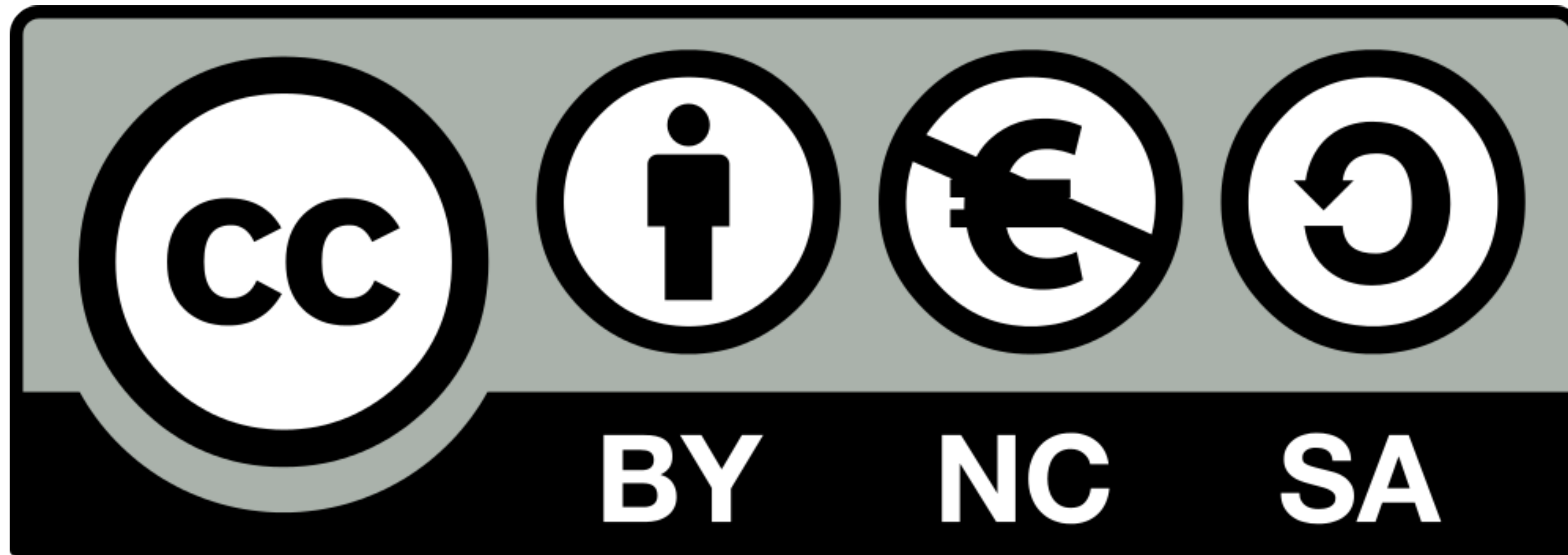


NEW SYMFONY TIPS & TRICKS

SymfonyCon
Paris - December 4, 2015

Javier Eguiluz

License of this presentation

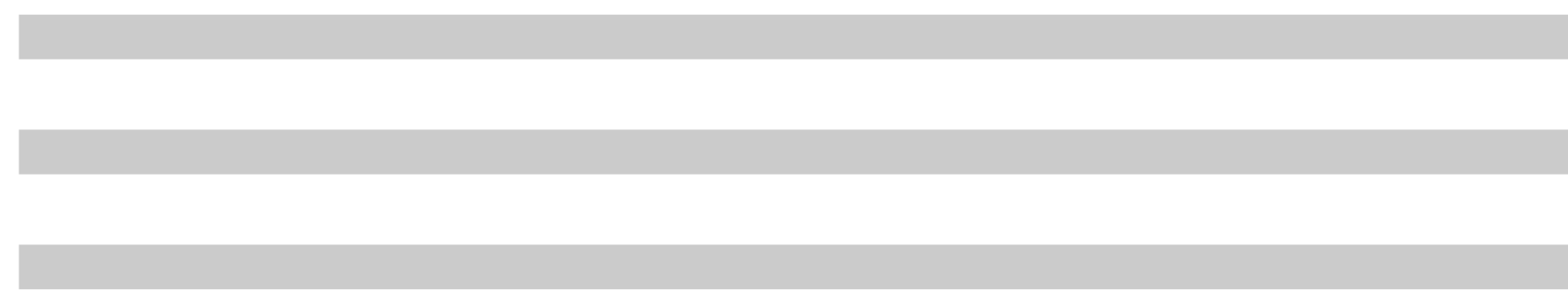


creativecommons.org/licenses/by-nc-sa/3.0

About me



Javier Eguiluz



evangelist and trainer at

SensioLabs

Créateur de  Symfony

**About
this talk**

What we'll talk about

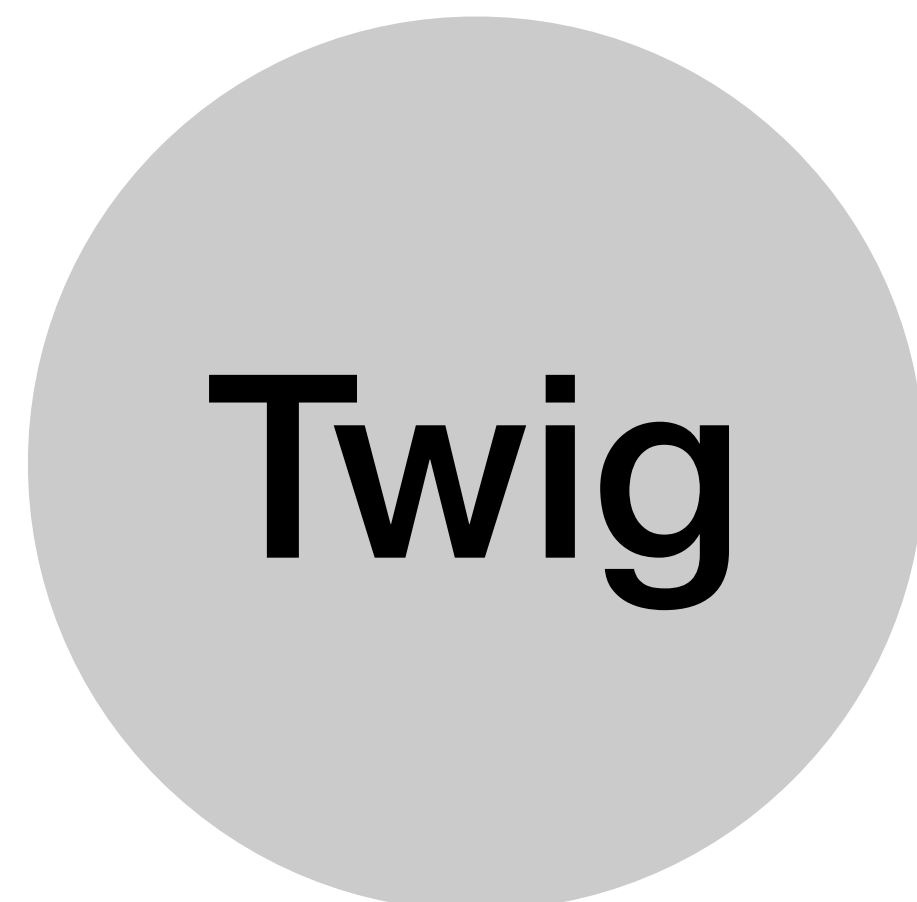
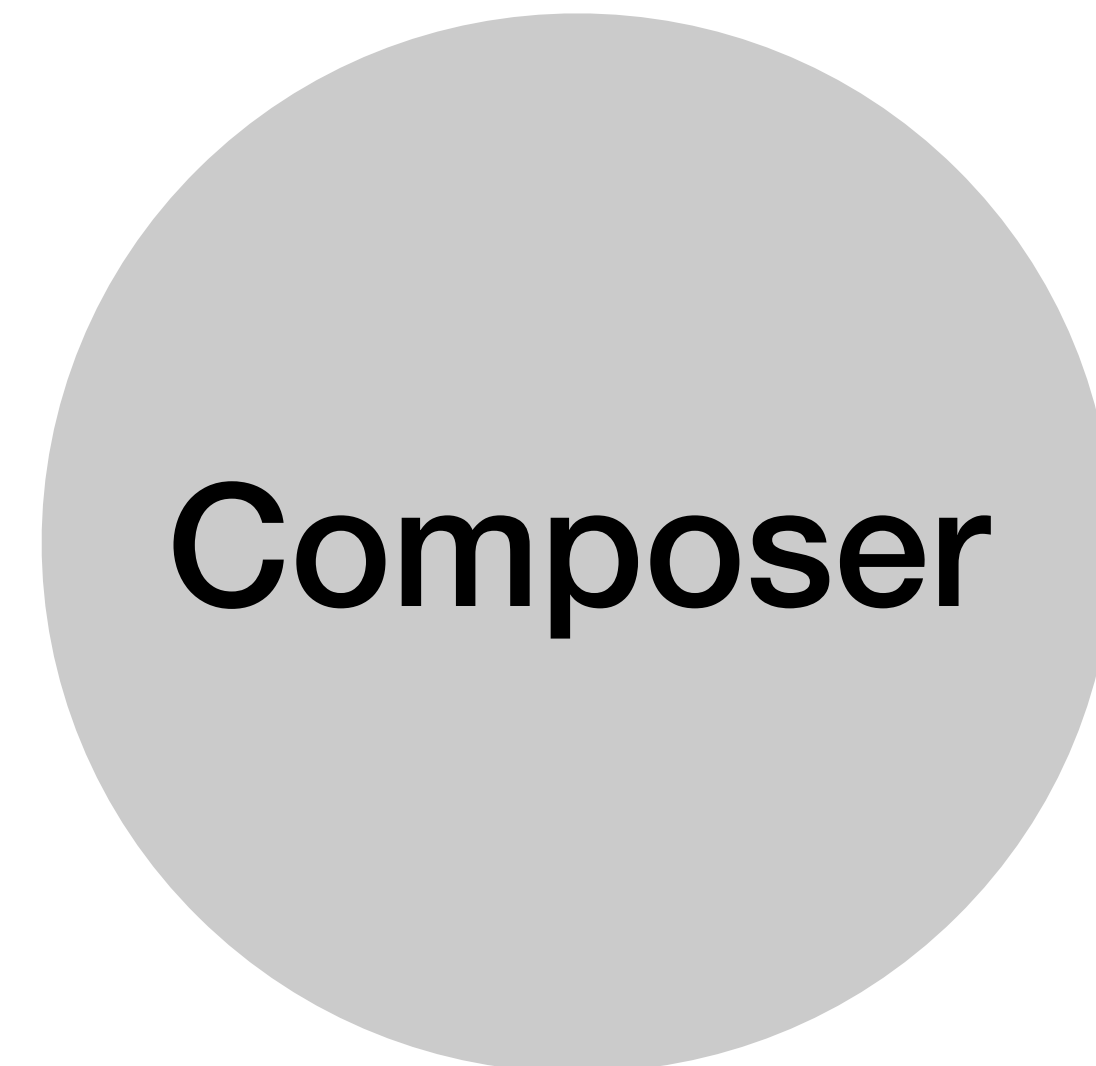
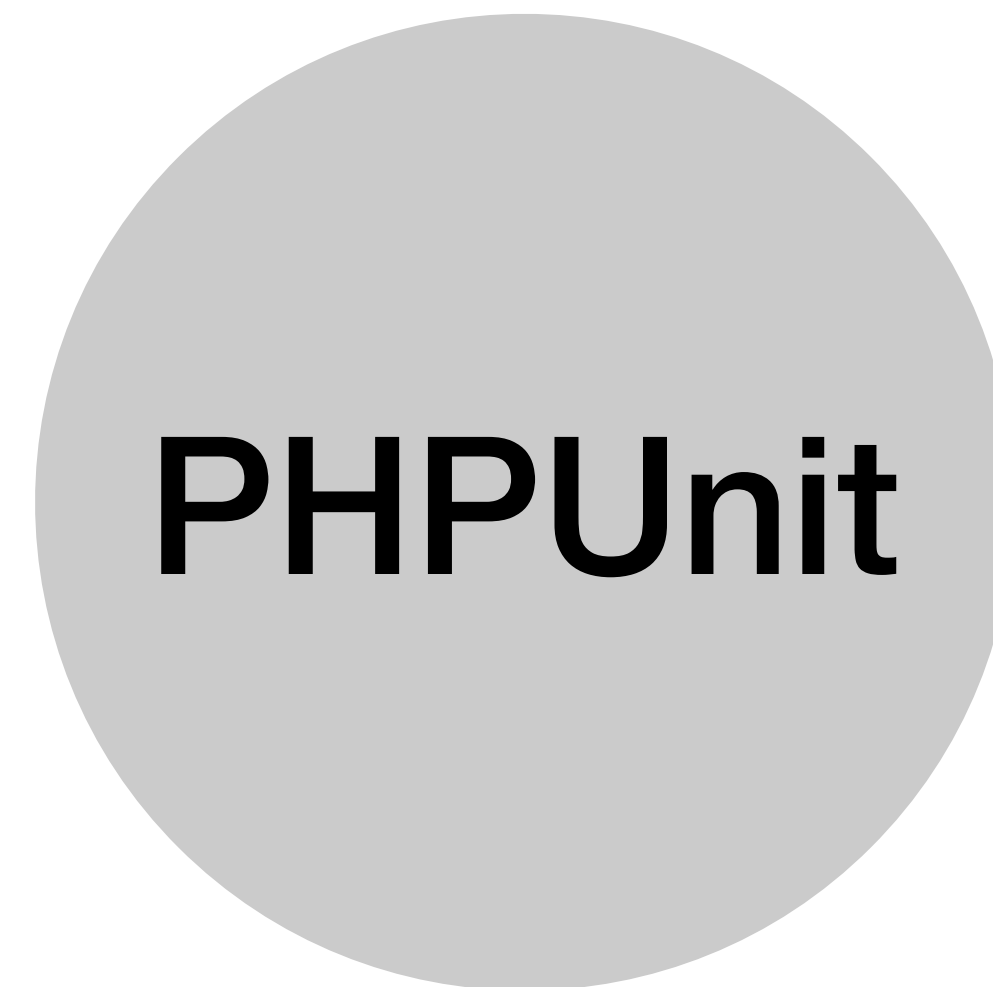
Tips &
tricks

