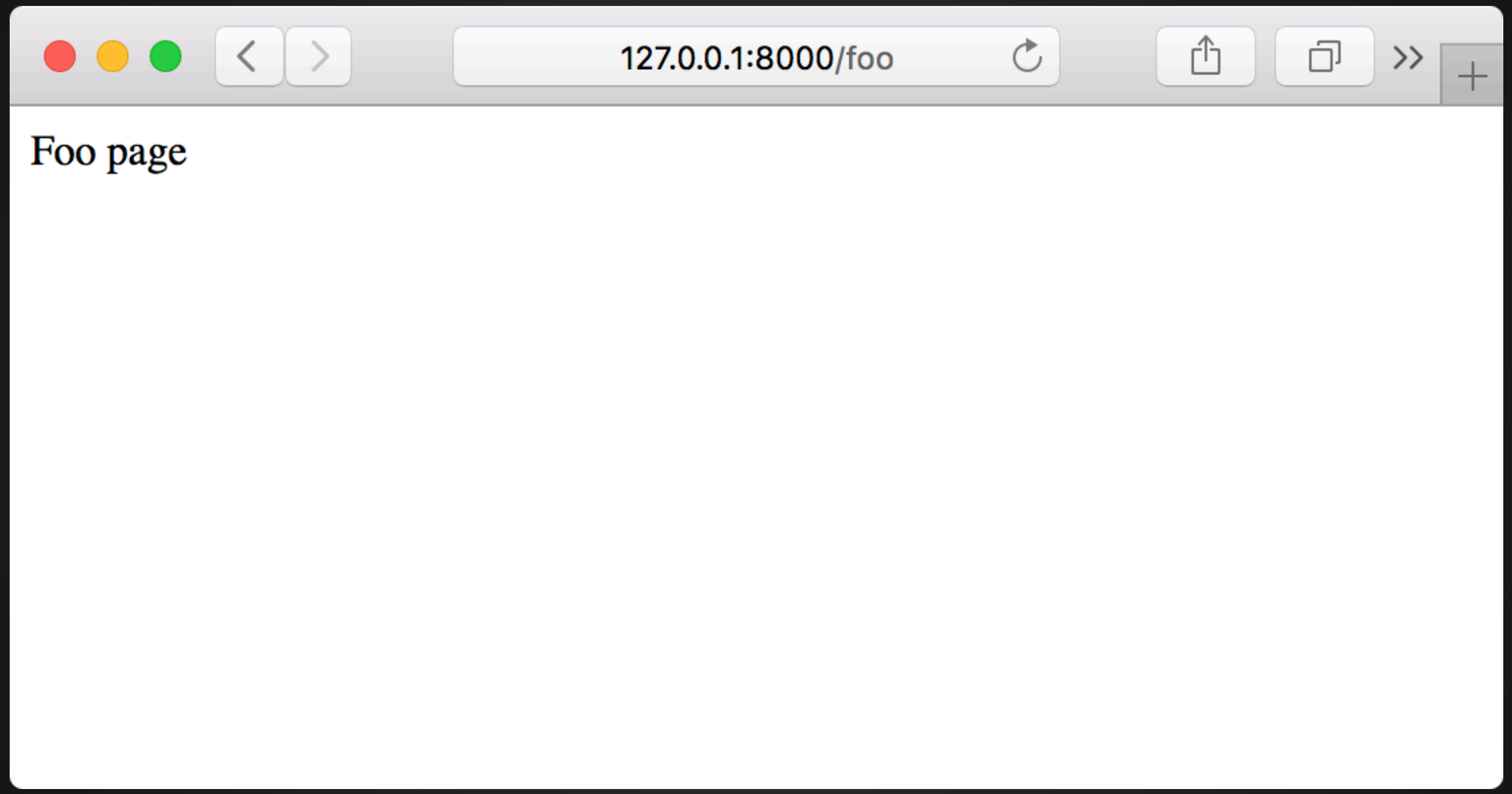
Autowiring



Autowiring

```php
class Kernel
{
    // …
    /**
     * Handle a Request and turn it in to a response.
     */
    public function handle(RequestInterface $request): ResponseInterface
    {
        $this->boot();

        $middlewares[] = $this->container->get(Authentication::class);
        $middlewares[] = $this->container->get(SecurityVoters::class);
        $middlewares[] = $this->container->get(Cache::class);
        $middlewares[] = new \App\Middleware\Router();

        $runner = (new \Relay\RelayBuilder())->newInstance($middlewares);

        return $runner($request, new Response());
    }
    // …
}
```

