

Building really fast apps

Tobias Nyholm

Why?

Tobias Nyholm

- Creating good stuff at Eneba.com
- Certified Symfony developer
- Symfony core member
- Symfony CARE
- PHP-Stockholm
- Open source

Open source

Assert NewRelicBundle
Neo4j FriendsOfApi/boilerplate
Backup-manager/symfony Guzzle Buzz nyholm/effective-interest-rate
Swap Puli LinkedIn API client
php-http/discovery NSA
PSR7 PHP-cache PHP-Translation
CacheBundle
KNP Github API league/geotools Mailgun
Stampie HTTPPlug happyr/normal-distribution-bundle
MailgunBundle php-http/httpplug-bundle SymfonyBundleTest
php-http/multipart-stream
SimpleBus integrations PHP-Geocoder
PSR HTTP clients BazingaGeocoderBundle

“Frameworks are slow”

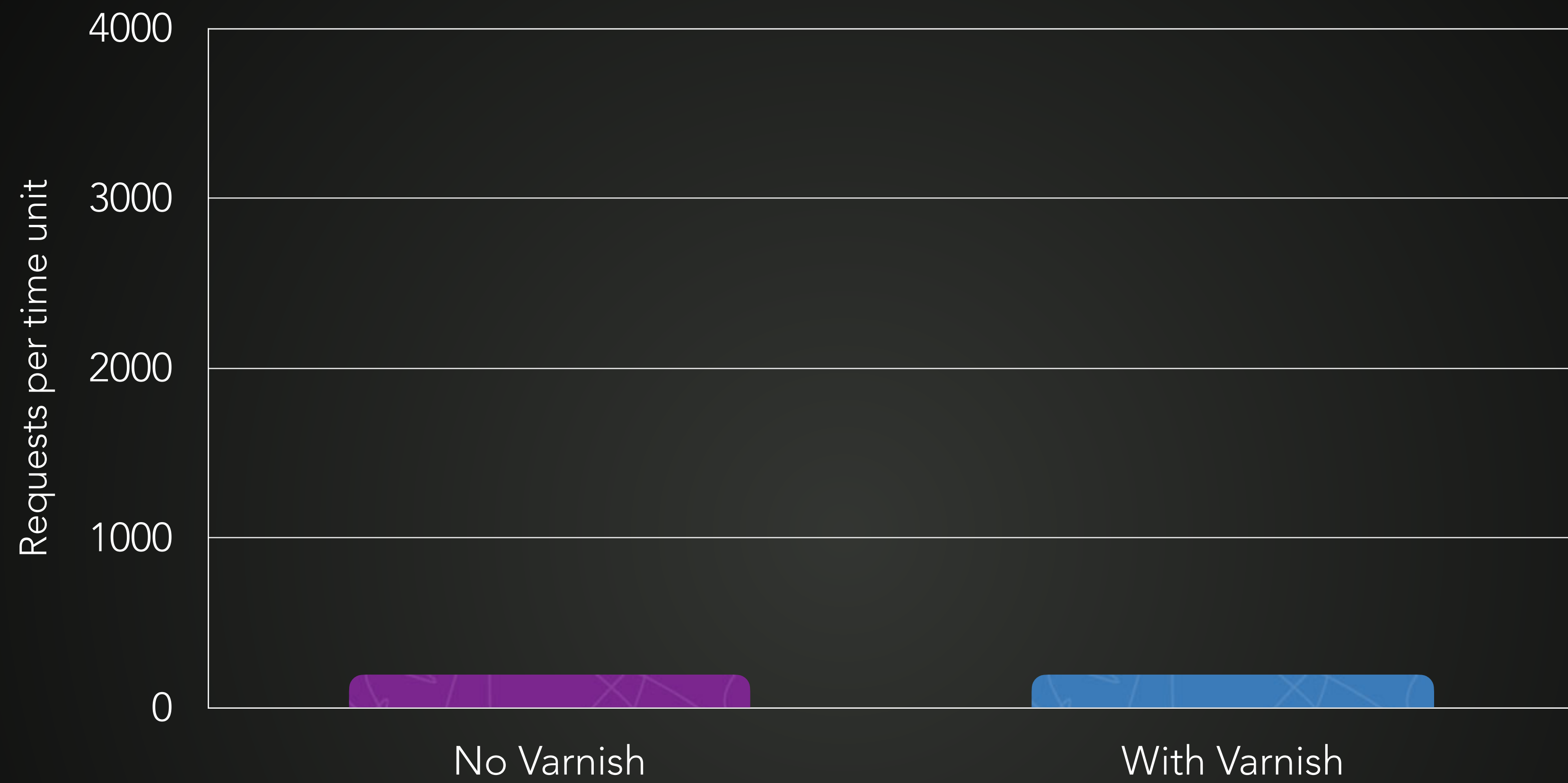
What is a framework?

“Frameworks are slow”

Let's talk about performance

Rule 1: Buy a better server

Rule 2: Use Varnish



Rule 3: Run less code


```
section .text
    global _start        ; must be declared for linker (ld)
```

```
_start:                ; tell linker entry point
```

```
    mov    edx,len      ; message length
    mov    ecx,msg      ; message to write
    mov    ebx,1         ; file descriptor (stdout)
    mov    eax,4         ; system call number (sys_write)
    int     0x80         ; call kernel
```

```
    mov    eax,1         ; system call number (sys_exit)
    int     0x80         ; call kernel
```

```
section .data
```

```
msg    db    'Hello, world!',0xa    ; our dear string
len     equ   $ - msg                ; length of our dear string
```

More code = More “ticks”

Hello world

Version	Lines of code	Lines executed	Function calls	Memory	Time
Symfony 1.4	172 000	2 401	10 588	1 200 Kb	198 ms
Symfony 2.8	415 700	1 928	5 051	1 600 Kb	100 ms
Symfony 3.4	433 600	2 649	4 004	1 600 Kb	82 ms
Symfony 4.1	136 000	1 029	1 421	500 Kb	31 ms
Symfony 4.4	117 000	1 409	1 767	568 Kb	16 ms

Hello world

Version	Lines of code	Lines executed	Function calls	Memory	Time
Symfony 4.4	117 000	1 409	1 767	568 Kb	16 ms
Symfony 5.4	123 000	1 411	1 620	520 Kb	16 ms
Symfony 6.4	133 000	1 584	1 839	579 Kb	16 ms
Symfony 7.2	144 000	1 562	1 855	583 Kb	16 ms

Fastest application

```
<?php // index.php

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

What is your application doing?