New
features

Rarely used
options



Symfony



**Before we
start**

Thanks to my awesome co-workers for sharing their knowledge!



Grégoire



Nicolas



Hugo



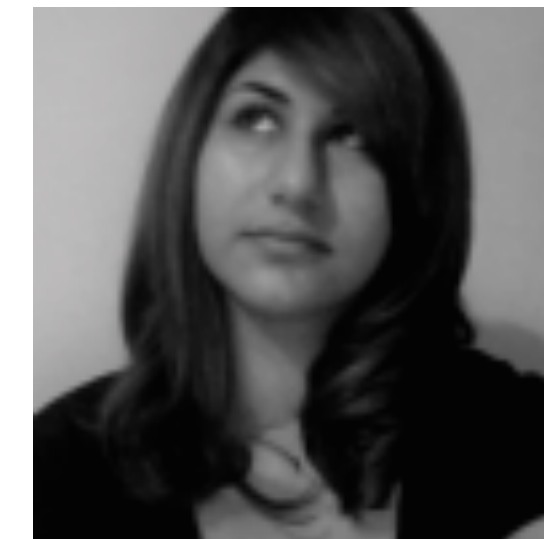
Charles



Tugdual



Julien



Sarah



Jérémie



Jérémie



Romain



FX

And thanks to the amazing Symfony Community too



Christophe



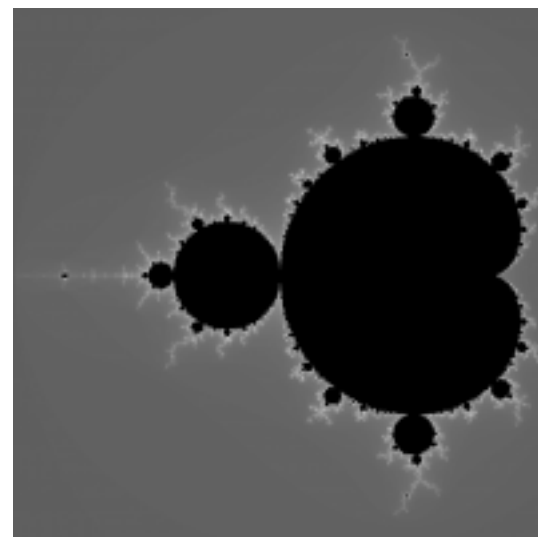
Benjamin



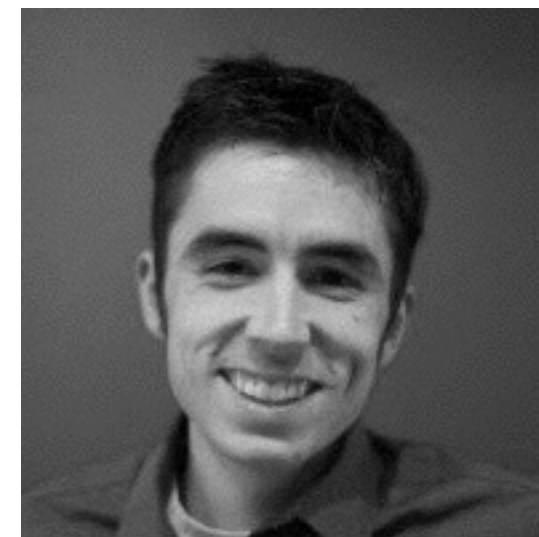
Tobias



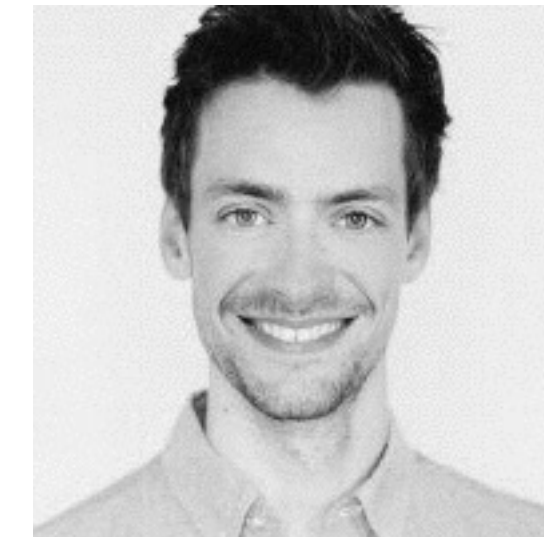
Wouter



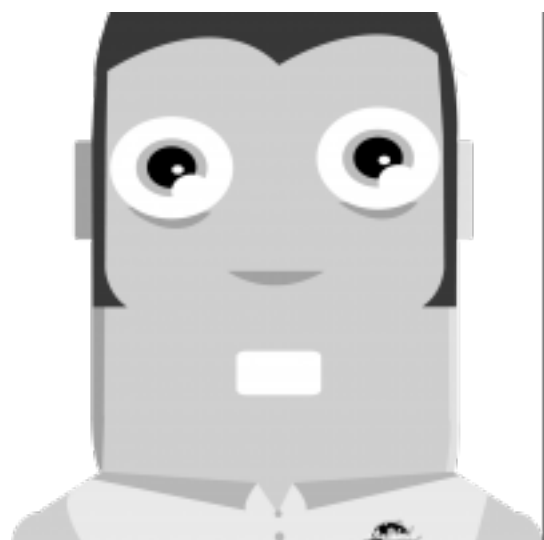
Christian



Ryan



Bernhard



Kevin



Jordi



Jakub



Lukas

Agenda

Doctrine

Composer

Twig

Config

Monolog

Security

Tests

Services

Console

mini-tip

Accessing request parameters

```
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\Request;

/** @Route("/blog") */
class BlogController extends Controller
{
    /** @Route("/", name="blog_index") */
    public function indexAction()
    {
        $value1 = $request->query->get('parameter1');
        $value2 = $request->request->get('parameter2');

        // ...
    }
}
```


Accessing request parameters

```
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\Request;
```

```
/** @Route("/blog") */
class BlogController extends Controller
{
    /** @Route("/", name="blog_index") */
    public function indexAction()
    {
        $value1 = $request->query->get('parameter1');
        $value2 = $request->request->get('parameter2');

        // ...
    }
}
```

everybody uses the **get()** method,
but there are other useful methods



ParameterBag defines some useful methods

```
// very useful methods
```

```
$pageNumber = $request->query->getInt('page');
```

```
$isPublished = $request->query->getBoolean('published');
```


ParameterBag defines some useful methods

```
// very useful methods
```

```
$pageNumber = $request->query->getInt('page');
```

```
$isPublished = $request->query->getBoolean('published');
```

```
// 'ABC+12.34DEF' -> 'ABCDEF'
```

```
$request->query->getAlpha('parameter');
```

```
// 'ABC+12.34DEF' -> 'ABC1234DEF'
```

```
$request->query->getAlnum('parameter');
```

```
// 'ABC+12.34DEF' -> '1234'
```

```
$request->query->getDigits('parameter');
```

Doctrine

Naming strategies

I learned this from



Ruud
Bijnen

More information

<http://doctrine-orm.readthedocs.org/projects/doctrine-orm/en/latest/reference/namingstrategy.html>

Setting custom table/properties names

```
namespace AppBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/** @ORM\Table(name="api_users") */
class ApiUsers
{
    /** @ORM\Column(type="string", name="api_token") */
    private $apiToken;

    /** @ORM\Column(type="datetime", name="created_at") */
    private $createdAt;
}
```

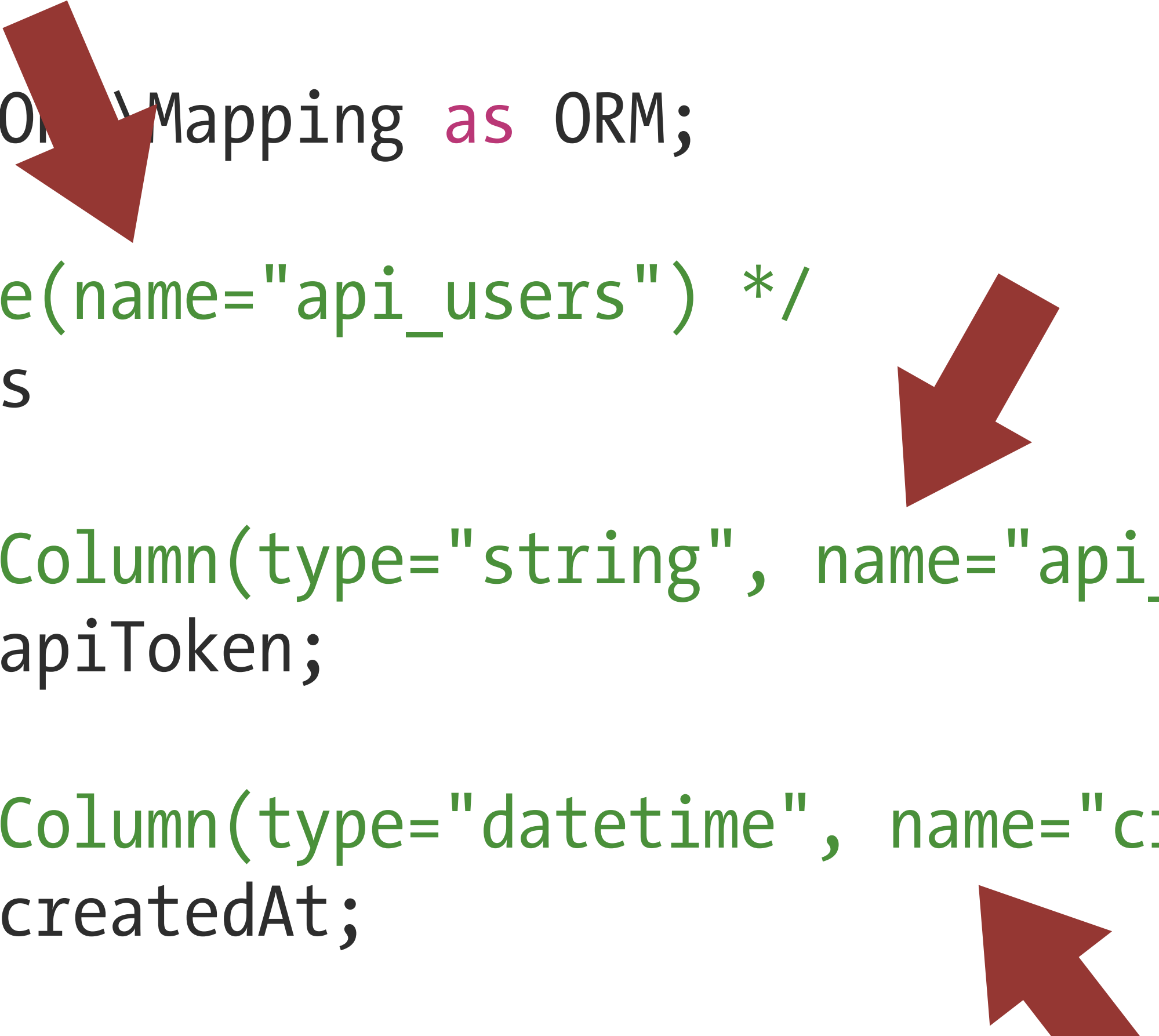
Setting custom table/properties names

```
namespace AppBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/** @ORM\Table(name="api_users") */
class ApiUsers
{
    /** @ORM\Column(type="string", name="api_token") */
    private $apiToken;

    /** @ORM\Column(type="datetime", name="created_at") */
    private $createdAt;
}
```