Autowiring



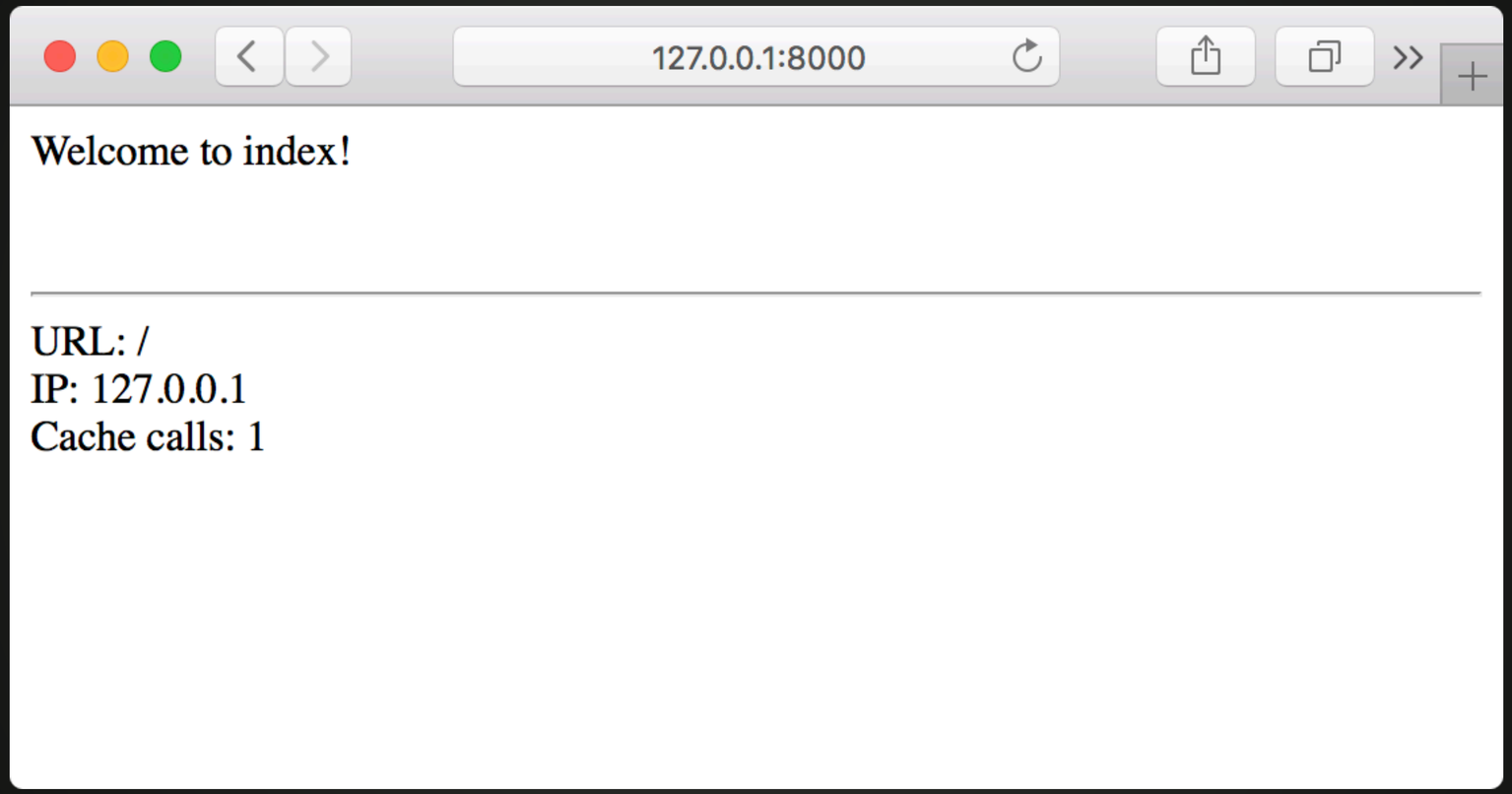
Toolbar

@tobiasnyholm

```php
class Toolbar implements MiddlewareInterface
{
    private $cacheDataCollector;

    public function __construct(CacheDataCollector $cacheDataCollector)
    {
        $this->cacheDataCollector = $cacheDataCollector;
    }

    public function __invoke(ServerRequestInterface $request, ResponseInterface $response, callable
    {
        $calls = $this->cacheDataCollector->getCalls();
        $getItemCalls = count($calls['getItem']);

        $content = $response->getBody()->__toString();
        $toolbar = <<<HTML
<br><br><br><hr>
URL: {$request->getUri()->getPath()}<br>
IP: {$request->getServerParams()['REMOTE_ADDR']}<br>
Cache calls: {$getItemCalls}<br>
HTML;

        $stream = (new StreamFactory())->createStream($content.$toolbar);
        $response = $response->withBody($stream);

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

```php
<?php

namespace App\DataCollector;

use Psr\Cache\CacheItemInterface;
use Psr\Cache\CacheItemPoolInterface;

class CacheDataCollector implements CacheItemPoolInterface
{
    /** @var CacheItemPoolInterface */
    private $real;
    private $calls;
    public function __construct(CacheItemPoolInterface $cache)
    {
        $this->real = $cache;
    }

    public function getCalls()
    {
        return $this->calls;
    }

