pixelart

CACHING AND PERFORMANCE

@ViennaPHP / 2016-03-23

Created by @PatrikKarisch

pixelart

HI!

pixelart

I'M PATRIK

pixelart

PHP BACKEND DEV AT PIXELART

pixelart

PHP SINCE 13 YEARS

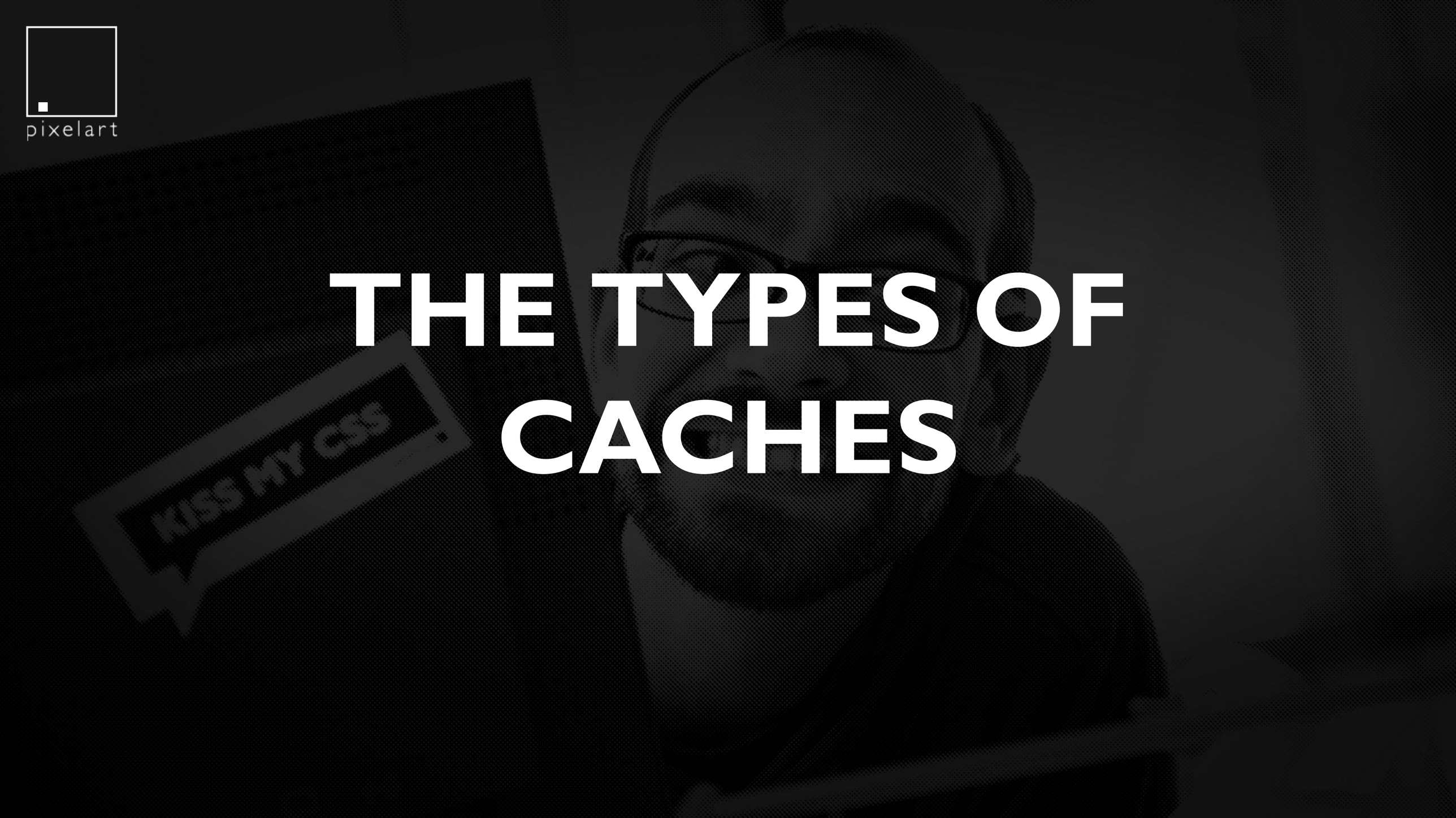
pixelart

SYMFONY SINCE 7 YEARS

pixelart

pixelart

THE TYPES OF CACHES



pixelart

APPLICATION CACHES

GATEWAY CACHES

PROXY CACHES

pixelart

BROWSER CACHES

pixelart

APPLICATION CACHES

SERVER-SIDE

pixelart

FULL CONTROL