Setting custom table/properties names

```
namespace AppBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/** @ORM\Table(name="api_users") */
class ApiUsers
{
    /** @ORM\Column(type="string", name="api_token") */
    private $apiToken;

    /** @ORM\Column(type="datetime", name="created_at") */
    private $createdAt;
}
```

Entity

ApiUsers

apiToken

createdAt

Database

api_users

api_token

created_at



Using a built-in Doctrine naming strategy


```
namespace AppBundle;

use Doctrine\ORM\Mapping as ORM;

/** @ORM\Table */
class ApiUsers
{
    /** @ORM\Column(type="string") */
    private $apiToken;

    /** @ORM\Column(type="datetime") */
    private $createdAt;
}
```

```
# app/config/config.yml
doctrine:
    dbal:
        # ...
    orm:
        naming_strategy: doctrine.orm.naming_strategy.underscore
```

 this is automatically translated into **api_token**

Defining a custom naming strategy

```
namespace Doctrine\ORM\Mapping;
```

```
interface NamingStrategy
```

```
{
```

```
    function classToTableName($className);
```

```
    function propertyToColumnName($propertyName);
```

```
    function referenceColumnName();
```

```
    function joinColumnName($propertyName);
```

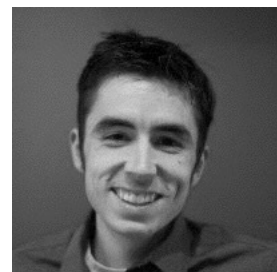
```
    function joinTableName($sourceEntity, $targetEntity, $propertyName);
```

```
    function joinKeyColumnName($entityName, $referencedColumnName);
```

```
}
```


Abstract relationships

I learned this from

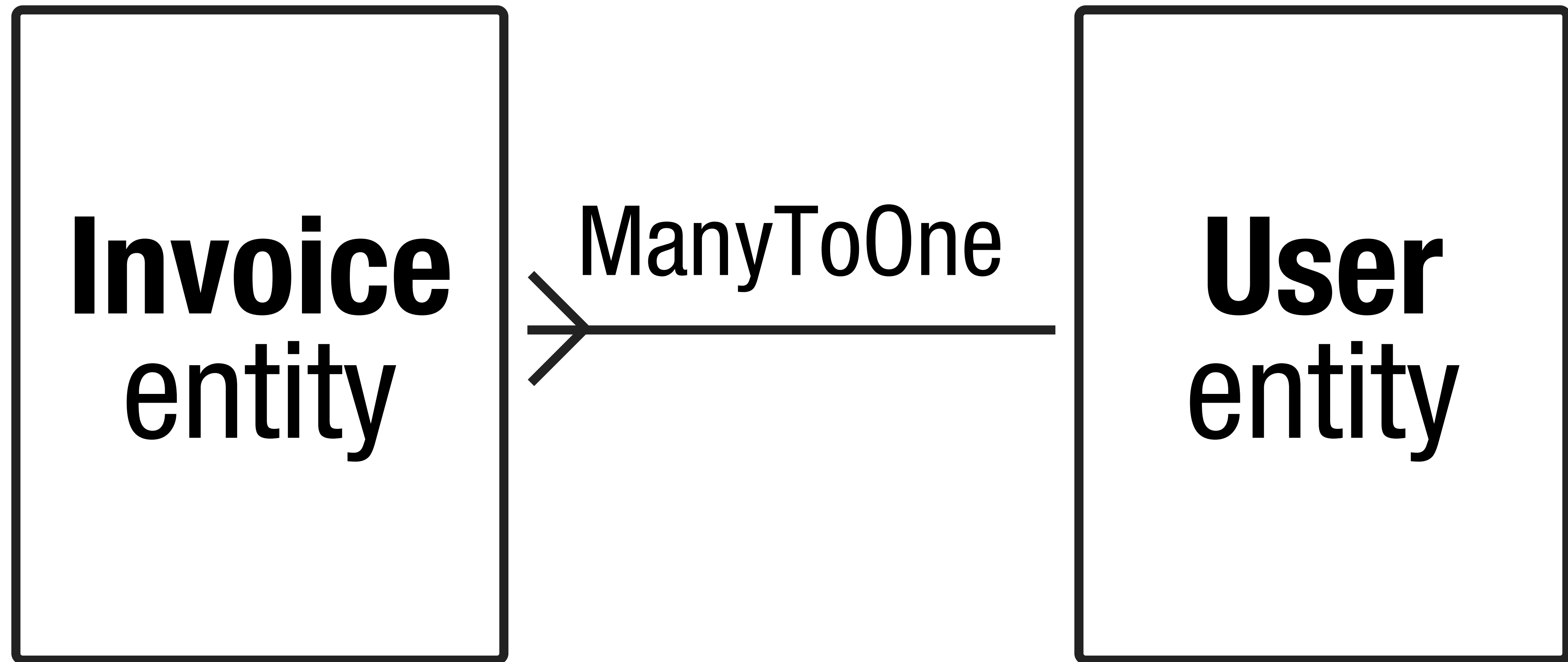


Ryan
Weaver

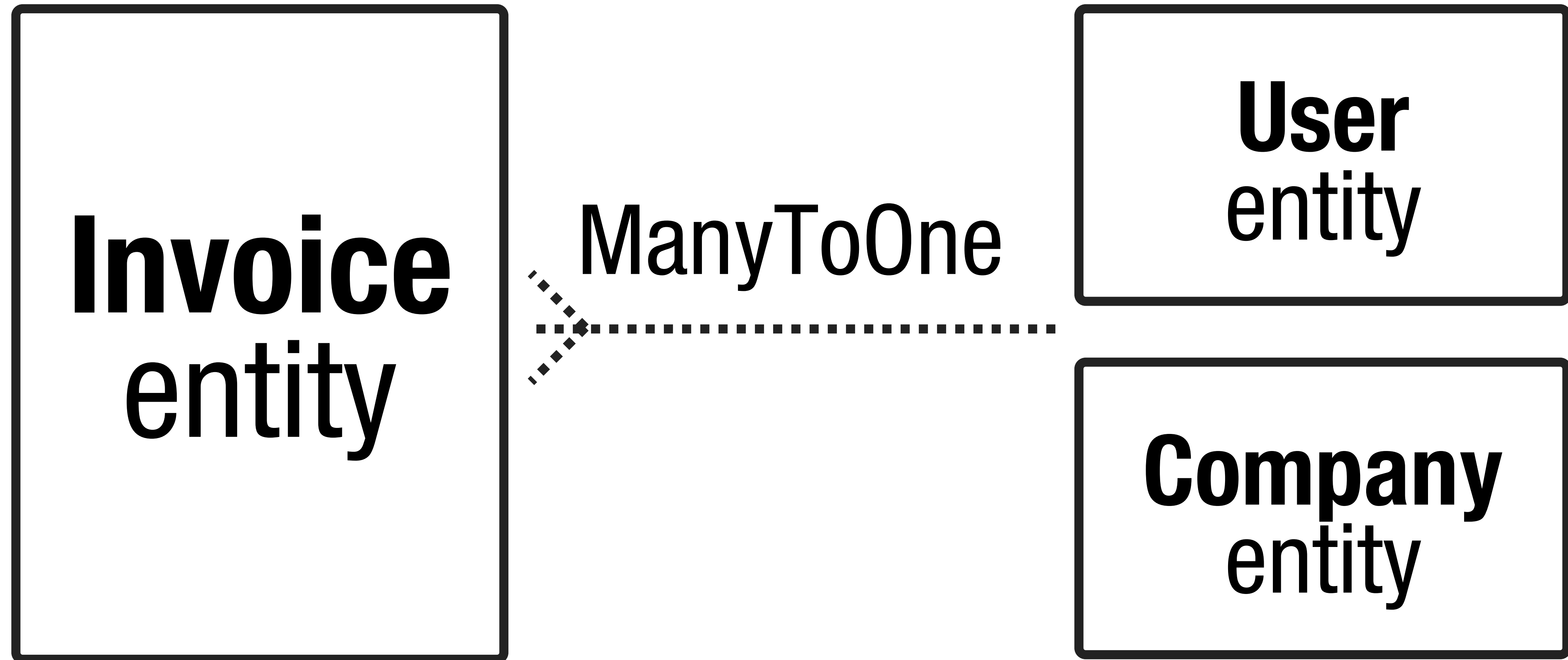
More information

http://symfony.com/doc/current/cookbook/doctrine/resolve_target_entity.html

Most entity relations are well defined



What if the target entity can change?



Our needs

- Create a generic **InvoiceBundle** able to work both with **User** and **Company** entities.
- No code change or special configuration is needed for the bundle.

1. Define an abstract "subject" interface

```
namespace Acme\InvoiceBundle\Model;
```

```
interface InvoiceSubjectInterface
{
    // Add here your custom methods
    // ...
}
```

2. Make entities implement this interface

```
use Doctrine\ORM\Mapping as ORM;  
use Acme\InvoiceBundle\Model\InvoiceSubjectInterface;
```

```
/** @ORM\Entity */  
class User implements InvoiceSubjectInterface  
{  
    // ...  
}
```

```
/** @ORM\Entity */  
class Company implements InvoiceSubjectInterface  
{  
    // ...  
}
```

different ↑
applications ↓

3. Configure the entity association

```
namespace Acme\InvoiceBundle\Entity;

use Doctrine\ORM\Mapping AS ORM;
use Acme\InvoiceBundle\Model\InvoiceSubjectInterface;

/** @ORM\Entity */
class Invoice
{
    /**
     * @ORM\ManyToOne(
     *     targetEntity="Acme\InvoiceBundle\Model\InvoiceSubjectInterface")
     */
    protected $subject;
}
```

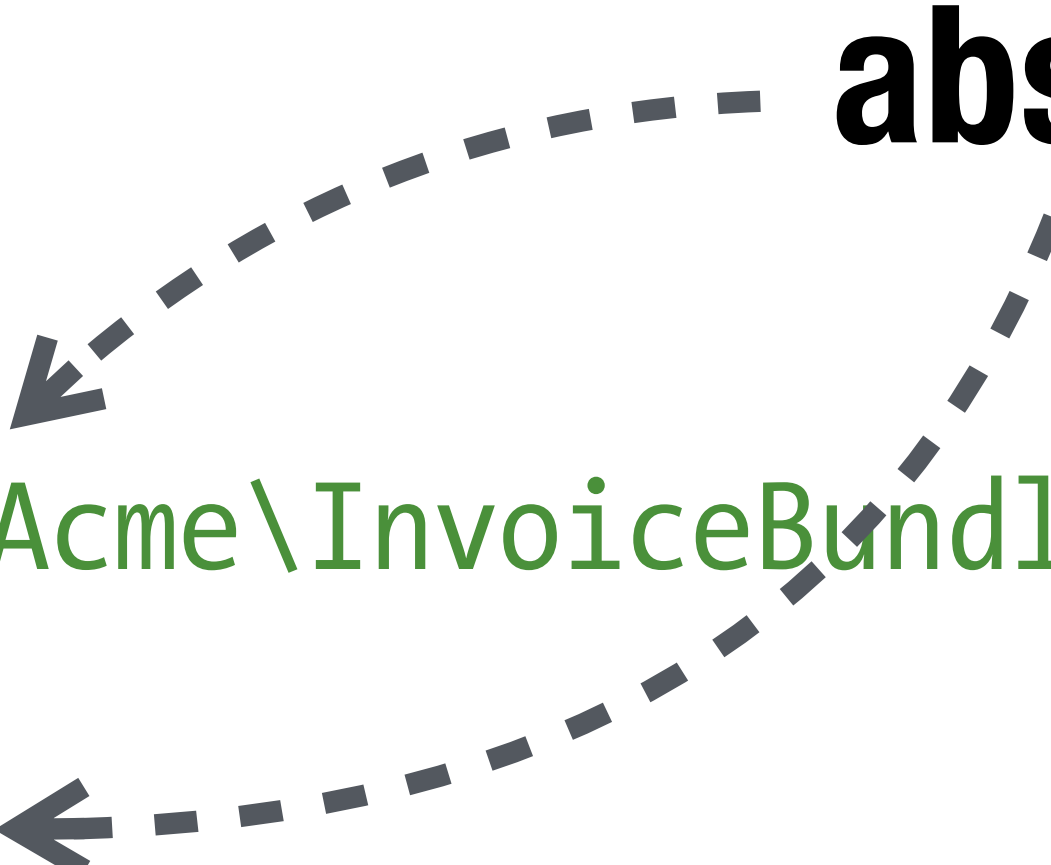
3. Configure the entity association

```
namespace Acme\InvoiceBundle\Entity;

use Doctrine\ORM\Mapping AS ORM;
use Acme\InvoiceBundle\Model\InvoiceSubjectInterface;

/** @ORM\Entity */
class Invoice
{
    /**
     * @ORM\ManyToOne(
     *     targetEntity="Acme\InvoiceBundle\Model\InvoiceSubjectInterface")
     */
    protected $subject;
}
```