    public function getItem($key)
    {
        $this->calls['getItem'][] = ['key'=>$key];
        return $this->real->getItem($key);
    }

    public function hasItem($key)
```

Toolbar



@tobiasnyholm

```php
class Toolbar implements MiddlewareInterface
{
    private $cacheDataCollector;

    public function __construct(CacheDataCollector $cacheDataCollector)
    {
        $this->cacheDataCollector = $cacheDataCollector;
    }

    public function __invoke(ServerRequestInterface $request, ResponseInterface $response, callable
    {
        $calls = $this->cacheDataCollector->getCalls();
        $getItemCalls = count($calls['getItem']);

        $content = $response->getBody()->__toString();
        $toolbar = <<<HTML
<br><br><br><hr>
URL: {$request->getUri()->getPath()}<br>
IP: {$request->getServerParams()['REMOTE_ADDR']}<br>
Cache calls: {$getItemCalls}<br>
HTML;

        $stream = (new StreamFactory())->createStream($content.$toolbar);
        $response = $response->withBody($stream);

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

Exception

Exception

```php
<?php

declare(strict_types=1);

namespace App\Controller;

use Nyholm\Psr7\Response;
use Psr\Http\Message\RequestInterface;

class ExceptionController
{
    public function run(RequestInterface $request)
    {
        throw new \RuntimeException('This is an exception');
    }
}
```