pixelart

CACHE DATA PARTIALLY

FOR SINGLE REQUEST

FOR SINGLE INSTANCE

BETWEEN ALL NODES

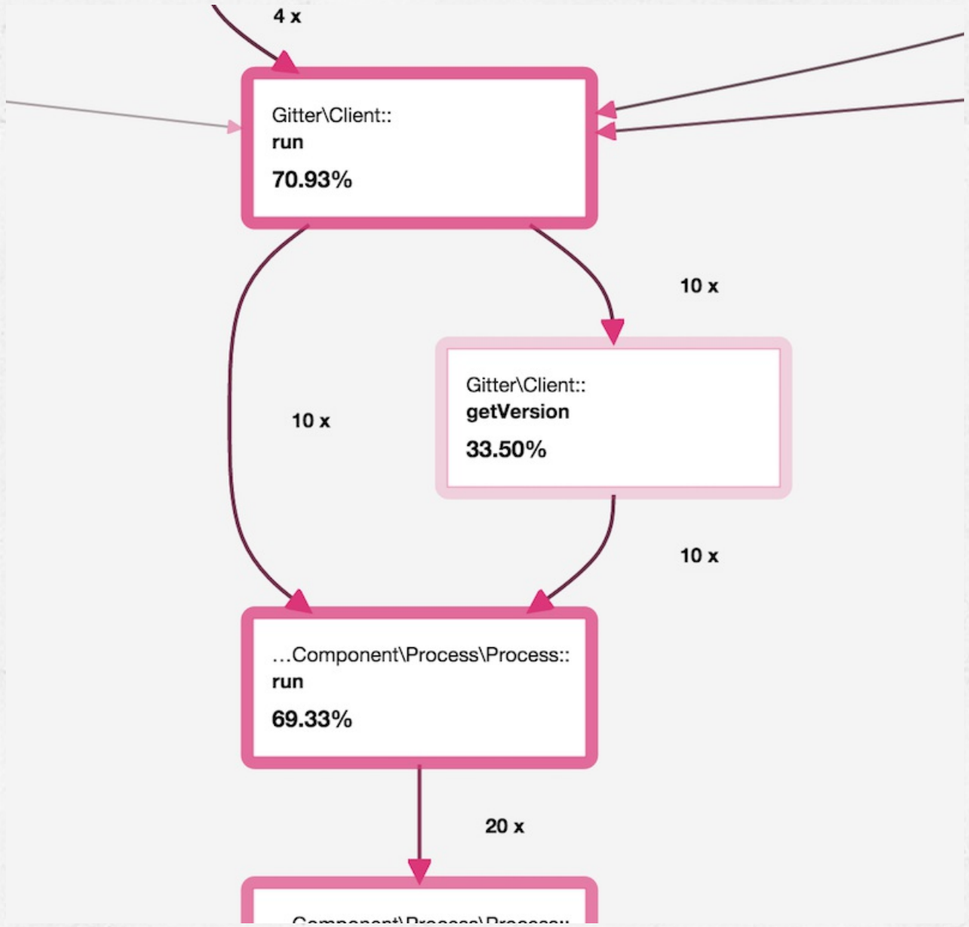
pixelart

SIMPLE STATIC

```
public function getVersion()
{
    $process = new Process($this->getPath() . ' --version');
    $process->run();

    if (!$process->isSuccessful()) {
        throw new \RuntimeException($process->getErrorOutput());
    }

    $version = substr($process->getOutput(), 12);
    return trim($version);
}
```