abstract target entity



4. Define the target entity (at each application)

```
# app/config/config.yml
```

```
doctrine:
```

```
  # ...
```

```
  orm:
```

```
    # ...
```

```
    resolve_target_entities:
```

```
      Acme\InvoiceBundle\Model\InvoiceSubjectInterface:
```

```
      Acme\AppBundle\Entity\Customer
```

this is where magic
happens: **dynamic entity
resolution** at runtime



Improved WHERE ... IN

I learned this from



Marco
Pivetta

More information

http://doctrine-orm.readthedocs.org/projects/doctrine-orm/en/latest/changelog/migration_2_5.html#query-api-where-in-query-using-a-collection-as-parameter

Common code when using collections

```
$categories = ...
```

```
$categoryIds = array();  
foreach ($categories as $category) {  
    $categoryIds[] = $category->getId();  
}
```

```
$queryBuilder = $this  
    ->where('model.category IN (:category_ids)')  
    ->setParameter('category_ids', $categoryIds)  
;
```

Common code when using collections

```
$categories = ...
```

```
$categoryIds = array();  
foreach ($categories as $category) {  
    $categoryIds[] = $category->getId();  
}
```

transform the
ArrayCollection
into an array of IDs



```
$queryBuilder = $this  
    ->where('model.category IN (:category_ids)')  
    ->setParameter('category_ids', $categoryIds)  
;
```

In Doctrine 2.5 this is no longer needed

```
$categories = ...
```

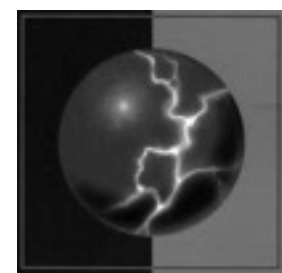
```
$categoryIds = array();  
foreach ($categories as $category) {  
    $categoryIds[] = $category->getId();  
}
```

```
$queryBuilder = $this  
    ->where('model.category IN (:categories)')  
    ->setParameter('categories', $categories)  
;
```

WHERE..IN
supports the use of
ArrayCollection

Default table options

I learned this from



Darien
Hager

More information

<https://github.com/doctrine/DoctrineBundle/pull/420>

Define the default table options

```
# app/config/config.yml
doctrine:
    dbal:
        default_table_options:
            charset: utf8mb4
            collate: utf8mb4_unicode_ci
            engine:  InnoDB
```

NEW

DoctrineBundle 1.6

Define the default table options

```
# app/config/config.yml
```

```
doctrine:
```

```
  dbal:
```

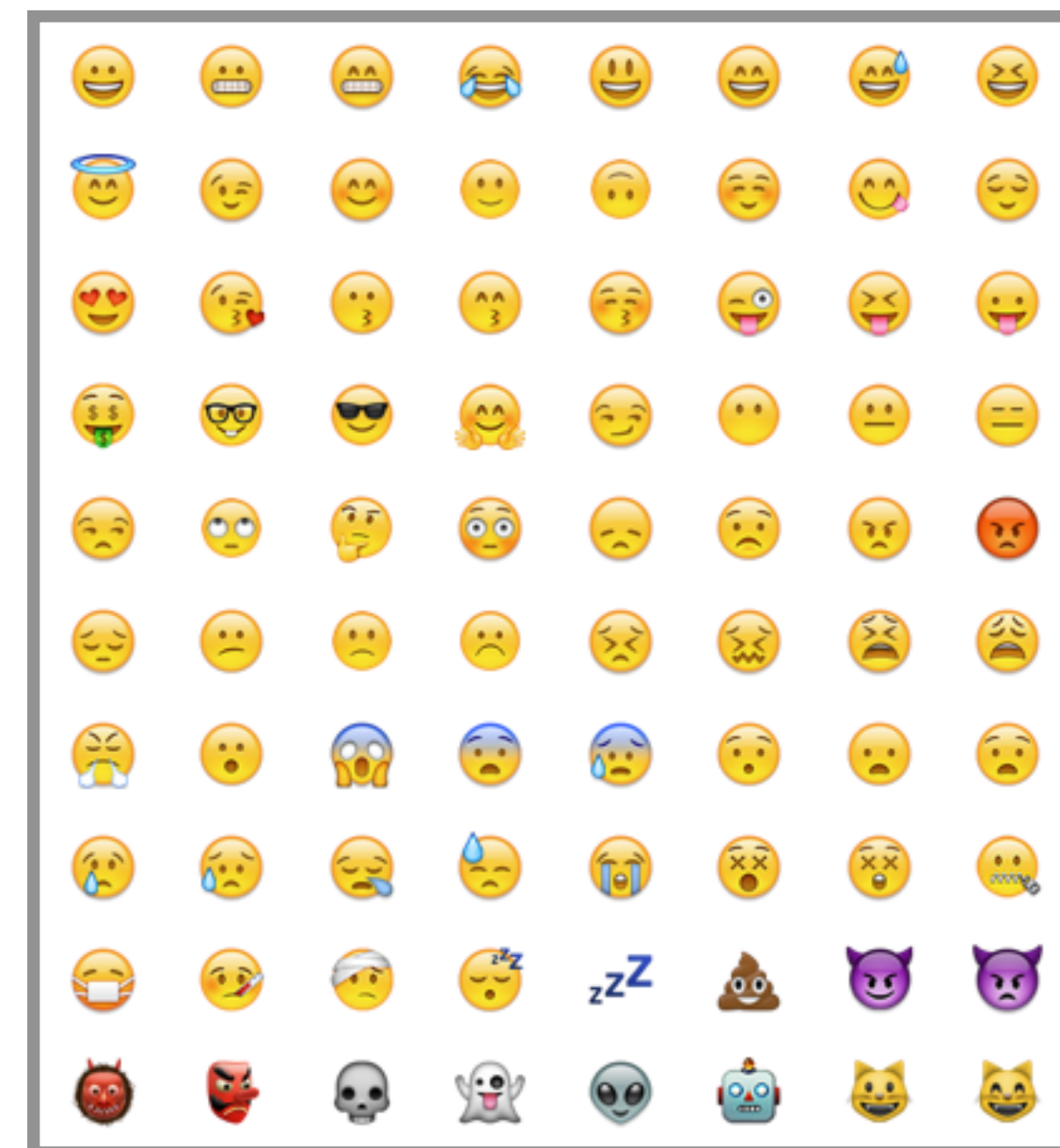
```
    default_table_options:
```

```
      charset: utf8mb4
```

```
      collate: utf8mb4_unicode_ci
```

```
      engine: InnoDB
```

required to
support emojis



NEW

DoctrineBundle 1.6

Savepoints

I learned this from



Christopher
Davis

More information

<https://github.com/doctrine/DoctrineBundle/pull/451>

<http://doctrine.readthedocs.org/en/latest/en/manual/transactions.html>

Normal Doctrine transaction

```
try {  
    $conn->beginTransaction();  
  
    // do something ...  
  
    $conn->commit();  
} catch(Exception $e) {  
    $conn->rollback();  
}
```

Doctrine transaction with save points

```
try {  
    $conn->beginTransaction('create_user');  
  
    // do something ...  
  
    $conn->commit('create_user');  
} catch(Exception $e) {  
    $conn->rollback('create_user');  
}
```

Give a name to a transaction to create a "save point"



Nesting Doctrine transactions

```
try {  
    $conn->beginTransaction();  
  
    // do something ...  
  
    $conn->beginTransaction('create_user');  
    try {  
        // do something ...  
  
        $conn->commit('create_user');  
    } catch(Exception $e) { $conn->rollback('create_user'); }  
  
    $conn->commit();  
} catch(Exception $e) { $conn->rollback(); }
```

Nested transaction
with a save point



Adding "save points" automatically

```
# app/config/config.yml
doctrine:
    dbal:
        use_savepoints: true
```

NEW DoctrineBundle 1.6

Adding "save points" automatically

```
# app/config/config.yml
doctrine:
    dbal:
        use_savepoints: true
```

NEW DoctrineBundle 1.6

Doctrine\DBAL\Connection

```
protected function _getNestedTransactionSavePointName()
{
    return 'DOCTRINE2_SAVEPOINT_'. $this->_transactionNestingLevel;
}
```

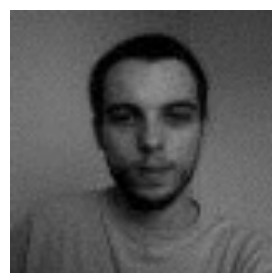
mini-tip

Optional things

I learned this from



Hugo
Hamon



Christophe
Coevoet




Grégoire
Pineau

The "action" suffix is optional for annotations

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

no need to call this
method **indexAction()**



The "action" suffix is optional for annotations

```
class DefaultController extends Controller  
{
```

```
  /**
```

```
   * @Route("/", name="homepage")
```

```
  */
```

```
  public function index(Request $request)
```

```
{
```

```
    // ...
```

```
}
```

```
}
```

no need to call this
method **indexAction()**



this only works when using
routing annotations



The "action" suffix is optional for annotations

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

no need to call this
method **indexAction()**

this only works when using
routing annotations

best used when the
controller class only
contains action methods

The "method" param is optional for listeners

```
# app/config/services.yml
```

```
services:
```

```
    app.exception_listener:
```

```
        class: AppBundle\EventListener\ExceptionListener
```

```
        tags:
```

```
            - {
```

```
                name: kernel.event_listener,
```

```
                event: kernel.exception,
```

```
                method: 'onKernelException'
```

```
            }
```

no need to define the
method parameter



The "method" param is optional for listeners

```
# app/config/services.yml
```

```
services:
```

```
    app.exception_listener:
```

```
        class: AppBundle\EventListener\ExceptionListener
```

```
        tags:
```

```
            - {
```

```
                name: kernel.event_listener,
```

```
                event: kernel.exception,
```

```
                method: 'onKernelException'
```

```
            }
```

no need to define the
method parameter



on + camelCased event name

The `is_granted()` check in error pages

```
{# app/Resources/TwigBundle/views/Exception/error.html.twig #}  
{% if app.user and is_granted('ROLE_ADMIN') %}
```

...

```
{% endif %}
```

Symfony 2.7 and previous need
this check to avoid issues in
pages not covered by a firewall.