```
<?php
```

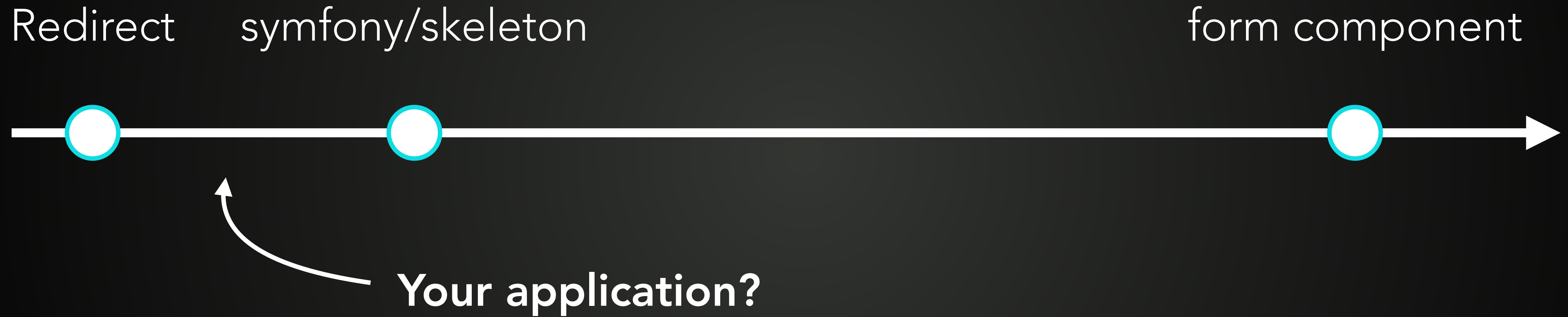
```
$uri = $_SERVER['REQUEST_URI'];  
$base = 'https://happy.com';
```

```
if (substr($uri, 0, 1) !== 'j') {  
    redirect($base);  
}
```

```
redirect(sprintf('%s/x/job/%d-x_x', $base, substr($uri, 1)));
```

```
function redirect($url)  
{  
    header('Location: '.$url);  
    exit(0);  
}
```


Complexity



Do you need this?

Yaml Translation Security
Serializer Validation
Forms
Finder
Debug **Do you need this?** Router
Event dispatcher
Dependency injection
HTTP Foundation
HTTP Kernel

Do we need HTTP?

HTTP Foundation

VS

`$_SERVER['REQUEST_URI']`

Messenger component?

Doctrine or just PDO?

@tobiasnyholm



Well tested libraries

Takes care of all edge cases

Nice APIs

More code

vs

You write code yourself

Think about your edge cases

Sometimes tricky APIs

Less code

Building a small application

<https://github.com/Nyholm/sunflower>

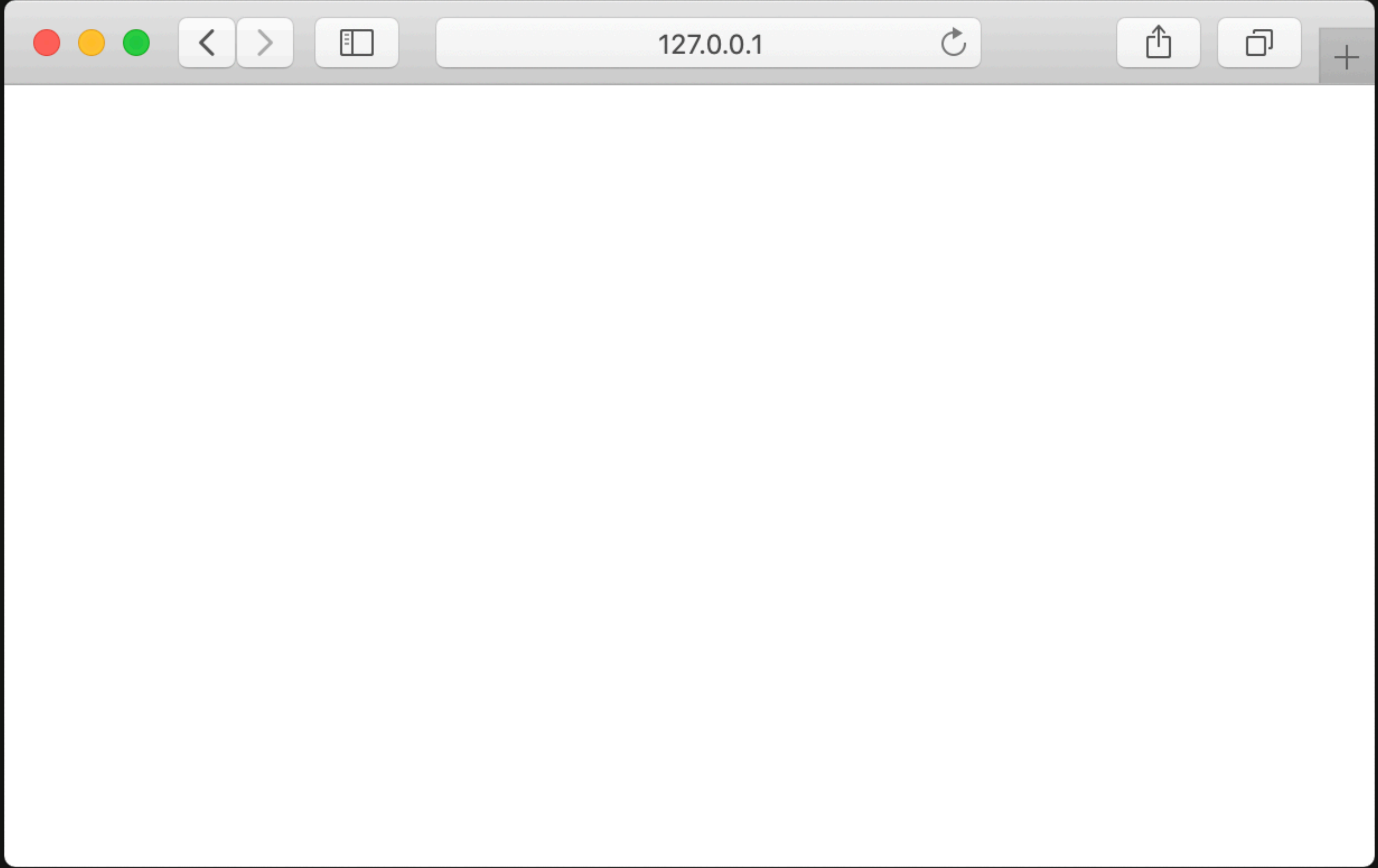
Middleware pattern


```
use Symfony\Component\HttpFoundation\Response;  
use Symfony\Component\HttpFoundation\Request;  
use App\Runner;
```

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

```
$request = Request::createFromGlobals();  
$response = new Response();
```

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

```
use Symfony\Component\HttpFoundation\Response;  
use Symfony\Component\HttpFoundation\Request;  
use App\Runner;
```

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

```
$request = Request::createFromGlobals();  
$response = new Response();
```

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

```
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\HttpFoundation\Request;
use App\Runner;

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

$request = Request::createFromGlobals();
$response = new Response();

$middleware[] = function(Request $request, Response $response, callable $next) {
    if ($request->getMethod() !== 'GET') {
        return new Response('Method not allowed', 405);
    }
    return $next($request, $response);
};

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



```
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\HttpFoundation\Request;
use App\Runner;

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

$request = Request::createFromGlobals();
$response = new Response();

$middleware[] = function(Request $request, Response $response, callable $next) {
    if ($request->getMethod() !== 'GET') {
        return new Response('Method not allowed', 405);
    }
    return $next($request, $response);
};

$middleware[] = function(Request $request, Response $response, callable $next) {
    $response->setContent('foobar');
    return $next($response, $response);
};

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



```
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\HttpFoundation\Request;
use App\Runner;