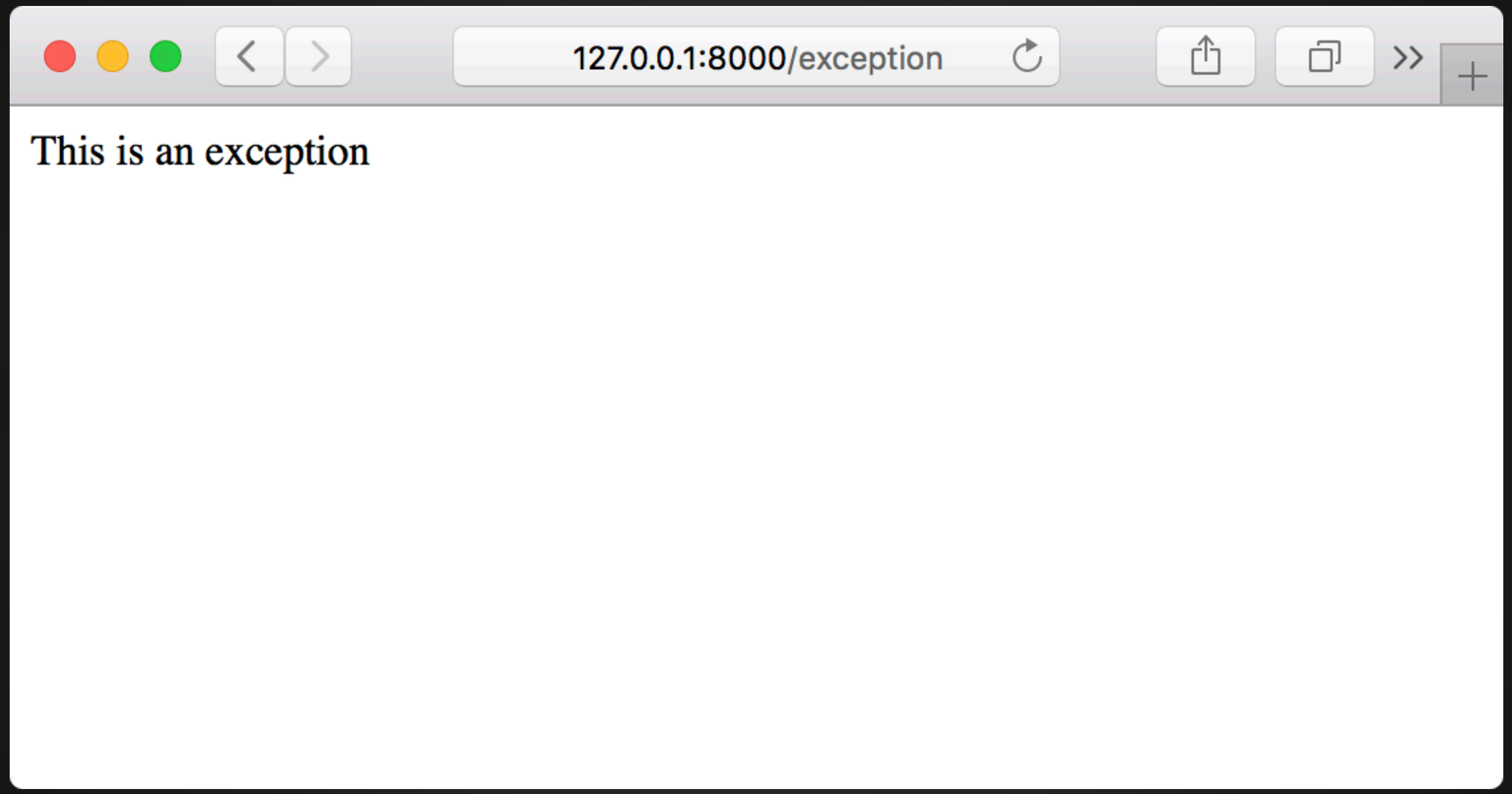
Exception

```php
<?php

namespace App\Middleware;

use Cache\Adapter\Apcu\ApcuCachePool;
use Nyholm\Psr7\Response;
use Psr\Cache\CacheItemPoolInterface;
use Psr\Http\Message\ResponseInterface;
use Psr\Http\Message\ServerRequestInterface;

class ExceptionHandler implements MiddlewareInterface
{
    public function __invoke(ServerRequestInterface $request, ResponseInterface $response, call
    {
        try {
            $response = $next($request, $response);
        } catch (\Throwable $exception) {
            $response = new Response(500, [], $exception->getMessage());
        }

        return $response;
    }
}
```