The `is_granted()` check in error pages

```
{# app/Resources/TwigBundle/views/Exception/error.html.twig #}  
{% if app.user and is_granted('ROLE_ADMIN') %}
```

...

```
{% endif %}
```

Symfony 2.7 and previous need
this check to avoid issues in
pages not covered by a firewall.




```
{% if is_granted('ROLE_ADMIN') %}
```

...

```
{% endif %}
```

Symfony 2.8 no longer
requires this check



Composer

Improve class loading performance

I learned this from



More information

<https://github.com/symfony/symfony/pull/16655>

Don't load test classes in production

```
{  
    "autoload": {  
        "psr-4": { "MyLibrary\\": "src/" }  
    },  
    "autoload-dev": {  
        "psr-4": { "MyLibrary\\Tests\\": "tests/" }  
    }  
}
```

This is what we
said during
SymfonyCon 2014



New "exclude-from-classmap" option

```
// composer.json
{
    "autoload": {
        "exclude-from-classmap": ["/Tests/", "/test/", "/tests/"]
    }
}
```

These classes are excluded
from the optimized autoloader



New "exclude-from-classmap" option

```
// composer.json
{
    "autoload": {
        "exclude-from-classmap": ["/Tests/", "/test/", "/tests/"]
    }
}
```



These classes are excluded
from the optimized autoloader

```
$ composer dump-autoload --optimize
```


New "exclude-from-classmap" option

```
// composer.json
{
  "autoload": {
    "exclude-from-classmap": ["**/Tests/", "/test/*"]
  }
}
```

****** matches anything




***** matches anything except **/**


github.com/symfony/symfony/pull/16397





This repository Search


Pull requestsIssuesGist




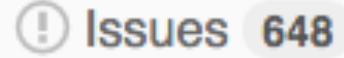
 symfony / symfony

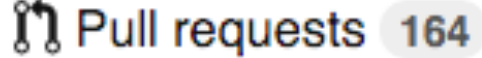
 Unwatch 1,017


 Unstar 11,208


 Fork 4,408

 Code

 Issues 648

 Pull requests 164

 Pulse

 Graphs

added the new Composer exclude-from-classmap option #16397

 Merged fabpot merged 1 commit into symfony:2.3 from fabpot:composer-exclude-from-classmap on 30 Oct

 Conversation 8

 Commits 1

 Files changed 38

+152 -38 

 Showing 38 changed files with 152 additions and 38 deletions.

UnifiedSplit

5  composer.json  

		@@ -75,7 +75,10 @@
75	75	"src/Symfony/Component/HttpFoundation/Resources/stubs",
76	76	"src/Symfony/Component/Intl/Resources/stubs"
77	77],
78	-	"files": ["src/Symfony/Component/Intl/Resources/stubs/functions.php"]
	78	+ "files": ["src/Symfony/Component/Intl/Resources/stubs/functions.php"],
	79	+ "exclude-from-classmap": [
	80	+ "**/Tests/"
	81	+]
79	82	},
80	83	"minimum-stability": "dev".

Twig

Deprecated filters

I learned this from



Fabien
Potencier

More information

<http://twig.sensiolabs.org/doc/advanced.html#deprecated-filters>

Applications evolve in time...

```
class AppExtension extends \Twig_Extension
{
    public function getFilters()
    {
        return array(
            new \Twig_SimpleFilter('old_filter', ...),
            new \Twig_SimpleFilter('new_filter', ...),
        );
    }
}
```

How can you deprecate
"old_filter" without
breaking the application?

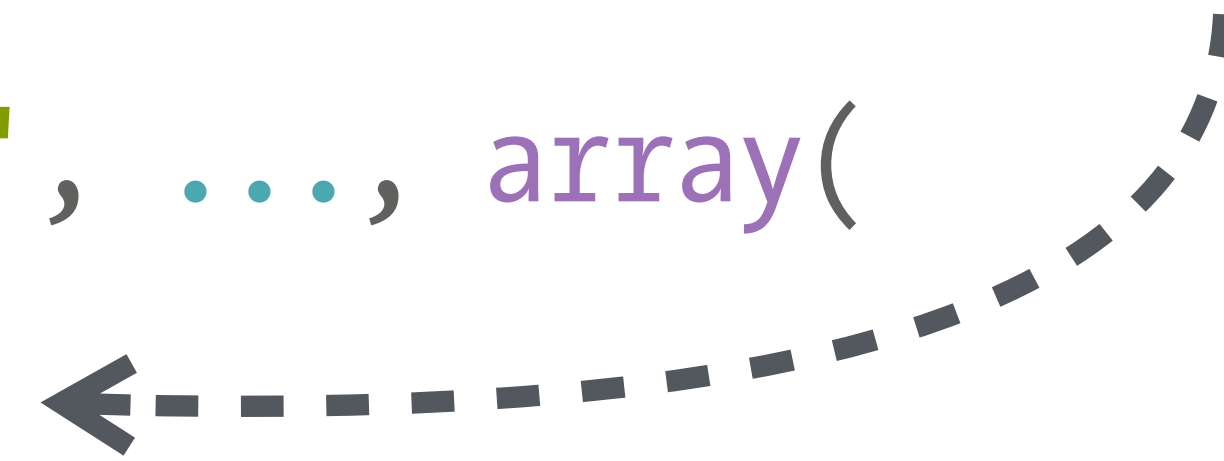


Filters can be deprecated

NEW Twig 1.23

```
class AppExtension extends \Twig_Extension
{
    public function getFilters()
    {
        return array(
            new \Twig_SimpleFilter('old_filter', ..., array(
                'deprecated' => true,
                'alternative' => 'new_filter',
            )),
            new \Twig_SimpleFilter('new_filter', ...),
        );
    }
}
```

This is how you deprecate filters and alert developers without breaking things



Check if some block exists

I learned this from



Martin
Hason

More information

<https://github.com/twigphp/Twig/pull/1831>

Avoid missing blocks

```
{% if 'title' is block %}  
    <title>{{ block('title') }}<title>  
{% endif %}
```

NEW Twig 1.2X

not merged yet

Variadic filters

I learned this from



Martin
Hason

More information

<https://github.com/twigphp/Twig/pull/1699>

The problem with filter arguments

```
{{ product.photo | image(400, 150, 0.9) }}
```

The problem with filter arguments

```
{{ product.photo | image(400, 150, 0.9) }}
```

What if I need to define more arguments?

The problem with filter arguments

```
{{ product.photo|image(400, 150, 0.9) }}
```

What if I need to define more arguments?

```
{{ product.photo|image(  
    width = 400, height = 150, opacity = 0.9  
) }}
```

↑ this is a valid solution for Twig, but the underlying PHP code is still very complex

Defining a filter with lots of arguments


```
<?php
$filter = new Twig_SimpleFilter('image', function (
    $path, $width, $height, $opacity
) {
    $path = ...
    $width = ...
    $height = ...
    $opacity = ...
});
```

Defining a variadic filter

```
$filter = new Twig_SimpleFilter('image', function (  
    $path, $options = array()  
) {  
  
    // ...  
  
}, array('is_variadic' => true));
```

Defining a variadic filter

```
$filter = new Twig_SimpleFilter('image', function (  
    $path, $options = array()  
) {  
    // ...  
}, array('is_variadic' => true));
```

 a single **variadic** parameter holds any number of passed parameters (unlimited)

Getting the values passed

```
$filter = new Twig_SimpleFilter('image', function (  
    $path, $options = array()  
) {  
    $path = ...  
    $width = $options['width'];  
    $height = $options['height'];  
    $opacity = $options['opacity'];  
  
}, array('is_variadic' => true));
```

Template source code

I learned this from



Nicolas
Grekas

More information

<https://github.com/twigphp/Twig/pull/1813>
<https://github.com/twigphp/Twig/pull/1807>

The Template class added a new method

```
abstract class Twig_Template implements Twig_TemplateInterface
{
    // ...

    public function getSource()
    {
        // ...
    }
}
```

NEW Twig 1.22

How to display the source code of the template

```
{% set template = _self %}  
{{ template.source|e }}
```

How to display the source code of the template

```
{% set template = _self %}  
{{ template.source|e }}
```



having the source code allows for
example to test if the template is
backwards/forwards compatible

The (dirty) secret of the getSource() method

```
/* default/index.html.twig */
class __TwigTemplate_c917e55e0a4ad9b0ed28003c15c48de756d875a69e121aca97d5a53e84eaef4f extends Twig_Template
{
    public function __construct(Twig_Environment $env) { }
    protected function doDisplay(array $context, array $blocks = array()) { // ... }

    public function getTemplateName()
    {
        return "default/index.html.twig";
    }

    // ...

    public function getDebugInfo()
    {
        return array ( 115 => 54, 109 => 53, 93 => 43, 68 => 22, 66 => 21, 57 => 15, 46 => 7, 41 => 4, 35 => 3, 11 => 1,);
    }
}
```

```
/* {% extends 'base.html.twig' %}*/
/* */
/* {% block body %}*/
/*     <div id="wrapper">*/
/*         <div id="container">*/
/*             <div id="welcome">*/
/*                 <h1><span>Welcome to</span> Symfony {{ constant('Symfony\\Component\\HttpKernel\\Kernel::VERSION') }}</h1>*/
/*             </div>*/
/*         ...
/*     */
```

Compiled Twig templates include
its full source code at the bottom

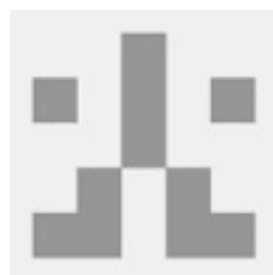
mini-tip

Email delivery whitelist

I learned this from



Terje
Bråten



E.
Weimann

More information

<https://github.com/symfony/symfony-docs/pull/4924>

<https://github.com/symfony/symfony-docs/pull/4925>

In dev environment, redirect all emails

```
# app/config/config_dev.yml
```

```
swiftmailer:
```

```
    delivery_address: dev@example.com
```

Whitelist some emails in dev environment

```
# app/config/config_dev.yml
```

```
swiftmailer:
```

```
    delivery_address: dev@example.com
```

```
    delivery_whitelist:
```

- "/notifications@example.com\$/"
- "/^admin@*\$/".

Whitelist some emails in dev environment

```
# app/config/config_dev.yml
```

```
swiftmailer:
```

```
    delivery_address: dev@example.com
```

```
    delivery_whitelist:
```

- "/notifications@example.com\$/"
- "/^admin@*\$/"



email addresses that match
the regular expression will be
sent (even in **dev** environment)

Configuration

Environment variables

I learned this from



Fabien
Potencier

More information

<http://github.com/symfony/symfony/pull/16403>

Setting ENV variables via the web server

```
<VirtualHost *:80>
```

```
    ServerName      Symfony
```

```
    DocumentRoot    "/path/to/symfony_2_app/web"
```

```
    DirectoryIndex  index.php index.html
```

```
    SetEnv           SYMFONY__DATABASE_USER      user
```

```
    SetEnv           SYMFONY__DATABASE_PASSWORD secret
```

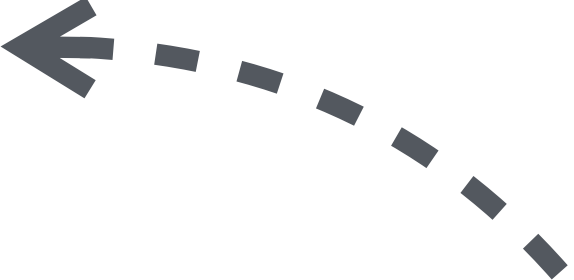
```
    <Directory "/path/to/symfony_2_app/web">
```

```
        AllowOverride All
```

```
        Allow from All
```

```
    </Directory>
```

```
</VirtualHost>
```



same as setting
database_user and
database_password
in **parameters.yml**

ENV variables are dumped into the container

```
// app/cache/dev/appDevDebugProjectContainer.xml
```

```
<container>
  <parameters>
    <parameter key="database_user">root</parameter>
    <parameter key="database_password">__secret__ </parameter>
    <!-- ... -->
  </parameters>
</container>
```

DEV
environment

```
// app/cache/prod/appProdProjectContainer.php
```

```
protected function getDefaultParameters()
{
    return array(
        // ...
        'database_user' => 'root',
        'database_password' => __secret__,
    );
}
```

PROD
environment

ENV variables are dumped into the container

```
// app/cache/dev/appDevDebugProjectContainer.xml
```

```
<container>
```

```
  <parameters>
```

```
    <parameter key="database_user">root</parameter>
```

```
    <parameter key="database_password">__secret__</parameter>
```

```
    <!-- ... -->
```

```
  </parameters>
```

```
</container>
```

DEV
environment

You can see the password.



```
// app/cache/prod/appProdProjectContainer.php
```

```
protected function getDefaultParameters()
```

```
{
```

```
    return array(
```

```
        // ...
```

```
        'database_user' => 'root',
```

```
        'database_password' => __secret__,
```

```
    );
```

```
}
```

PROD
environment

ENV variables are dumped into the container

```
// app/cache/dev/appDevDebugProjectContainer.xml
```

```
<container>
```

```
  <parameters>
```

```
    <parameter key="database_user">root</parameter>
```

```
    <parameter key="database_password">__secret__</parameter>
```

```
    <!-- ... -->
```

```
  </parameters>
```

```
</container>
```

DEV
environment

You can see the password.



```
// app/cache/prod/appProdProjectContainer.php
```

```
protected function getDefaultParameters()
```

```
{
```

```
  return array(
```

```
    // ...
```

```
    'database_user' => 'root',
```

```
    'database_password' => __secret__,
```

```
  );
```

```
}
```

PROD
environment

It's static (if ENV variable
changes, app breaks)



This is what you want...


```
// app/cache/prod/appProdProjectContainer.php
protected function getDefaultParameters()
{
    return array(
        // ...
        'database_user' => 'root',
        'database_password' => getenv("SYMFONY__DATABASE_PASSWORD");
    );
}
```

This is what you want...

```
// app/cache/prod/appProdProjectContainer.php
protected function getDefaultParameters()
{
    return array(
        // ...
        'database_user' => 'root',
        'database_password' => getenv("SYMFONY__DATABASE_PASSWORD");
    );
}
```




... but we failed at
implementing this feature.


github.com/symfony/symfony/pull/16403



This repository Search

[Pull requests](#) [Issues](#) [Gist](#)


  

 [symfony](#) / [symfony](#)

[Unwatch](#) 1,020 [Unstar](#) 11,226 [Fork](#) 4,409

[Code](#) [Issues](#) 654 [Pull requests](#) 168 [Pulse](#) [Graphs](#)

[DependencyInjection] added a way to reference dynamic environment variables in service definitions #16403

fabpot comm

Q

Bug fix?