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

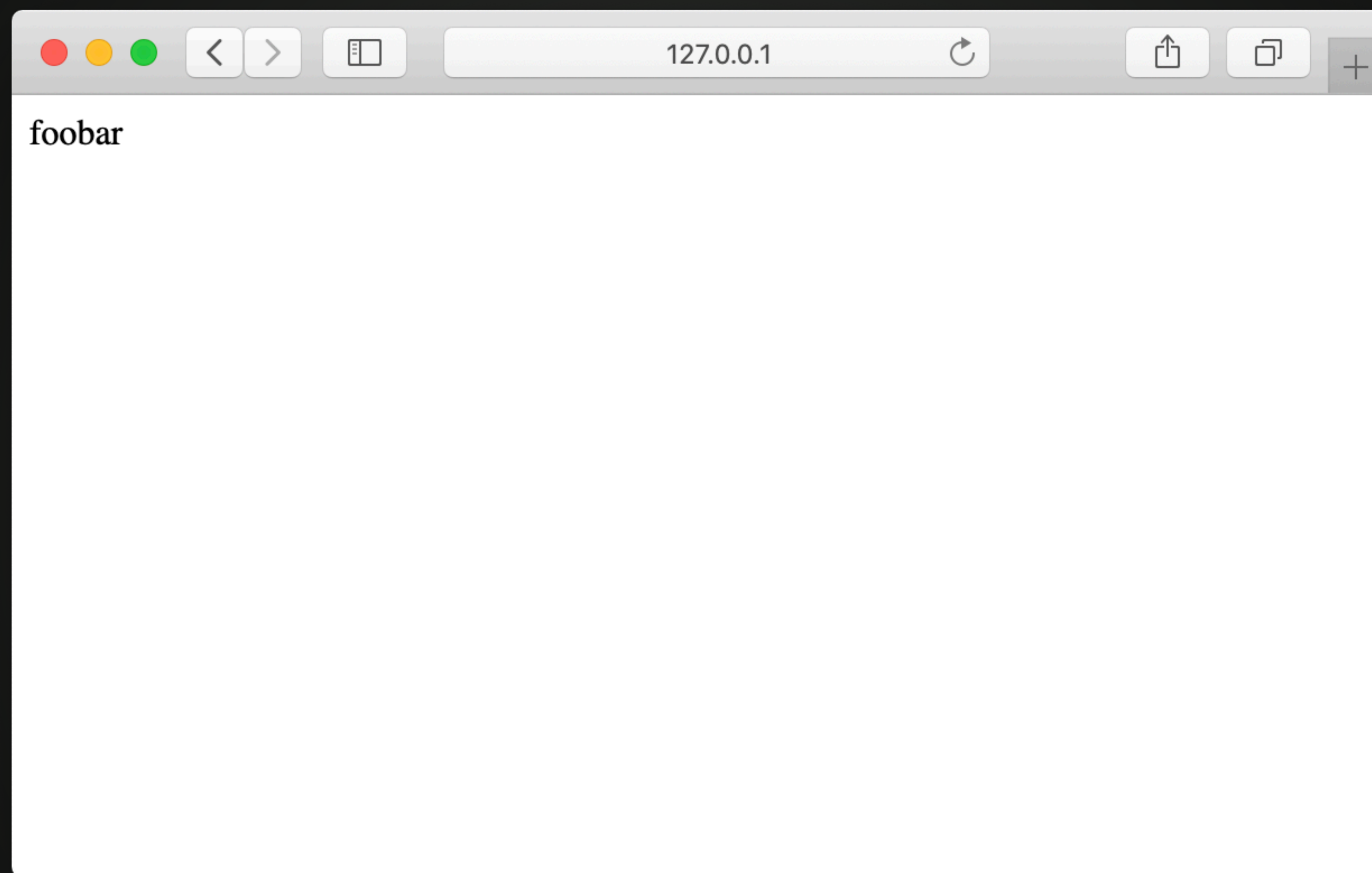
$request = Request::createFromGlobals();
$response = new Response();

$middleware[] = function(Request $request, Response $response, callable $next) {
    if ($request->getMethod() !== 'GET') {
        return new Response('Method not allowed', 405);
    }
    return $next($request, $response);
};

$middleware[] = function(Request $request, Response $response, callable $next) {
    $response->setContent('foobar');
    return $next($response, $response);
};

$runner = new Runner($middleware);
$response = $runner($request, $response);

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



```
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\HttpFoundation\Request;
use App\Runner;

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

$request = Request::createFromGlobals();
$response = new Response();

$middleware[] = function(Request $request, Response $response, callable $next) {
    if ($request->getMethod() !== 'GET') {
        return new Response('Method not allowed', 405);
    }
    return $next($request, $response);
};

$middleware[] = function(Request $request, Response $response, callable $next) {
    $response->setContent('foobar');
    return $next($response, $response);
};

$runner = new Runner($middleware);
$response = $runner($request, $response);

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



```
namespace App;

use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\HttpFoundation\Request;

class Runner
{
    /** @var callable[] */
    private $queue;

    public function __construct(array $queue)
    {
        $this->queue = $queue;
    }

    public function __invoke(Request $request, Response $response)
    {
        $middleware = array_shift($this->queue);
        if (null === $middleware) {
            return $response;
        }

        return $middleware($request, $response, $this);
    }
}
```



```
<?php
```

```
namespace App\Middleware;
```

```
use Symfony\Component\HttpFoundation\Response;
```

```
use Symfony\Component\HttpFoundation\Request;
```

```
interface MiddlewareInterface
```

```
{
```

```
    public function __invoke(Request $request, Response $response, callable $next);
```

```
}
```

```
<?php
```

```
use Symfony\Component\HttpFoundation\Response;  
use Symfony\Component\HttpFoundation\Request;  
use App\Runner;
```

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

```
$request = Request::createFromGlobals();  
$response = new Response();
```

```
$middleware[] = new Cache;  
$middleware[] = new NewRelic;  
$middleware[] = new Security;  
$middleware[] = new Router;
```

```
$runner = new Runner($middleware);  
$response = $runner($request, $response);
```

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

```
class Router implements MiddlewareInterface
{
    private $geocoderController;
    private $ipController;
    public function __construct(GeocoderController $geocoderController, IpController $ipController)
    {
        $this->geocoderController = $geocoderController;
        $this->ipController = $ipController;
    }
    public function __invoke(ServerRequestInterface $request, ResponseInterface $response, callable $uri = $request->getUri()->getPath());

    switch ($uri) {
        case '/':
            $response = $this->geocoderController->geocodeAction($request, $response);
            break;
        case '/locale':
            $response = $this->ipController->localeAction($request, $response);
            break;
        case '/from-ip':
            $response = $this->ipController->ipAction($request, $response);
            break;
        default:
            $response = $response->withStatus(404);
            $response->getBody()->write('Not Found');
            break;
    }
    return $next($request, $response);
}
```



```
<?php
```

```
use Symfony\Component\HttpFoundation\Response;  
use Symfony\Component\HttpFoundation\Request;  
use App\Runner;
```

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