Exception

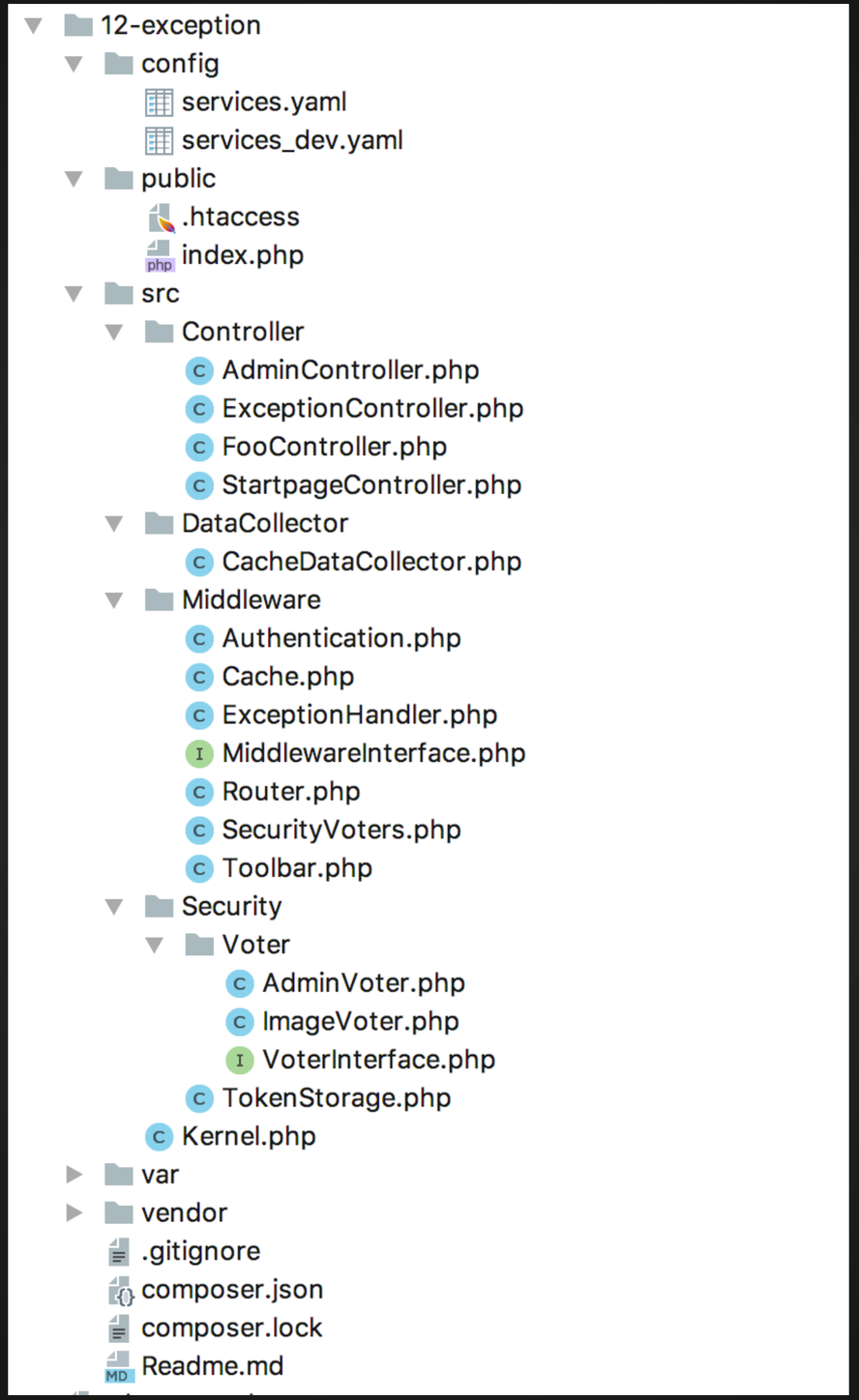
```php
class Kernel
{
    /**
     * Handle a Request and turn it in to a response.
     */
    public function handle(RequestInterface $request): ResponseInterface
    {
        $this->boot();

        $middlewares[] = $this->container->get(ExceptionHandler::class);
        $middlewares[] = $this->container->get(Authentication::class);
        $middlewares[] = $this->container->get(SecurityVoters::class);
        $middlewares[] = $this->container->get(Cache::class);
        $middlewares[] = new \App\Middleware\Router();
        $middlewares[] = $this->container->get(Toolbar::class);

        $runner = (new \Relay\RelayBuilder())->newInstance($middlewares);

        return $runner($request, new Response());
    }
}
```