from 24 Days of Blackfire

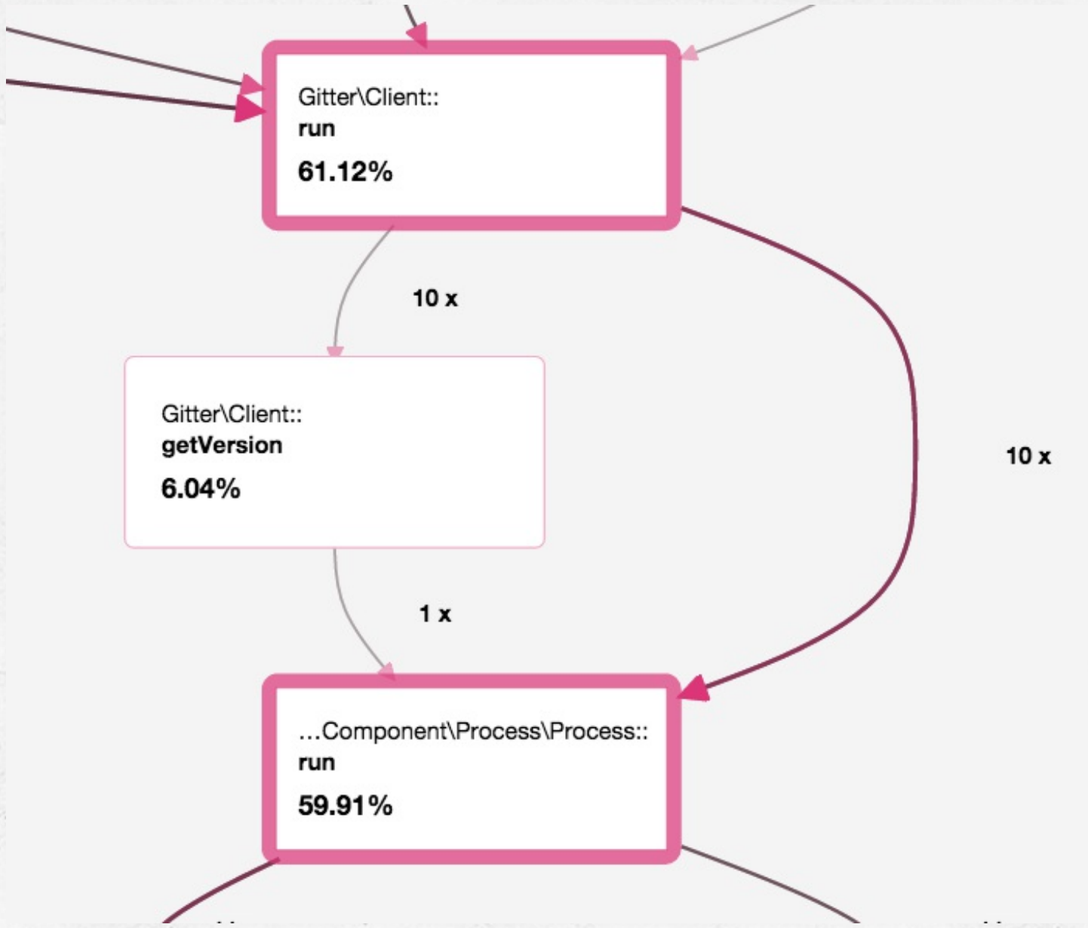
PROFILE BEFORE



APPLIED STATIC

```
public function getVersion()
{
    static $version;

    if (null !== $version) {
        return trim($version);
    }

    $process = new Process($this->getPath() . ' --version');
    $process->run();

    if (!$process->isSuccessful()) {
        throw new \RuntimeException($process->getErrorOutput());
    }

    $version = substr($process->getOutput(), 12);
    return trim($version);
}
```

PROFILE AFTER



pixelart

USE THE LIB, LUKE

```
$ composer require doctrine/cache
```

```
public function getVersion()
{
    $cacheKey = sha1(__CLASS__ . __METHOD__ . 'version');
    if ($this->cache->contains($cacheKey) {
        return $this->cache->get($cacheKey);
    }

    // ..

    $this->cache->save($cacheKey, $version);

    return $version;
}
```

pixelart

PER REQUEST OR FOR TESTING

```
use Doctrine\Common\Cache\ArrayCache;

$app['cache'] = $app->share(function () {
    return new ArrayCache();
});
```

pixelart

PER INSTANCE

```
use Doctrine\Common\Cache\ApcCache;

$app['cache'] = $app->share(function () {
    return new ApcCache();
});
```

```
use Doctrine\Common\Cache\XcacheCache;

$app['cache'] = $app->share(function () {
    return new XcacheCache();
});
```