New featu

BC breaks?

Deprecations?

Tests pass?

Fixed tickets

License


no

no

yes

#10138, #7555

MIT

fabpot commented 19 days ago

Owner

Conclusion: This is not possible, and by far. The semantic configuration used by bundle extensions is resolved before being passed to the extension. If we do not resolve it anymore, each extension would have to resolve values on a case-by-case basis AND would need to keep the non-resolved value. That might be a problem when a parameter has influences over the configuration.

So, a generic system is just **not possible without rethinking the whole system.**

Assignee


No one assigned

Notifications




Unsubscribe


You're receiving notifications because you're subscribed to this

github.com/Incenteev/DynamicParametersBundle

 This repository

[Pull requests](#) [Issues](#) [Gist](#)


  

 [Incenteev](#) / [DynamicParametersBundle](#)




[Watch](#) 4 [Star](#) 19 [Fork](#) 1

[Code](#) [Issues](#) 1 [Pull requests](#) 1 [Pulse](#) [Graphs](#)

Branch: [master](#) [DynamicParametersBundle / README.md](#) [Find file](#) [Copy path](#)

 **stof** Add a service to access dynamic parameters at runtime 1ac957b on 23 Dec 2014

1 contributor

106 lines (76 sloc) | 4.12 KB [Raw](#) [Blame](#) [History](#)   

DynamicParametersBundle

This bundle provides a way to read parameters from environment variables at runtime. The value defined in the container parameter is used as fallback when the environment variable is not available.

build passing

Scrutinizer 8.79

coverage 63 %

SLInsight ★★

stable No Release

unstable 1.0.x-dev

license MIT

Installation

Installation is a quick (I promise!) 2 step process:

1. Download IncenteevDynamicParametersBundle
2. Enable the bundle

**Alternative to
environment
variables**

1. Create a "credentials" file in the prod server

```
# /etc/credentials/example.com.yml
```

```
parameters:
```

```
  database_user:      ...
```

```
  database_password:  ...
```

```
  mailer_user:        ...
```

```
  mailer_password:    ...
```

```
  secret:             ...
```

```
  github_client_id:   ...
```

```
  github_client_secret: ...
```

2. Load the credentials file in the application

```
# app/config/config.yml
```

```
imports:
```

- { resource: parameters.yml }
- { resource: services.yml }
- { resource: security.yml }
- { resource: admin.yml }
- { resource: '/etc/credentials/example.com.yml',
 ignore_errors: true }

```
# ...
```

3. In "dev" machine

- Credentials file doesn't exist but no error is triggered (because of ignore_errors).
- App uses the regular parameters.yml file.

4. In "prod" machine

- Credentials file overrides parameters.yml
- The right parameters are available anywhere (e.g. console commands)
- Developers can't see the production configuration options.
- Requires discipline to add/remove options.

Monolog

Custom channels

Creating a custom channel is painless

```
# app/config/config.yml
```

```
monolog:
```

```
    channels: ["marketing"]
```


Creating a custom channel is painless

```
# app/config/config.yml
```

```
monolog:
```

```
    channels: ["marketing"]
```

```
$this->get('monolog.logger.marketing')->info('.....');
```

Creating a custom channel is painless

```
# app/config/config.yml
```

```
monolog:
```

```
    channels: ["marketing"]
```

```
$this->get('monolog.logger.marketing')->info('.....');
```

```
# app/logs/dev.log
```

```
[2015-11-30 17:44:19] marketing.INFO: ..... [] []
```

Creating a custom channel is painless

```
# app/config/config.yml
monolog:
  channels: ["marketing"]
```

now you can process these messages apart from the rest of logs (e.g. save them in a different file)

```
$this->get('monolog.logger.marketing')->info('.....');
```

```
# app/logs/dev.log
[2015-11-30 17:44:19] marketing.INFO: ..... [] []
```



The channel name can be a parameter

```
# app/config/config.yml
```

```
parameters:
```

```
    app_channel: 'marketing'
```

```
monolog:
```

```
    channels: [%app_channel%]
```

```
services:
```

```
    app_logger:
```

```
        class: ...
```

```
        tags: [{ name: monolog.logger, channel: %app_channel% }]
```

NEW

MonologBundle 2.8

Custom formatters

I learned this from



Grégoire
Pineau

More information

[http://symfony.com/doc/current/cookbook/logging/monolog.html#changing—the-formatter](http://symfony.com/doc/current/cookbook/logging/monolog.html#changing-the-formatter)

Custom monolog processor

```
namespace AppBundle\Logger\OrderProcessor;

use AppBundle\Entity\Order;

class OrderProcessor
{
    public function __invoke(array $record)
    {
        if (isset($record['context']['order']) && $record['context']['order'] instanceof Order) {
            $order = $record['context']['order'];
            $record['context']['order'] = [
                'number' => $order->getNumber(),
                'user'    => $order->get...,
                'total'   => $order->get...,
                'status'  => $order->get...,
            ];
        }

        return $record;
    }
}
```

```
# app/config/config_prod.yml
services:
    monolog_processor:
        class: AppBundle\Logger\OrderProcessor
        tags:
            - { name: monolog.processor }
```

This is what we
said during
SymfonyCon 2014



Creating simple custom formatters is painless

```
# app/config/config.yml
```

```
services:
```

```
    app.log.formatter:
```

```
        class: 'Monolog\Formatter\LineFormatter'
```

```
        public: false
```

```
        arguments:
```

- "%%level_name%% [%%datetime%%] [%%channel%%] %%message%%\n%%context%%\n\n"
- 'H:i:s'
- false

Creating simple custom formatters is painless

```
# app/config/config.yml
```

```
services:
```

```
    app.log.formatter:
```

```
        class: 'Monolog\Formatter\LineFormatter'
```

```
        public: false
```

```
        arguments:
```

- "%%level_name%% [%%datetime%%] [%%channel%%] %%message%%\n%%context%%\n\n"
- 'H:i:s'
- false

the format of
each log line



Creating simple custom formatters is painless

```
# app/config/config.yml
```

```
services:
```

```
  app.log.formatter:
```

```
    class: 'Monolog\Formatter\LineFormatter'
```

```
    public: false
```

```
    arguments:
```

- "%%level_name%% [%%datetime%%] [%%channel%%] %%message%%\n %%context%%\n\n"
- 'H:i:s'
- false

the format of
each log line



the format of
%datetime% placeholder



Creating simple custom formatters is painless

```
# app/config/config.yml
```

```
services:
```

```
  app.log.formatter:
```

```
    class: 'Monolog\Formatter\LineFormatter'
```

```
    public: false
```

```
    arguments:
```

- "%%level_name%% [%%datetime%%] [%%channel%%] %%message%%\n%%context%%\n\n"
- 'H:i:s'
- false

the format of
each log line



the format of
%datetime% placeholder



Creating simple custom formatters is painless

```
# app/config/config.yml
```

```
services:
```

```
  app.log.formatter:
```

```
    class: 'Monolog\Formatter\LineFormatter'
```

```
    public: false
```

```
    arguments:
```

```
      - "%%level_name%% [%%datetime%%] [%%channel%%] %%message%%\n\n      %%context%%\n\n"
```

```
      - 'H:i:s'
```

```
      - false
```

the format of
each log line



the format of
%datetime% placeholder



ignore \n inside the
log messages?



Enable the custom log formatter

```
# app/config/config_dev.yml
```

```
monolog:
```

```
  handlers:
```

```
    main:
```

```
      type:      stream
```

```
      path:      "%kernel.logs_dir%/%kernel.environment%.log"
```

```
      level:     debug
```

```
      formatter: app.log.formatter
```

 - add the **formatter** option

Custom log format

```
INFO [18:38:21] [php] The Symfony\Component
\DependencyInjection\Reference::isStrict method is
deprecated since version 2.8 and will be removed in 3.0.
{"type":16384,"file":"myproject/vendor/symfony/symfony/
src/Symfony/Component/DependencyInjection/
Reference.php","line":73,"level":28928}
```

```
INFO [18:38:21] [request] Matched route "homepage".
{"route_parameters":{"_controller":"AppBundle\
\Controller\
\DefaultController::indexAction","_route":"homepage"},"r
equest_uri":"http://127.0.0.1:8000/"}
```

```
INFO [18:38:21] [security] Populated the TokenStorage
with an anonymous Token.
[]
```

```
DEBUG [18:38:21] [event] Notified event "kernel.request"
to listener "Symfony\Component\HttpKernel\EventListener
\DebugHandlersListener::configure".
[]
```

```
DEBUG [18:38:21] [event] Notified event "kernel.request"
to listener "Symfony\Component\HttpKernel\EventListener
\ProfilerListener::onKernelRequest".
```

Default log format

```
[2015-11-30 18:38:21] php.INFO: The Symfony\Component
\DependencyInjection\Reference::isStrict method is
deprecated since version 2.8 and will be removed in 3.0.
{"type":16384,"file":"myproject/vendor/symfony/symfony/
src/Symfony/Component/DependencyInjection/
Reference.php","line":73,"level":28928} []
```

```
[2015-11-30 18:38:21] request.INFO: Matched route
"homepage". {"route_parameters":
{"_controller":"AppBundle\Controller\
\DefaultController::indexAction","_route":"homepage"},"r
equest_uri":"http://127.0.0.1:8000/"}
```

```
[2015-11-30 18:38:21] security.INFO: Populated the
TokenStorage with an anonymous Token. [] []
```

```
[2015-11-30 18:38:21] event.DEBUG: Notified event
"kernel.request" to listener "Symfony\Component
\HttpKernel\EventListener
\DebugHandlersListener::configure". [] []
```

```
[2015-11-30 18:38:21] event.DEBUG: Notified event
"kernel.request" to listener "Symfony\Component
\HttpKernel\EventListener
\ProfilerListener::onKernelRequest". [] []
```

```
[2015-11-30 18:38:21] event.DEBUG: Notified event
"kernel.request" to listener "Symfony\Component
\HttpKernel\EventListener\DumpListener::configure". []
[]
```

Monolog "plugins"

I learned this from




Jordi
Boggiano

More information




<http://github.com/Seldaek/monolog/wiki/Third-Party-Packages>

github.com/Seldaek/monolog/wiki/Third-Party-Packages





This repository Search


Pull requestsIssuesGist



Seldaek / monolog

 Watch 257

 Star 4,075

 Fork 744

<> Code

! Issues 25

🔗 Pull requests 8

📖 Wiki

📶 Pulse

📊 Graphs

Third Party Packages

EditNew Page

This page lists third party packages implementing useful Monolog handlers, formatters or processors.