Exception



Exception

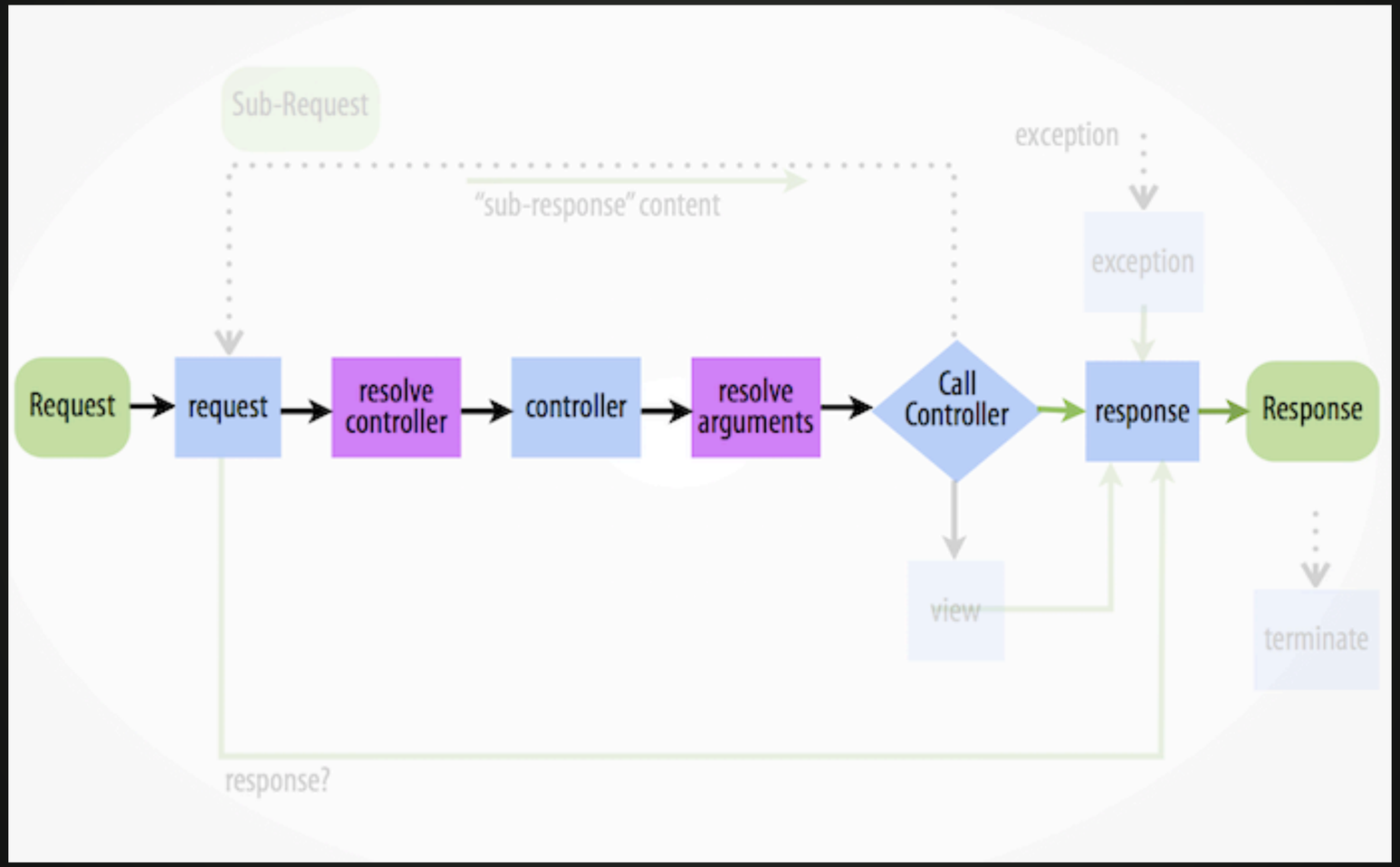
```php
<?php

namespace App\Middleware;

use Cache\Adapter\Apcu\ApcuCachePool;
use Nyholm\Psr7\Response;
use Psr\Cache\CacheItemPoolInterface;
use Psr\Http\Message\ResponseInterface;
use Psr\Http\Message\ServerRequestInterface;

class ExceptionHandler implements MiddlewareInterface
{
    public function __invoke(ServerRequestInterface $request, ResponseInterface $response, call
    {
        try {
            $response = $next($request, $response);
        } catch (\Throwable $exception) {
            $response = new Response(500, [], $exception->getMessage());
        }

        return $response;
    }
}
```



```
{
    "name": "tobias/4-eventLoop",
    "authors": [
        {
            "name": "Tobias Nyholm",
            "email": "tobias.nyholm@gmail.com"
        }
    ],
    "autoload": {
        "psr-4": {
            "App\\": "src/"
        }
    },
    "require": {
        "cache/filesystem-adapter": "^1.0",
        "cache/void-adapter": "^1.0",
        "nyholm/psr7": "^0.3.0",
        "relay/relay": "^1.1",
        "symfony/config": "^4.0",
        "symfony/dependency-injection": "^4.0",
        "symfony/yaml": "^4.0"
    },
    "config": {
        "sort-packages": true
    }
}
```