pixelart

FOR ALL NODES

```
use Doctrine\Common\Cache\MemcachedCache;

$app['cache'] = $app->share(function ($app) {
    $cache = new MemcachedCache();
    $cache->setMemcached($app['memcached_connection']);

    return $cache;
});
```

```
use Doctrine\Common\Cache\RedisCache;

$app['cache'] = $app->share(function ($app) {
    $cache = new RedisCache();
    $cache->setRedis($app['redis_connection']);

    return $cache;
});
```

pixelart

USING DOCTRINE ORM?

```
use Doctrine\Common\Cache\ApcCache;
use Doctrine\ORM\Configuration;

$config = new Configuration();
$config->setQueryCacheImpl(new ApcCache());
$config->setResultCacheImpl(new \Doctrine\Common\Cache\ApcCache());
$config->setMetadataCacheImpl(new \Doctrine\Common\Cache\ApcCache());
```

pixelart

WITH SYMFONY

```
#app/config.yml

doctrine:
#...
    orm:
        # ...
        default_entity_manager: default
        entity_managers:
            default:
                metadata_cache_driver: apc
                result_cache_driver: apc
                query_cache_driver: apc
                #...
```

pixelart

THE RESULT CACHE

```
$query = $em->createQuery('select u from \Entities\User u');
$query->useResultCache(true);

// set the lifetime
$query->setResultCacheLifetime(3600);

// set the id to use instead of generated one
$query->setResultCacheId('my_custom_id');

// or all in one call
$query->useResultCache(true, 3600, 'my_custom_id');
```

WHAT TO CACHE

HEAVY COMPUTATIONS

pixelart

HEAVY COMPUTATIONS

process calls as seen before

pixelart

HEAVY COMPUTATIONS

process calls as seen before

menu structures

pixelart

HEAVY COMPUTATIONS

process calls as seen before

menu structures

rendered partials

SQL QUERIES

SQL QUERIES

fat joins

pixelart

SQL QUERIES

fat joins

slow queries

pixelart

SQL QUERIES

fat joins

slow queries

often called queries

NETWORK REQUESTS

pixelart

NETWORK REQUESTS

API to data which hardly changes

pixelart

NETWORK REQUESTS

API to data which hardly changes or with lower lifetime to data APIs

pixelart

NETWORK REQUESTS

API to data which hardly changes or with lower lifetime to data APIs

Google Geocoding requests ;)

pixelart

PARSED CONFIG

YAML

XML

Annotations

etc.