```
$request = Request::createFromGlobals();  
$response = new Response();
```

```
$middleware[] = new Cache;  
$middleware[] = new NewRelic;  
$middleware[] = new Security;  
$middleware[] = new Router;
```

```
$runner = new Runner($middleware);  
$response = $runner($request, $response);
```

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


Custom kernel

```
class Kernel
{
    private $booted = false;
    private $debug;
    private $env;

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

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

    /**
     * Handle a Request and turn it in to a response.
     */
    public function handle(Request $request): Response
    {
        $this->boot();

        $middleware[] = $this->container->get(Cache::class);
        $middleware[] = $this->container->get(NewRelic::class);
        $middleware[] = $this->container->get(Security::class);
        $middleware[] = $this->container->get(Router::class);

        $runner = new Runner($middleware);
    }
}
```

```
# 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']
```



```
        $container = new \CachedContainer();
    } else {
        $container = new ContainerBuilder();
        $container->setParameter('kernel.project_dir', $this->getProjectDir());
        $container->setParameter('kernel.environment', $this->env);
```

```
        $container->registerForAutoconfiguration(EventSubscriberInterface::class)
            ->addTag('kernel.event_subscriber');
```

```
        $container->registerForAutoconfiguration(Command::class)
            ->addTag('console.command');
```

```
        $container->addCompilerPass(new AddConsoleCommandPass());
        $container->addCompilerPass(
            new RegisterListenersPass(EventDispatcherInterface::class),
            PassConfig::TYPE_BEFORE_REMOVING
        );
```

```
        $fileLocator = new FileLocator($this->getProjectDir().'/config');
        $loader = new YamlFileLoader($container, $fileLocator);
        try {
            $loader->load('services.yaml');
            $loader->load('services_'.$this->env.'.yaml');
        } catch (FileLocatorFileNotFoundException $e) {
        }
    }
```

```
    $container->compile();
```

```
    //dump the container
```

Too much boilerplate?

Use the FrameworkBundle

Improve performance

Improve performance

#1: Buy a better server

#2: Use Varnish

#3: Run less code

Improve performance

#1: Buy a better computer

#2: Use cache

#3: Run less code

0

Tip count

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by:

All instance types

Current generation

Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)								
	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
☐	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
☒	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes
☐	General purpose	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
☐	General purpose	t3a.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes
☐	General purpose	t3a.micro	2	1	EBS only	Yes	Up to 5 Gigabit	Yes
☐	General purpose	t3a.small	2	2	EBS only	Yes	Up to 5 Gigabit	Yes
☐	General purpose	t3a.medium	2	4	EBS only	Yes	Up to 5 Gigabit	Yes
☐	General purpose	t3a.large	2	8	EBS only	Yes	Up to 5 Gigabit	Yes
☐	General purpose	t3a.xlarge	4	16	EBS only	Yes	Up to 5 Gigabit	Yes
☐	General purpose	t3a.2xlarge	8	32	EBS only	Yes	Up to 5 Gigabit	Yes

1

Tip count

Cache in Symfony

2

Tip count

```
# config/packages/cache.yaml
framework:
  cache:
    pools:
      app.redis_cache_chain:
        default_lifetime: 31536000 # One year
        adapters:
          - cache.adapter.array
          - cache.adapter.apcu
          - {name: cache.adapter.redis, provider: 'redis://user:password@example.com' }
```


Doctrine cache

2

Tip count

```
doctrine:
    orm:
        metadata_cache_driver:
            type: service
            id: doctrine.system_cache_provider
        query_cache_driver:
            type: service
            id: doctrine.system_cache_provider
        result_cache_driver:
            type: service
            id: doctrine.result_cache_provider

services:
    doctrine.result_cache_provider:
        class: Symfony\Component\Cache\DoctrineProvider
        public: false
        arguments:
            - '@app.redis_cache_chain'
    doctrine.system_cache_provider:
        class: Symfony\Component\Cache\DoctrineProvider
        public: false
        arguments:
            - '@app.file_cache_chain'
```

Result cache

3

Tip count

```
class PageRepository extends ServiceEntityRepository
{
    public function getContent(string $page)
    {
        return $this->createQueryBuilder('p')
            ->where('p.slug = :slug')
            ->setParameter('slug', $page)
            ->getQuery()
            ->useResultCache(true, 3600)
            ->getOneOrNullResult();
    }
}
```

Cache HTTP calls


```
class TwitterClient
{
    private $client;
    private $logger;

    public function __construct(HttpClientInterface $client, LoggerInterface $logger)
    {
        $this->client = $client;
        $this->logger = $logger;
    }

    public function getTweets(): array
    {
        $response = $this->client->request('GET', 'https://api.twitter.com/latest-tweets');

        if (200 !== $response->getStatusCode()) {
            $this->logger->error('Could not get tweets from Twitter');
            return [];
        }

        return $response->toArray();
    }
}
```

Average time waiting on Twitter (math)

4

Tip count

Measure window: **600 s**

Number of function calls: $2 \text{ calls/s} * 600 \text{ s} = \mathbf{1200}$

Twitter API: **0,5 s**

Total requests: **1200 req**

Total time: $0,5 * 1200 = \mathbf{600 \text{ s}}$

Average time: $600 / 1200 = 0,5 \text{ s} = \mathbf{500 \text{ ms}}$

```
use Psr\Log\LoggerInterface;
use Symfony\Component\HttpClient\HttpClient;
use Symfony\Contracts\Cache\CacheInterface;
use Symfony\Contracts\Cache\ItemInterface;
use Symfony\Contracts\HttpClient\HttpClientInterface;

class TwitterClient
{
    private $cache;
    private $client;
    private $logger;

    public function __construct(CacheInterface $cache, HttpClientInterface $client, LoggerInterface $logger)
    {
        $this->cache = $cache;
        $this->client = $client;
        $this->logger = $logger;
    }

    public function getTweets(): array
    {
        $tweets = $this->cache->get(
            'latest-tweets',
            \Closure::fromCallable([$this, 'fetchLatestTweets'])
        );

        return $tweets;
    }
}
```


Average time waiting on Twitter (math)

4

Tip count

Measure window: **600 s**

Number of function calls: $2 \text{ calls/s} * 600 \text{ s} = \mathbf{1200}$

Twitter API: **0,5 s**

Total requests: **10 req** (we cache for 60s)

Total time: $0,5 * 10 = \mathbf{5 \text{ s}}$

Average time: $5 / 1200 \approx \mathbf{4 \text{ ms}}$

4 ms < 500 ms

(Class) local cache


```
class CircleArea
{

    public function compute(float $radius)
    {
        return $radius * $radius * $this->getPi();
    }

    private function getPi(): float
    {
        return 20 * atan(1/7) + 8 * atan(3/79);
    }
}
```

```
class CircleArea
{
    private $pi;

    public function compute(float $radius)
    {
        return $radius * $radius * $this->getPi();
    }

    private function getPi(): float
    {
        if ($this->pi === null) {
            $this->pi = 20 * atan(1/7) + 8 * atan(3/79);
        }

        return $this->pi;
    }
}
```

Postpone work

6

Tip count

```
framework:
  messenger:
    transports:
      async: amqp://guest:guest@localhost:5672/%2f/messages

    routing:
      'App\Message\YourMessage' : async
```


Code generation

Lazy services