Where is Symfony?

HTTP objects



Controllers

Event loop

kernel.request

kernel.controller

kernel.view

kernel.response

kernel.finish_request

kernel.terminate

Cache

PSR-6
PSR-16

Container

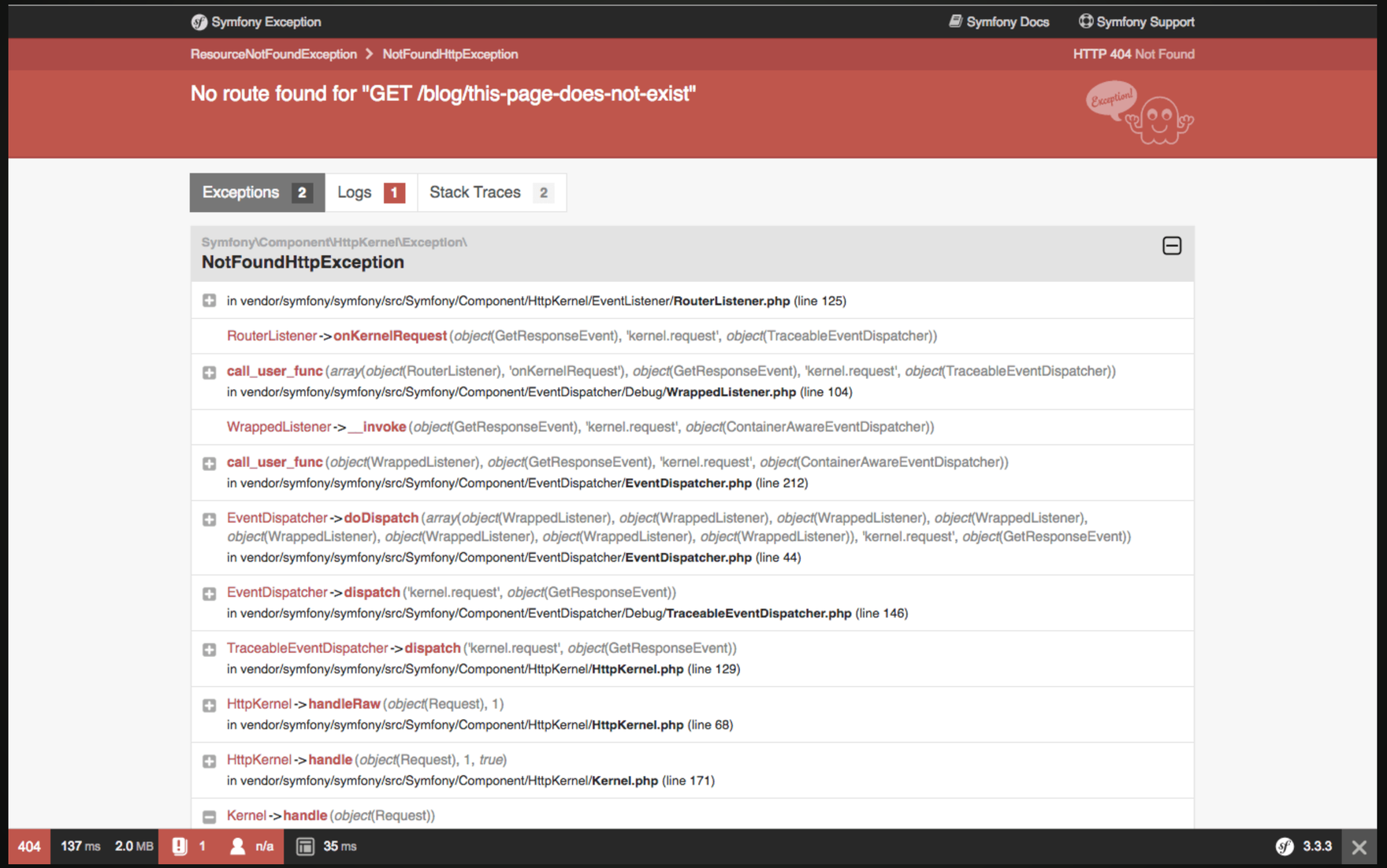
Router

Security

Autowiring

Toolbar

Exception


Symfony Exception
Symfony Docs
Symfony Support
ResourceNotFoundException > NotFoundHttpException
HTTP 404 Not Found
No route found for "GET /blog/this-page-does-not-exist"
Exceptions 2
Logs 1
Stack Traces 2
Symfony\Component\HttpKernel\Exception\
NotFoundHttpException
in vendor/symfony/symfony/src/Symfony/Component/HttpKernel/EventListener/RouterListener.php (line 125)
RouterListener->onKernelRequest(object(GetResponseEvent), 'kernel.request', object(TraceableEventDispatcher))
call_user_func(array(object(RouterListener), 'onKernelRequest'), object(GetResponseEvent), 'kernel.request', object(TraceableEventDispatcher))
in vendor/symfony/symfony/src/Symfony/Component/EventDispatcher/Debug/WrappedListener.php (line 104)
WrappedListener->__invoke(object(GetResponseEvent), 'kernel.request', object(ContainerAwareEventDispatcher))
call_user_func(object(WrappedListener), object(GetResponseEvent), 'kernel.request', object(ContainerAwareEventDispatcher))
in vendor/symfony/symfony/src/Symfony/Component/EventDispatcher/EventDispatcher.php (line 212)