pixelart

COMPILE STUFF TO PHP

pixelart

COMPILE STUFF TO PHP

Twig templates

pixelart

COMPILE STUFF TO PHP

Twig templates

Symfony DI container

pixelart

COMPILE STUFF TO PHP

Twig templates

Symfony DI container

Symfony routing

pixelart

COMPILE STUFF TO PHP

Twig templates
Symfony DI container
Symfony routing
Symfony translations

pixelart

SYMFONY ROUTING

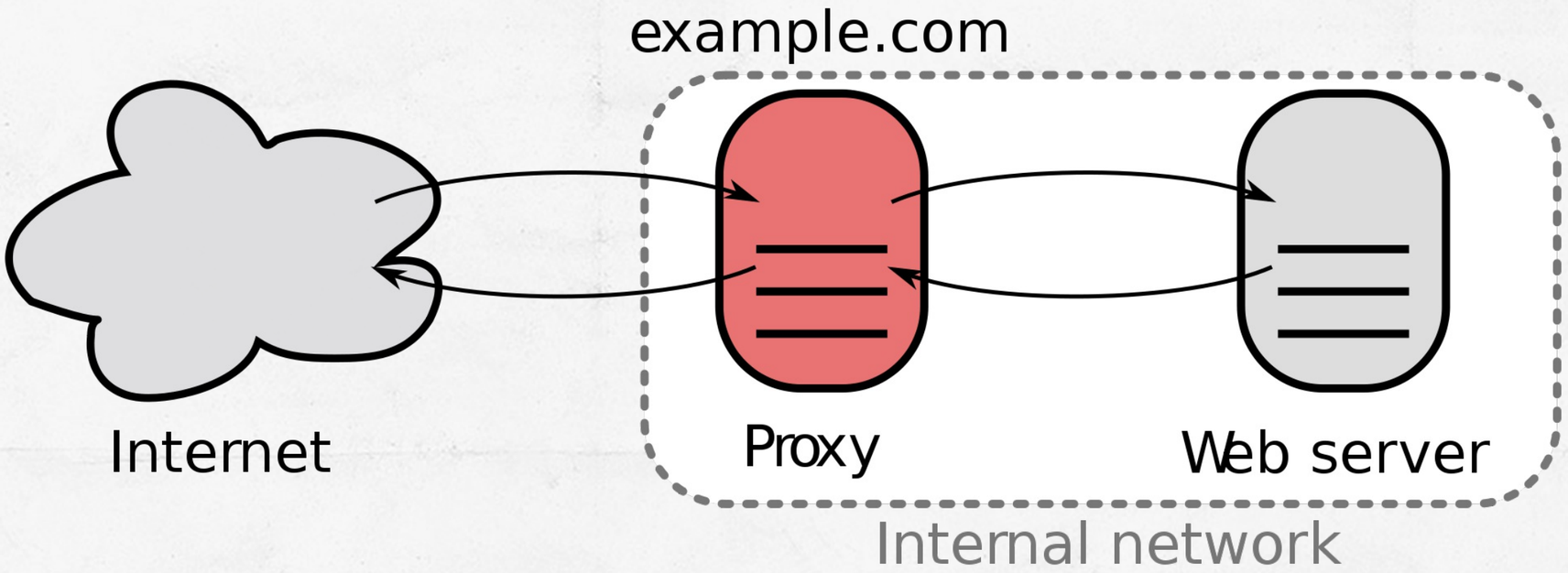
```
class DefaultController extends Controller
{
    /**
     * @Route("/home", name="homepage")
     */
    public function homeAction(Request $request)
}
```

```
public function match($pathinfo)
{
    // stuff..

    // homepage
    if ($pathinfo === '/home') {
        return [
            '_controller' => 'AppBundle\\Controller\\DefaultController:
            '_route' => 'homepage',
        ];
    }

    throw 0 < count($allow)
        ? new MethodNotAllowedException(array_unique($allow))
        : new ResourceNotFoundException();
}
```

pixelart

GATEWAY CACHES

SERVER-SIDE



pixelart

EDGE-SIDE-INCLUDE (ESI)

pixelart

COMPLEX INVALIDATION

pixelart

COMPLEX INVALIDATION

"There are only two hard things in Computer Science: cache invalidation and naming things." - Phil Karlton