```
class MyService
```

```
{
```

```
    private $logger;
```

```
    private $bar;
```

```
    private $client;
```

```
    private $foo;
```

```
    private $cache;
```

```
    private $other;
```

```
    public function __construct(
```

```
        LoggerInterface $logger,
```

```
        BarService $bar,
```

```
        HttpClientInterface $client,
```

```
        FooService $foo,
```

```
        CacheInterface $cache,
```

```
        MyOtherService $other
```

```
    )
```

```
    {
```

```
        $this->logger = $logger;
```

```
        $this->bar = $bar;
```

```
        $this->client = $client;
```

```
        $this->foo = $foo;
```

```
        $this->cache = $cache;
```

```
        $this->other = $other;
```

```
    }
```

```
}
```

8

Tip count


```
$ composer require symfony/proxy-manager-bridge
```

```
# config/services.yaml
services:
    App\MyService:
        lazy: true
```

```
final class RequestListener implements EventSubscriberInterface
{
    // ... (Constructor with may dependencies)
    public static function getSubscribedEvents(): array
    {
        return [ KernelEvents::REQUEST => ['onRequest', 3] ];
    }

    /**
     * Verify accept header on the /api/ path.
     */
    public function onRequest(RequestEvent $event): void
    {
        if (!$event->isMasterRequest()) {
            return;
        }

        if (0 !== \mb_strpos($event->getRequest()->getPathInfo(), '/api/')) {
            return;
        }

        $request = $event->getRequest();

        // ...
        // Use all these dependencies
    }
}
```

9

Tip count

"Lazy" is global :(

10

Tip count

Service subscribers

```
final class RequestListener implements EventSubscriberInterface, ServiceSubscriberInterface
{
    private $locator;
    public function __construct(ContainerInterface $locator)
    {
        $this->locator = $locator;
    }

    public static function getSubscribedServices()
    {
        return [
            FooService::class,
            BarService::class,
            BigDeperndency::class,
        ];
    }

    private function getService(string $name)
    {
        if (!$this->locator->has($name)) {
            throw new \LogicException('Service not defined');
        }

        return $this->locator->get($name);
    }

    // ...
}
```



```
use App\Entity\User;
use Doctrine\ORM\EntityRepository;
use Symfony\Bridge\Doctrine\Form\Type\EntityType;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;

class UserForm extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder->add('user', EntityType::class, [
            'class' => User::class,
            'choice_label' => 'email',
            'query_builder' => static function (EntityRepository $repo) {
                return $repo->createQueryBuilder('e')
                    ->where('e.createdAt > :date')
                    ->setParameter('date', new \DateTimeImmutable('-1year'));
            },
        ]);
    }
}
```

```
use App\Entity\User;
use Doctrine\ORM\EntityRepository;
use Symfony\Bridge\Doctrine\Form\Type\EntityType;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;

class UserForm extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder->add('user', EntityType::class, [
            'class' => User::class,
            'choice_label' => 'email',
            'query_builder' => static function (EntityRepository $repo) {
                return $repo->createQueryBuilder('e')
                    ->where('e.createdAt > :date')
                    ->setParameter('date',
                        (new \DateTimeImmutable('-1year'))->setTime(0, 0, 0));
            },
        ]);
    }
}
```


215.55 ms **SELECT** u0_.id **AS** id_0, u0_.uuid **AS** uuid_1, u0_.email **AS** email_2, u0_.type **AS** type_4, u0_.locale **AS** locale_5, u0_.enabled **AS** enabled_10, u0_.loginAt **AS** loginAt_11, u0_.roles **AS** roles_12, u0_.banned **AS** banned_13, u0_.createdAt **AS** createdAt_16, u0_.updatedAt **AS** updatedAt_17, u0_.company_id **AS** company_id_18 **FROM** User u0_ **WHERE** u0_.createdAt > ?

Parameters:

[▼
 "2019-10-09 00:00:00"
]

[Hide formatted query](#) [View runnable query](#) [Explain query](#)

```
SELECT
  u0_.id AS id_0,
  u0_.uuid AS uuid_1,
  u0_.email AS email_2,
  u0_.type AS type_4,
  u0_.locale AS locale_5,
  u0_.enabled AS enabled_10,
  u0_.loginAt AS loginAt_11,
  u0_.roles AS roles_12,
  u0_.banned AS banned_13,
  u0_.createdAt AS createdAt_16,
  u0_.updatedAt AS updatedAt_17,
  u0_.company_id AS company_id_18,
FROM
  User u0_
WHERE
  u0_.createdAt > ?
```


215.55 ms

```
SELECT u0_.id AS id_0, u0_.uuid AS uuid_1, u0_.email AS email_2, u0_.type AS type_4, u0_.locale AS locale_5,
u0_.enabled AS enabled_10, u0_.loginAt AS loginAt_11, u0_.roles AS roles_12, u0_.banned AS banned_13, u0_.createdAt
AS createdAt_16, u0_.updatedAt AS updatedAt_17, u0_.company_id AS company_id_18    FROM User u0_ WHERE u0_.create-
dAt > ?
```

Parameters:

[▼

"2019-10-09 00:00:00"

]

[View formatted query](#) [View runnable query](#) [Hide query explanation](#)

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	u0_		ALL					292230	33.33	Using where

CREATE INDEX my_idx ON User (createdAt)

```
215.55 ms  SELECT u0_.id AS id_0, u0_.uuid AS uuid_1, u0_.email AS email_2, u0_.type AS type_4, u0_.locale AS locale_5,
u0_.enabled AS enabled_10, u0_.loginAt AS loginAt_11, u0_.roles AS roles_12, u0_.banned AS banned_13, u0_.createdAt
AS createdAt_16, u0_.updatedAt AS updatedAt_17, u0_.company_id AS company_id_18    FROM User u0_ WHERE u0_.create-
dAt > ?
```

Parameters:

```
[▼
  "2019-10-09 00:00:00"
]
```

[View formatted query](#) [View runnable query](#) [Hide query explanation](#)

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	u0_		ALL					292230	33.33	Using where

```
0.88 ms  SELECT u0_.id AS id_0, u0_.uuid AS uuid_1, u0_.email AS email_2, u0_.type AS type_4, u0_.locale AS locale_5,
u0_.enabled AS enabled_10, u0_.loginAt AS loginAt_11, u0_.roles AS roles_12, u0_.banned AS banned_13, u0_.createdAt
AS createdAt_16, u0_.updatedAt AS updatedAt_17, u0_.company_id AS company_id_18    FROM User u0_ WHERE u0_.create-
dAt > ?
```

Parameters:

```
[▼
  "2019-10-09 00:00:00"
]
```

[View formatted query](#) [View runnable query](#) [Hide query explanation](#)

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	u0_		range	my_idx	my_idx	5		1	100.00	Using index condition


```
use App\Entity\User;
use Doctrine\ORM\EntityRepository;
use Symfony\Bridge\Doctrine\Form\Type\EntityType;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;

class UserForm extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder->add('user', EntityType::class, [
            'class' => User::class,
            'choice_label' => 'email',
            'query_builder' => static function (EntityRepository $repo) {
                return $repo->createQueryBuilder('e')
                    ->where('e.createdAt > :date')
                    ->setParameter('date',
                        (new \DateTimeImmutable('-1year'))->setTime(0, 0, 0));
            },
        ]);
    }
}
```