Handlers

- [nackjicholson/monolog-gitter-im](#) sends log records to the [gitter.im](#) API.
- [logentries/logentries-monolog-handler](#) sends log records to a [logentries.com](#) account.
- [mead-steve/mono-snag](#) sends log records to a [bugsnag](#) account.
- [bartlett/monolog-callbackfilterhandler](#) handler wrapper that filters records based on a list of callback functions.
- [waza-ari/monolog-mysql](#) allows to store log messages in a MySQL Table
- [bartlett/monolog-growlhandler](#) handler that send notifications to Growl on Mac OS X and Windows
- [yegortokmakov/monolog-fluentd](#) handler that send notifications to Fluentd / td-agent
- [do3meli/monolog-cassandra](#) Cassandra database handler based on the [DataStax PHP-](#)


▼ Pages 2


[Home](#)

[Third Party Packages](#)

Clone this wiki locally

https://github.com/Seldaek/



 Clone in Desktop

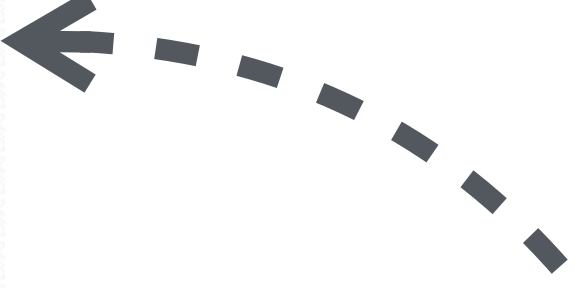
bramus/monolog-colored-line-formatter

- Shows log messages in color according to their level.
- It uses ANSI Escape Sequences.

```
[2015-01-05 00:00:04] DEMO.DEBUG: Lorem ipsum dolor sit amet, consectetur adipiscing elit. [] []  
[2015-01-05 00:00:04] DEMO.INFO: Lorem ipsum dolor sit amet, consectetur adipiscing elit. [] []  
[2015-01-05 00:00:04] DEMO.NOTICE: Lorem ipsum dolor sit amet, consectetur adipiscing elit. [] []  
[2015-01-05 00:00:04] DEMO.WARNING: Lorem ipsum dolor sit amet, consectetur adipiscing elit. [] []  
[2015-01-05 00:00:04] DEMO.ERROR: Lorem ipsum dolor sit amet, consectetur adipiscing elit. [] []  
[2015-01-05 00:00:04] DEMO.CRITICAL: Lorem ipsum dolor sit amet, consectetur adipiscing elit. [] []  
[2015-01-05 00:00:04] DEMO.ALERT: Lorem ipsum dolor sit amet, consectetur adipiscing elit. [] []  
[2015-01-05 00:00:04] DEMO.EMERGENCY: Lorem ipsum dolor sit amet, consectetur adipiscing elit. [] []
```

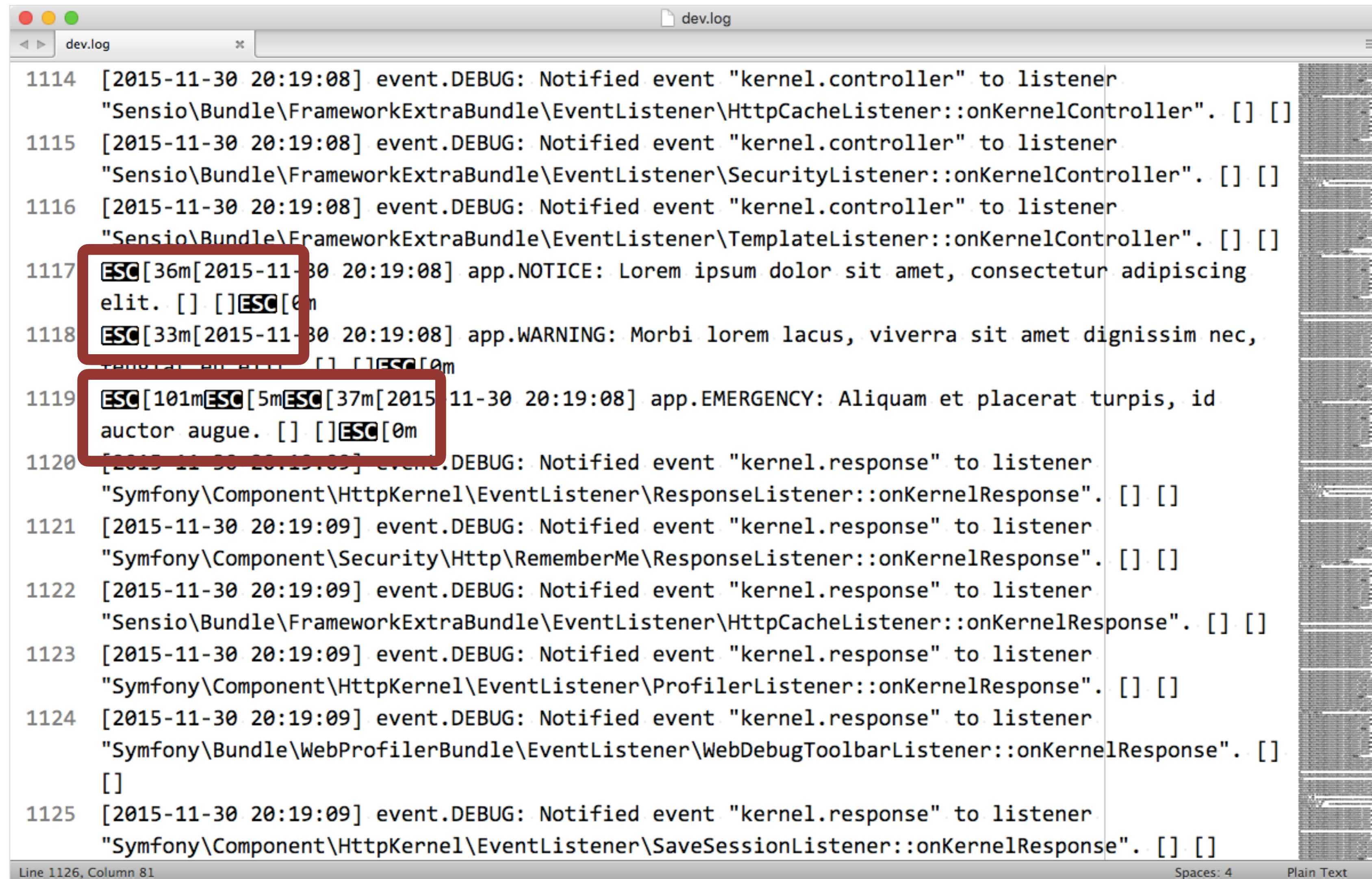

Colored logs in action

```
3. /Users/javier? (tail)
[2015-11-30 20:19:08] event.DEBUG: Notified event "kernel.controller" to listener "Sensio\Bundle\FrameworkExtraBundle\Event\Listener\ControllerListener::onKernelController". [] []
[2015-11-30 20:19:08] event.DEBUG: Notified event "kernel.controller" to listener "Sensio\Bundle\FrameworkExtraBundle\Event\Listener\ParamConverterListener::onKernelController". [] []
[2015-11-30 20:19:08] event.DEBUG: Notified event "kernel.controller" to listener "Sensio\Bundle\FrameworkExtraBundle\Event\Listener\HttpCacheListener::onKernelController". [] []
[2015-11-30 20:19:08] event.DEBUG: Notified event "kernel.controller" to listener "Sensio\Bundle\FrameworkExtraBundle\Event\Listener\SecurityListener::onKernelController". [] []
[2015-11-30 20:19:08] event.DEBUG: Notified event "kernel.controller" to listener "Sensio\Bundle\FrameworkExtraBundle\Event\Listener\TemplateListener::onKernelController". [] []
[2015-11-30 20:19:08] app.NOTICE: Lorem ipsum dolor sit amet, consectetur adipiscing elit. [] []
[2015-11-30 20:19:08] app.WARNING: Morbi lorem lacus, viverra sit amet dignissim nec, feugiat eu elit. [] []
[2015-11-30 20:19:09] event.DEBUG: Notified event "kernel.response" to listener "Symfony\Component\HttpKernel\Event\Listener\ResponseListener::onKernelResponse". [] []
[2015-11-30 20:19:09] event.DEBUG: Notified event "kernel.response" to listener "Symfony\Component\Security\Http\RememberMe\ResponseListener::onKernelResponse". [] []
[2015-11-30 20:19:09] event.DEBUG: Notified event "kernel.response" to listener "Sensio\Bundle\FrameworkExtraBundle\Event\Listener\HttpCacheListener::onKernelResponse". [] []
[2015-11-30 20:19:09] event.DEBUG: Notified event "kernel.response" to listener "Symfony\Component\HttpKernel\Event\Listener\ProfilerListener::onKernelResponse". [] []
[2015-11-30 20:19:09] event.DEBUG: Notified event "kernel.response" to listener "Symfony\Bundle\WebProfilerBundle\Event\Listener\WebDebugToolbarListener::onKernelResponse". [] []
[2015-11-30 20:19:09] event.DEBUG: Notified event "kernel.response" to listener "Symfony\Component\HttpKernel\Event\Listener\SaveSessionListener::onKernelResponse". [] []
```



if your editor or
console supports
ANSI escape
sequences

Colored logs in action



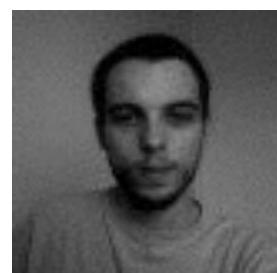
```
1114 [2015-11-30 20:19:08] event.DEBUG: Notified event "kernel.controller" to listener  
      "Sensio\Bundle\FrameworkExtraBundle\Event\Listener\HttpCacheListener::onKernelController". [] []  
1115 [2015-11-30 20:19:08] event.DEBUG: Notified event "kernel.controller" to listener  
      "Sensio\Bundle\FrameworkExtraBundle\Event\Listener\SecurityListener::onKernelController". [] []  
1116 [2015-11-30 20:19:08] event.DEBUG: Notified event "kernel.controller" to listener  
      "Sensio\Bundle\FrameworkExtraBundle\Event\Listener\TemplateListener::onKernelController". [] []  
1117 \ESC[36m[2015-11-30 20:19:08] app.NOTICE: Lorem ipsum dolor sit amet, consectetur adipiscing  
      elit. [] []\ESC[0m  
1118 \ESC[33m[2015-11-30 20:19:08] app.WARNING: Morbi lorem lacus, viverra sit amet dignissim nec,  
      [] []\ESC[0m  
1119 \ESC[101m\ESC[5m\ESC[37m[2015-11-30 20:19:08] app.EMERGENCY: Aliquam et placerat turpis, id  
      auctor augue. [] []\ESC[0m  
1120 [2015-11-30 20:19:09] event.DEBUG: Notified event "kernel.response" to listener  
      "Symfony\Component\HttpKernel\Event\Listener\ResponseListener::onKernelResponse". [] []  
1121 [2015-11-30 20:19:09] event.DEBUG: Notified event "kernel.response" to listener  
      "Symfony\Component\Security\Http\RememberMe\ResponseListener::onKernelResponse". [] []  
1122 [2015-11-30 20:19:09] event.DEBUG: Notified event "kernel.response" to listener  
      "Sensio\Bundle\FrameworkExtraBundle\Event\Listener\HttpCacheListener::onKernelResponse". [] []  
1123 [2015-11-30 20:19:09] event.DEBUG: Notified event "kernel.response" to listener  
      "Symfony\Component\HttpKernel\Event\Listener\ProfilerListener::onKernelResponse". [] []  
1124 [2015-11-30 20:19:09] event.DEBUG: Notified event "kernel.response" to listener  
      "Symfony\Bundle\WebProfilerBundle\Event\Listener\WebDebugToolbarListener::onKernelResponse". [] []  
1125 [2015-11-30 20:19:09] event.DEBUG: Notified event "kernel.response" to listener  
      "Symfony\Component\HttpKernel\Event\Listener\SaveSessionListener::onKernelResponse". [] []
```

if your editor or
console doesn't
support ANSI
escape sequences

mini-tip

Tweaking the web debug toolbar

I learned this from

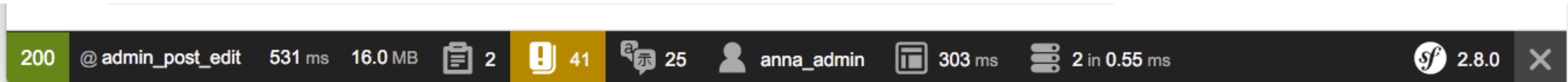
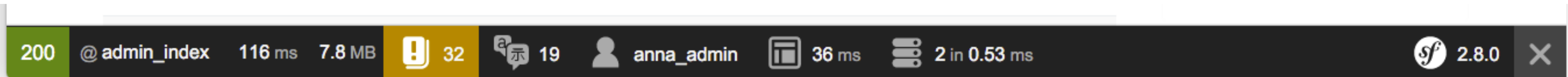
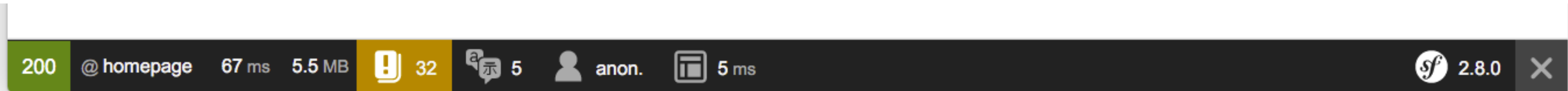