pixelart

VARNISH

NGINX

pixelart

SYMFONY HTTPCACHE

USAGE

pixelart

HTTP-HEADERS

like browser cache

pixelart

EXTRA HTTP-REQUESTS

for banning and flushing

pixelart

USE THE LIB, LUKE

FOS HttpCache library

```
$ composer require friendsofsymfony/http-cache
```

pixelart

COMMON MISTAKES

DIDN'T CLEANUP COOKIES

pixelart

CACHE USER SPECIFIC CONTENT

pixelart

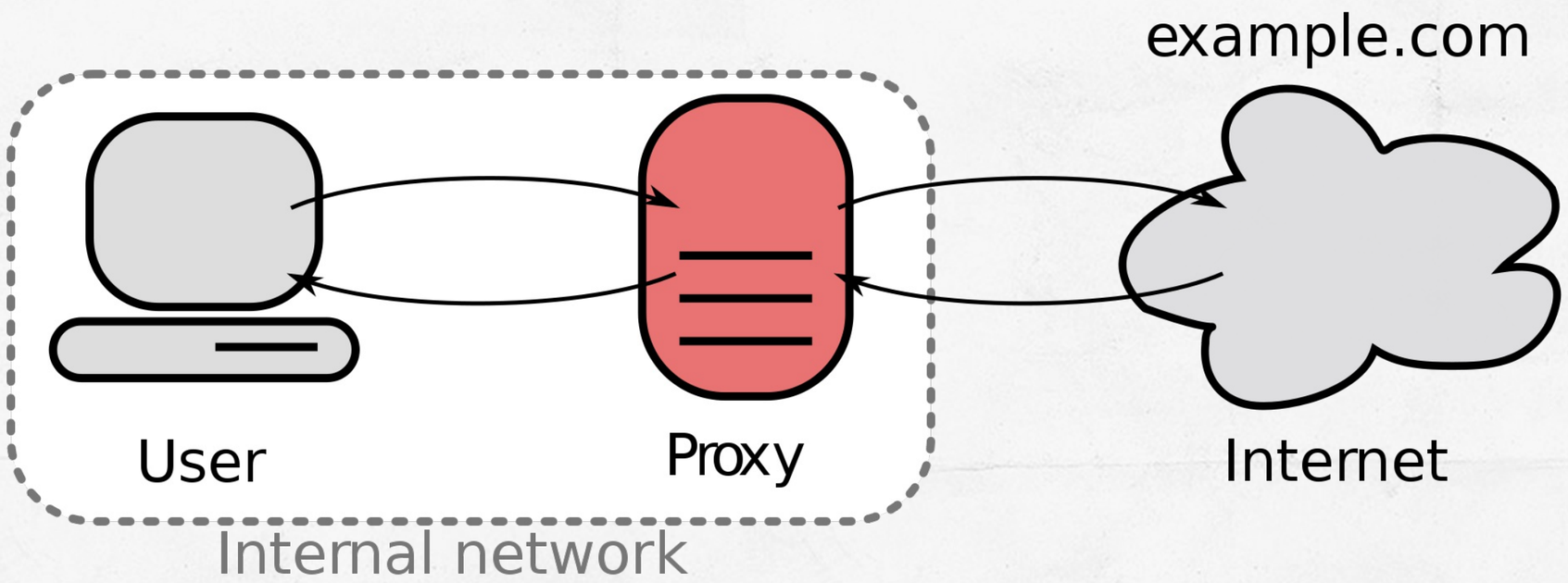
WRONG CACHE HEADERS

UNSAFE REQUESTS

pixelart

PROXY CACHES

CLIENT/ISP-SIDE



pixelart

LIKE BROWSER CACHES

BUT SHARED

e.g from the ISP or a corporate proxy

pixelart

THE MOST ANNOYING ONE

pixelart

BEHAVES NOT EVERYTIME LIKE THE STANDARD SAYS

pixelart

BROWSER CACHES

pixelart

LESS REQUESTS AND TRAFFIC

CLIENT-SIDE

NO CONTROL

NO INVALIDATION

SIMPLE LIFETIME

pixelart

OR WITH VALIDATION

pixelart

OR BOTH

pixelart

EXPIRATION

```
Cache-Control: max-age=1800, s-maxage=600
```

pixelart

EXPIRATION

```
public function fooAction(/* .. */)
{
    // ...

    $response = $this->render(/* .. */);
    $response->setPublic(); // make sure the response is public/cacheabl
    $response->setMaxAge(1800);
    $response->setSharedMaxAge(600);

    return $response;
}
```

VALIDATION WITH ETAG

```
ETag: 1456468486
```

VALIDATION WITH ETAG

```
ETag: 1456468486
```

```
If-None-Match: 1456468486

304 Not Modified
```

pixelart

VALIDATION WITH ETAG

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

public function fooAction(Request $request, /* .. */)
{
    // Get the minimum information to compute the ETag
    $foo = ...;

    $response = new Response();
    $response->setPublic(); // make sure the response is public/cacheable
    $response->setETag($foo->computeETag());

    // Check that the Response is not modified for the given Request
    if ($response->isNotModified($request)) {
        // return the 304 Response immediately
        return $response;
    }

    // render the stuff

    return $response;
}
```

pixelart

VALIDATION WITH LAST-MODIFIED

```
Last-Modified: Wed, 27 January 2015 19:30:20 GMT
```

VALIDATION WITH LAST-MODIFIED