```
use App\Entity\User;
use Doctrine\ORM\EntityRepository;
use Symfony\Bridge\Doctrine\Form\Type\EntityType;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\Extension\Core\Type\ChoiceType;
use Symfony\Component\Form\FormBuilderInterface;

class UserForm extends AbstractType
{
    /** @var EntityRepository */
    private $userRepo;
    // ...
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $rows = $this->userRepo->createQueryBuilder('e')
            ->select('e.id', 'e.email')
            ->where('e.createdAt > :date')
            ->setParameter('date', (new \DateTimeImmutable('-1year'))->setTime(0, 0, 0))
            ->getQuery()
            ->getArrayResult();

        $choices = [];
        foreach ($rows as $row) {
            $choices[$row['email']] = $row['id'];
        }

        $builder->add('user', ChoiceType::class, [
            'choices' => $choices,
```


n + 1 problem

```
class Manager
{
    public function getProductsForCountry(string $country): array
    {
        $webshop = $this->webshopRepository->createQueryBuilder('e')
            ->where('e.country = :country')
            ->setParameter('country', $country)
            ->getQuery()
            ->getOneOrNullResult();

        return $webshop->getPoducts();
    }
}

$products = $manager->getProductsForCountry('sv');
foreach ($products as $p) {
    echo $p->getName();
}
```


n + 1 problem

```
class Manager
{
    public function getProductsForCountry(string $country): array
    {
        $webshop = $this->webshopRepository->createQueryBuilder('e')
            ->select('e', 'product')
            ->join('e.products', 'product')
            ->where('e.country = :country')
            ->setParameter('country', $country)
            ->getQuery()
            ->getOneOrNullResult();

        return $webshop->getPoducts();
    }
}

$products = $manager->getProductsForCountry('sv');
foreach ($products as $p) {
    echo $p->getName();
}
```

n + 1 problem

14

Tip count

```
class Manager
{
    public function getProductsForCountry(string $country): array
    {
        $products = $this->productRepository->createQueryBuilder('p')
            ->join('p.webshop', 'e')
            ->where('e.country = :country')
            ->setParameter('country', $country)
            ->getQuery()
            ->getOneOrNullResult();

        return $products;
    }
}
```

Data transfer

```
class Product
{
    // ...
    /**
     * @var string
     * @ORM\Column(type="string", length=255)
     */
    private $name;

    /**
     * 1000+ characters
     * @var string
     * @ORM\Column(type="text")
     */
    private $description;

    /**
     * Binary image stored in database
     * @ORM\Column(type="binary")
     */
    private $image;

    // ...
}
```


Data transfer

15

Tip count

```
class Product
{
    // ...
    /**
     * @var string
     * @ORM\Column(type="string", length=255)
     */
    private $name;

    /**
     * @var ProductMetadata
     * @ORM\OneToOne(targetEntity="ProductMetadata", inversedBy="product")
     */
    private $metadata;

    // ...
}
```

Data transfer

15

Tip count

```
class ProductMetadata
{
    // ...
    /**
     * @var Product
     * @ORM\OneToOne(targetEntity="Product", mappedBy="metadata")
     */
    private $product;

    /**
     * 1000+ characters
     * @var string
     * @ORM\Column(type="text")
     */
    private $description;

    /**
     * Binary image stored in database
     * @ORM\Column(type="binary")
     */
    private $image;

    // ...
}
```

16

Tip count

doctrine/orm 2.4: NEW


```
class Product
{
    // ...
    /**
     * @var string
     * @ORM\Column(type="string", length=255)
     */
    private $name;

    /**
     * 1000+ characters
     * @var string
     * @ORM\Column(type="text")
     */
    private $description;

    /**
     * Binary image stored in database
     * @ORM\Column(type="binary")
     */
    private $image;

    // ...
}
```

```
class SimpleProduct
{
    private $name;
    private $category;

    public function __construct($name, $category)
    {
        $this->name = $name;
        $this->category = $category;
    }

    // ...
}
```

16

Tip count

```
class ProductRepository extends EntityRepository
{
    /**
     * @return SimpleProduct[]
     */
    public function getSimpleProducts() : array
    {
        $queryBuilder = $this->createQueryBuilder('p');

        $queryBuilder
            ->select(
                sprintf('NEW %s(p.name, p.category)', SimpleProduct::class)
            )
            ->where('p.featured = :true')
            ->setParameter('true', true);

        return $queryBuilder->getQuery()->getResult();
    }
}
```


Your PHP.ini

```
apc.enable_cli = 1
date.timezone = UTC
session.auto_start = Off
short_open_tag = Off