EventDispatcher->doDispatch(array(object(WrappedListener), object(WrappedListener), object(WrappedListener), object(WrappedListener), object(WrappedListener), object(WrappedListener), object(WrappedListener), object(WrappedListener)), 'kernel.request', object(GetResponseEvent))
in vendor/symfony/symfony/src/Symfony/Component/EventDispatcher/EventDispatcher.php (line 44)
EventDispatcher->dispatch('kernel.request', object(GetResponseEvent))
in vendor/symfony/symfony/src/Symfony/Component/EventDispatcher/Debug/TraceableEventDispatcher.php (line 146)
TraceableEventDispatcher->dispatch('kernel.request', object(GetResponseEvent))
in vendor/symfony/symfony/src/Symfony/Component/HttpKernel/HttpKernel.php (line 129)
HttpKernel->handleRaw(object(Request), 1)
in vendor/symfony/symfony/src/Symfony/Component/HttpKernel/HttpKernel.php (line 68)
HttpKernel->handle(object(Request), 1, true)
in vendor/symfony/symfony/src/Symfony/Component/HttpKernel/Kernel.php (line 171)
Kernel->handle(object(Request))
404 137 ms 2.0 MB 1 n/a 35 ms 3.3.3

This was simple Symfony

What is Symfony?

nyholm/psr7	→	symfony/http-foundation
relay/relay	→	symfony/event-dispatcher
php-cache/*	→	symfony/cache
Custom kernel	→	symfony/http-kernel
Custom security	→	symfony/security

Questions?

https://joind.in/talk/d7eff