```
Last-Modified: Wed, 27 January 2015 19:30:20 GMT
```

```
If-Modified-Since: Wed, 27 January 2015 19:30:20 GMT

304 Not Modified
```

VALIDATION WITH LAST-MODIFIED

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

public function fooAction(Request $request, /* .. */)
{
    // Get the minimum information
    $foo = ...;

    $response = new Response();
    $response->setPublic(); // make sure the response is public/cacheable
    $response->setLastModified($foo->getUpdatedAt());

    // Check that the Response is not modified for the given Request
    if ($response->isNotModified($request)) {
        // return the 304 Response immediately
        return $response;
    }

    // render the stuff
```

VALIDATION WITH BOTH

```
ETag: 1456468486
Last-Modified: Wed, 27 January 2015 19:30:20 GMT
```

VALIDATION WITH BOTH

```
ETag: 1456468486
Last-Modified: Wed, 27 January 2015 19:30:20 GMT
```

```
If-None-Match: 1456468486
If-Modified-Since: Wed, 27 January 2015 19:30:20 GMT

304 Not Modified
```

pixelart

VALIDATION WITH BOTH

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

public function fooAction(Request $request, /* .. */)
{
    // Get the minimum information to compute the ETag
    $foo = ...;

    $response = new Response();
    $response->setPublic(); // make sure the response is public/cacheable
    $response->setETag($foo->computeETag());
    $response->setLastModified($foo->getUpdatedAt());

    // Check that the Response is not modified for the given Request
    if ($response->isNotModified($request)) {
        // return the 304 Response immediately
        return $response;
    }

    // render the stuff

    return $response;
}
```

EXPIRATION AND VALIDATION

```
Cache-Control: max-age=1800, s-maxage=600
ETag: 1456468486
Last-Modified: Wed, 27 January 2015 19:30:20 GMT
```

pixelart

EXPIRATION AND VALIDATION

```
Cache-Control: max-age=1800, s-maxage=600
ETag: 1456468486
Last-Modified: Wed, 27 January 2015 19:30:20 GMT
```

Only if max age is reached and entry is stale:

pixelart

EXPIRATION AND VALIDATION

```
Cache-Control: max-age=1800, s-maxage=600
ETag: 1456468486
Last-Modified: Wed, 27 January 2015 19:30:20 GMT
```

Only if max age is reached and entry is stale:

```
If-None-Match: 1456468486
If-Modified-Since: Wed, 27 January 2015 19:30:20 GMT

304 Not Modified
```

pixelart

TAKE CARE OF

PUBLIC/PRIVATE

pixelart

PUBLIC BY DEFAULT

```
Cache-Control: max-age=1800, s-maxage=600
```

PUBLIC IS CACHED IN PROXY AND GATEWAY CACHES

pixelart

PRIVATE ONLY IN THE BROWSER CACHE

```
Cache-Control: private, max-age=1800, must-revalidate
```

LIFETIME

pixelart

TEMPLATE CHANGES ON ETAG AND LAST-MODIFIED CALCULATION

pixelart

THE CUSTOMER

pixelart

"WHY IS THE SITE NOT CHANGED?"

pixelart

"NO WAY! THAT IS TOO LONG!"

for blog sites with little to no changes..

pixelart

"I NEED THE CHANGE IMMEDIATELY!"

USE CASES

pixelart

BLOG/NEWS PAGES

pixelart

BLOG/NEWS PAGES

without comments

pixelart

BLOG/NEWS PAGES

without comments

long lifetime (e.g. 1 day)

pixelart

BLOG/NEWS PAGES

without comments

long lifetime (e.g. 1 day)

and validation

HOMEPAGE

pixelart

HOMEPAGE

mixed static contents

pixelart

HOMEPAGE

mixed static contents

e.g. 1 hour

pixelart

HOMEPAGE

mixed static contents

e.g. 1 hour

possible validation

MAP SEARCH

pixelart

MAP SEARCH

lifetime depends on the possible result objects (e.g. retailer)

pixelart

MAP SEARCH

lifetime depends on the possible result objects (e.g. retailer)

use validation

ARTICLE SEARCH, OTHER FILTERS

pixelart

ARTICLE SEARCH, OTHER FILTERS

no caching (too much combinations)

pixelart

ARTICLE SEARCH, OTHER FILTERS

no caching (too much combinations)
but validation too save bandwidth

pixelart

Q & A

pixelart

THANKS

www.karisch.guru

twitter.com/@PatrikKarisch