opcache.interned_strings_buffer = 16
opcache.max_accelerated_files = 20000
opcache.memory_consumption = 256
opcache.validate_timestamps = 0
realpath_cache_size = 4096K
realpath_cache_ttl = 600
```

<https://symfony.com/doc/current/performance.html>

17

Tip count

Frontend?

18

Tip count

Run less code in frontend

Frontend optimisations

19
Tip count

- Small HTML pages, limited cookies
- Only include JS and CSS that are used
- `<link rel="preconnect dns-prefetch" href="fonts.googleapis.com">`
- `<script async src="script.js"></script>`
- HTTP2 & brotli/gzip responses
- `font-display: swap;`
- Optimise & lazy load images, Use CDN
- `cache-control: max-age=31449600,immutable`

20

Tip count

Webpack encore

Chrome's Lighthouse

21
Tip count



@tobiasnyholm



Improve performance

#1: Buy a better computer

#2: Use cache

#3: Run less code

Improve performance

#1: Buy a better computer

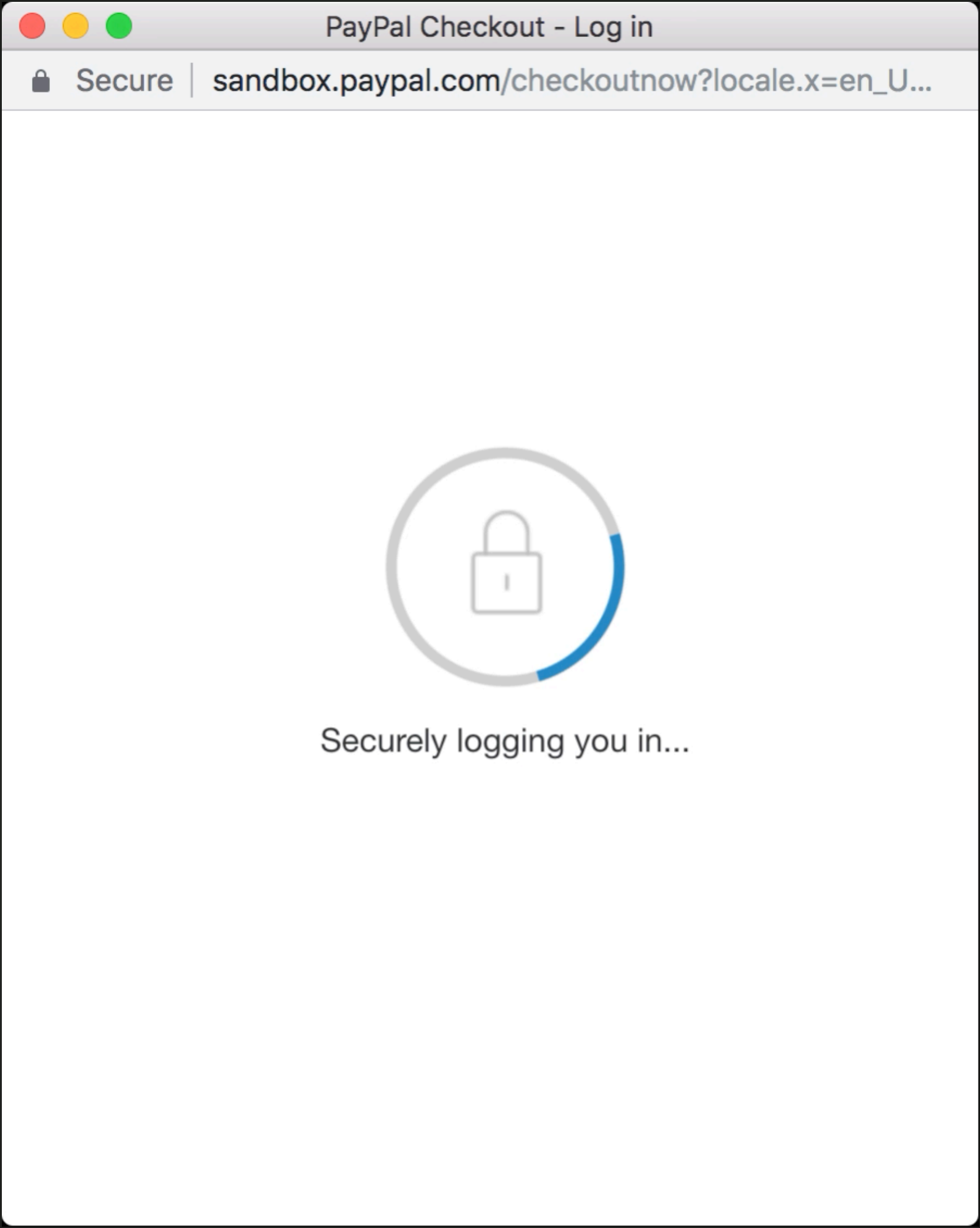
#2: Use cache

#3: Run less code

#4: Lie and cheat

48

Tip count



Tip count

#scandinavia

Friday, October 18th

Last updated less than a minute ago... [Load new messages](#)

Andreas 11:13 AM
Så, hvem skal på SymfonyCon?



📎 | Message #scandinavia

@ 😊

Profile your application



Happyr



1.91 s



23.6 MB

Functions

Recommendations

Assertions

search



Function / Metric

% Excl. ▾

% Incl.

Calls



Symfony\Component\Debug\DebugClassLoader::loadClass



1 177



Doctrine\Common\Annotations\CachedReader::getPropertyAnnotations



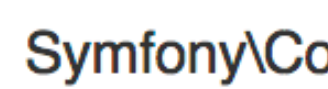
4 941



Composer\Autoload\ClassLoader::findFileWithExtension



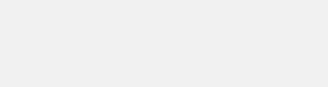
1 623



Symfony\Component\EventDispatcher\ContainerAwareEventDispatcher::sortListeners



90



ContainerD6uysxa\appDevDebugProjectContainer::load



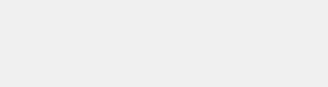
148



Doctrine\ORM\Mapping\Driver\AnnotationDriver::loadMetadataForClass



49



Symfony\Component\Cache\Adapter\ApcuAdapter::getItem



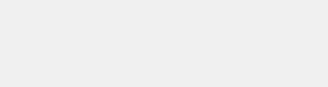
2 651



Doctrine\Common\Annotations\CachedReader::getLastModification



1 325



filemtime



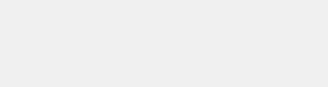
5 740



Symfony\Component\VarDumper\Cloner\VarCloner::doClone



16



Symfony\Component\Cache\DoctrineProvider::doFetch



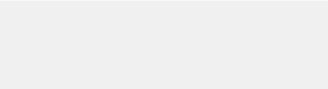
2 651



Symfony\Component\Config\Resource\DirectoryResource::isFresh



62



Symfony\Component\Cache\Adapter\TraceableAdapter::getItem



2 651



Doctrine\Common\Annotations\CachedReader::getLastModification@1



1 254



Symfony\Component\Config\Resource\SelfCheckingResourceChecker::isFresh



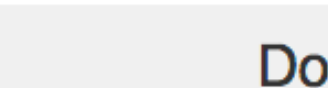
1 993



Symfony\Component\VarDumper\Cloner\VarCloner::castObject



2 421



Doctrine\Common\Annotations\CachedReader::getTraitLastModificationTime



1 594



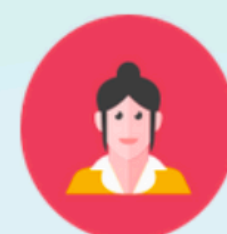
Symfony\Component\Debug\DebugClassLoader::loadClass@1



338

HTTP Pipeline - as a Restaurant

Client



Customer

Server



Restaurant

Request



Order

Response



Food

Application



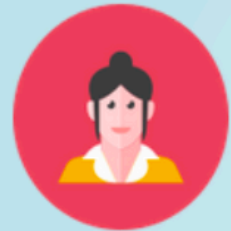
Chef

HTTP Daemon



Waiter/Waitress

Restaurant PHP



Customer enters restaurant



Waitress takes order



Waitress **creates chef** and gives order



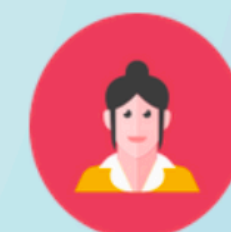
Chef makes food



Waitress gives food to customer



Waitress **brutally murders** chef




```
<?php  
  
require_once dirname(__FILE__) . '/../vendor/autoload_runtime.php';  
  
use App\Kernel;  
  
return function (array $context) {  
    return new Kernel($context[ 'APP_ENV' ], (bool) $context[ 'APP_DEBUG' ] );  
};
```

<https://github.com/php-runtime/runtime>



FRANKEN

PHP

Memory leaks

< 10ms

< 5ms

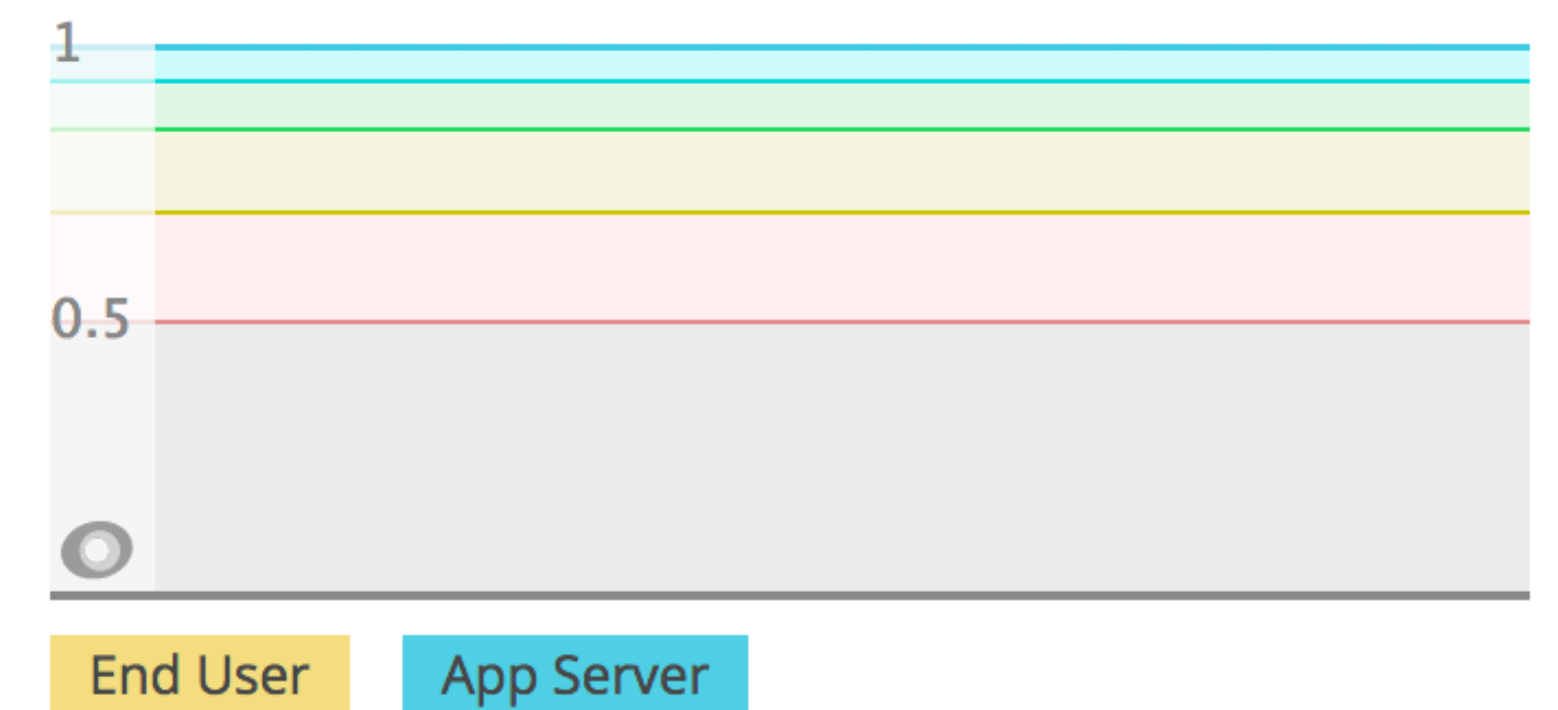
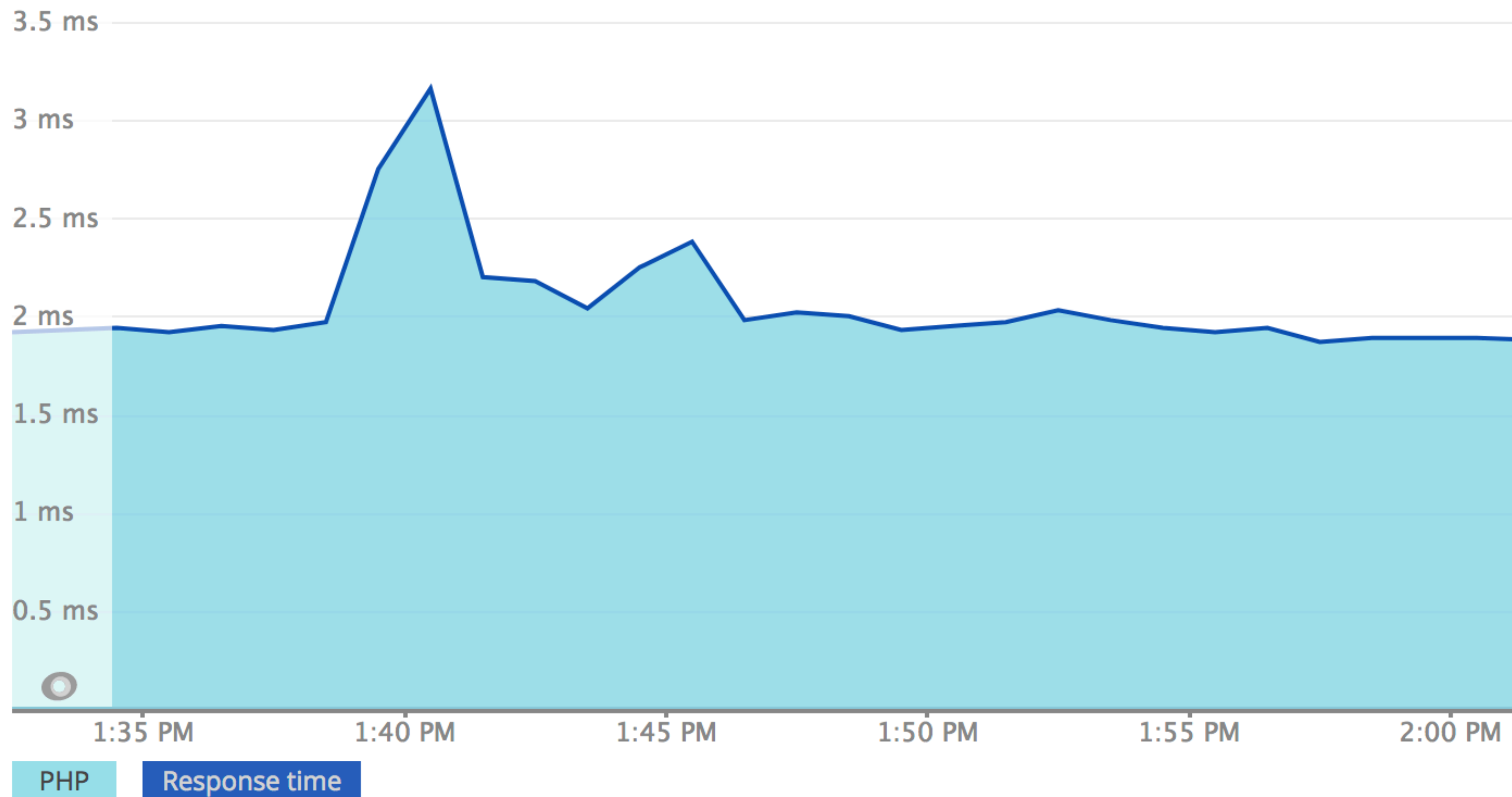
Web transactions time ▾

3.2 ms
APP SERVER

0 s
BROWSER

Apdex score ?

1.0 [0.5] NS [7.0]*
APP SERVER BROWSER



Throughput

9.59 rpm
AVERAGE