Christophe
Coevoet

More information

http://symfony.com/doc/current/cookbook/profiler/data_collector.html

The new web debug toolbar



the new toolbar is less cluttered because it hides panels that don't contain information

Reordering toolbar panels

```
# app/config/services.yml
```

```
services:
```

```
    data_collector.request:
```

```
        class: Symfony\Component\HttpKernel\DataCollector\RequestDataCollector
```

```
        tags:
```

- { name: "data_collector", template: "@WebProfiler/Collector/request.html.twig",
id: "request", priority: "-512" }

```
    data_collector.security:
```

```
        class: 'Symfony\Bundle\SecurityBundle\DataCollector\SecurityDataCollector'
```

```
        arguments: ['@security.token_storage', '@security.role_hierarchy', '@security.logout_url_generator']
```

```
        tags:
```

- { name: "data_collector", template: "@Security/Collector/security.html.twig",
id: "security", priority: "512" }

Reordering toolbar panels

```
# app/config/services.yml
```

```
services:
```

```
data_collector.request:
```

```
class: Symfony\Component\HttpKernel\DataCollector\RequestDataCollector
```

```
tags:
```

```
- { name: "data_collector", template: "@WebProfiler/Collector/request.html.twig",  
    id: "request", priority: "-512" }
```

```
data_collector.security:
```

```
class: 'Symfony\Bundle\SecurityBundle\DataCollector\SecurityDataCollector'
```

```
arguments: ['@security.token_storage', '@security.role_hierarchy', '@security.logout_url_generator']
```

```
tags:
```

```
- { name: "data_collector", template: "@Security/Collector/security.html.twig",  
    id: "security", priority: "512" }
```



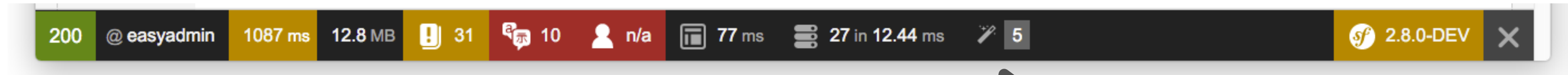
Built-in panels priorities

Panel	Priority
request	335
time	330
memory	325
ajax	315
form	310
exception	305
logger	300
events	290

Panel	Priority
router	285
translation	275
security	270
twig	257
swiftmailer	245
dump	240
config	-255

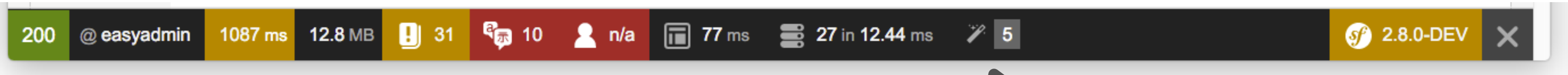
Adding your panels between the built-in panels

```
services:  
  app.data_collector:  
    # ...  
    tags:  
      - { name: "data_collector", template: "...", id: "..." } (NO EXPLICIT PRIORITY)
```

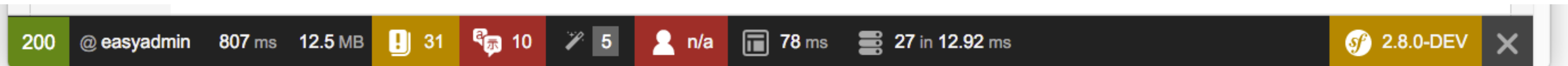


Adding your panels between the built-in panels

```
services:
  app.data_collector:
    # ...
    tags:
      - { name: "data_collector", template: "...", id: "..."} (NO EXPLICIT PRIORITY)
```



```
tags:
  - { name: "data_collector", template: "...", id: "...", priority: "273" }
```

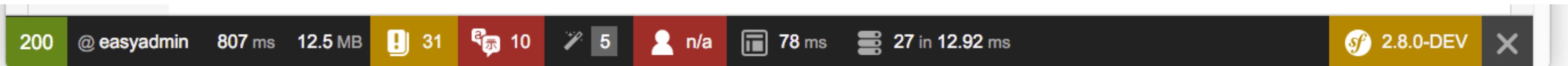


Adding your panels between the built-in panels

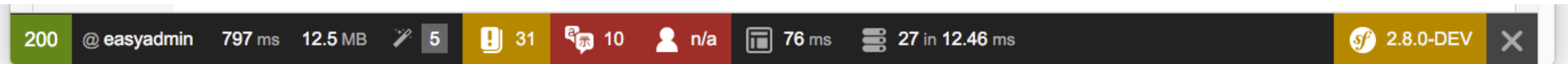
```
services:  
  app.data_collector:  
    # ...  
    tags:  
      - { name: "data_collector", template: "...", id: "..."} (NO EXPLICIT PRIORITY)
```



```
tags:  
  - { name: "data_collector", template: "...", id: "...", priority: "273" }
```



```
tags:  
  - { name: "data_collector", template: "...", id: "...", priority: "320" }
```



Remove panels

services:

data_collector.translation:

class: Symfony\Component\Translation\DataCollector\TranslationDataCollector

arguments: ['@translator.data_collector']

tags:

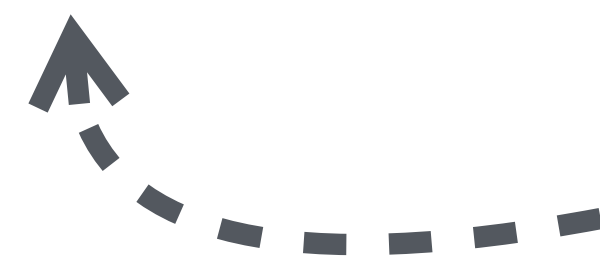
- { name: "data_collector" }

data_collector.request:

class: Symfony\Component\HttpKernel\DataCollector\RequestDataCollector

tags:

- { name: "data_collector" }



to remove a panel, define its service without any argument in its **data_collector** tag

Remove panels

services:

data_collector.translation:

class: Symfony\Component\Translation\DataCollector\TranslationDataCollector

arguments: ['@translator.data_collector']

tags:

- { name: "data_collector" }

 remove the **Request** panel

data_collector.request:

class: Symfony\Component\HttpKernel\DataCollector\RequestDataCollector

tags:

- { name: "data_collector" }

 remove the **Translation** panel

 to remove a panel, define its service without any argument in its **data_collector** tag

Security

Success and failure handlers

I learned this from



Fabien
Potencier

More information

<https://github.com/symfony/symfony-docs/issues/4258>

Success/failure handlers are
available since Symfony 2.0
(2011) ...

Success/failure handlers are
available since Symfony 2.0
(2011) ...

... but lots of developers
don't use them because they
are not documented.

The security.interactive_login event

```
services:
    login_listener:
        class: AppBundle\Listener\LoginListener
        arguments: ['@security.token_storage', '@doctrine']
        tags:
            - { name: 'kernel.event_listener', event: 'security.interactive_login' }
```



the most common event to "do things"
after the user successfully logs in

Defining a success handler

```
namespace AppBundle\Security;

use Symfony\Component\Security\Http\Authentication\AuthenticationSuccessHandlerInterface;

class LoginHandler implements AuthenticationSuccessHandlerInterface
{
    public function onAuthenticationSuccess(Request $request, TokenInterface $token)
    {
        // do something ...

        return $this->redirect('...');
        return new Response('...');
    }
}
```

Defining a success handler

you just need to implement
this interface...

```
namespace AppBundle\Security;
```

```
use Symfony\Component\Security\Http\Authentication\AuthenticationSuccessHandlerInterface;
```

```
class LoginHandler implements AuthenticationSuccessHandlerInterface
```

```
{
```

```
    public function onAuthenticationSuccess(Request $request, TokenInterface $token)
```

```
    {
```

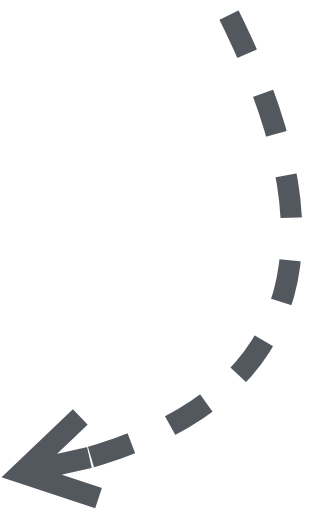
```
        // do something ...
```

```
        return $this->redirect('...');
```

```
        return new Response('...');
```

```
    }
```

```
}
```



Defining a success handler

you just need to implement this interface...

```
namespace AppBundle\Security;
```

```
use Symfony\Component\Security\Http\Authentication\AuthenticationSuccessHandlerInterface;
```

```
class LoginHandler implements AuthenticationSuccessHandlerInterface
```

```
{
```

```
    public function onAuthenticationSuccess(Request $request, TokenInterface $token)
```

```
    {
```

```
        // do something ...
```

```
        return $this->redirect('...');
```

```
        return new Response('...');
```

```
    }
```

```
}
```

...and return a Response instance

Enabling the success handler

```
# app/config/services.yml
```

```
services:
```

```
    app.login_handler:
```

```
        class: AppBundle\Security\LoginHandler
```

```
        arguments: ...
```

```
# app/config/security.yml
```

```
firewalls:
```

```
    main:
```

```
        pattern: ^/
```

```
        form_login:
```

```
            success_handler: app.login_handler
```

Enabling the success handler

```
# app/config/services.yml
```

```
services:
```

```
    app.login_handler:
```

```
        class: AppBundle\Security\LoginHandler
```

```
        arguments: ...
```

```
# app/config/security.yml
```

```
firewalls:
```

```
    main:
```

```
        pattern: ^/
```

```
        form_login:
```

```
            success_handler: app.login_handler
```

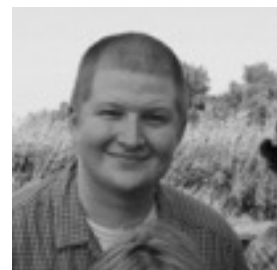
the LoginHandler class is
executed when the user logs
in successfully and after the
event.interactive_login



Tests

Meaningful data providers

I learned this from



Scott
Warren

More information

<http://www.thisprogrammingthing.com/2015/making-dataproviders-more-maintainable>

Common data providers

```
public function getUserData()  
{  
    return array(  
        array(true, false, false),  
        array(true, true, false),  
        array(true, true, true),  
    );  
}
```

```
/** @dataProvider getUserData */  
public function testRegistration($register, $enable, $notify)  
{  
    // ...  
}
```

When an error happens...


```
$ phpunit -c app
```

```
F..
```

```
Time: 137 ms, Memory: 12.50Mb
```

```
There was 1 failure:
```

```
1) AppBundle\Tests\Controller\DefaultControllerTest::testRegistration  
   with data set #0 (true, false, false)
```

 very hard to understand
the exact error

Meaningful data providers

```
public function getUserData()  
{  
    return array(  
        'register user only'  
        'register and enable'  
        'register, enable and notify'  
    );  
}
```

array keys are the
labels of each data set



=> array(true, false, false),
=> array(true, true, false),
=> array(true, true, true),

```
/** @dataProvider getUserData */  
public function testRegistration($register, $enable, $notify)  
{  
    // ...  
}
```

When an error happens...


```
$ phpunit -c app
```

```
F..
```

```
Time: 137 ms, Memory: 12.50Mb
```

```
There was 1 failure:
```

```
1) AppBundle\Tests\Controller\DefaultControllerTest::testRegistration  
   with data set "register user only" (true, false, false)
```

 meaningful error
messages

Generating data providers

I learned this from



Tugdual
Saunier

More information

<http://php.net/manual/en/language.generators.overview.php>

Common data providers

```
public function dataProvider()  
{  
    public function providerBasicDrivers()  
    {  
        return array(  
            array('doctrine.orm.cache.apc.class',      array('type' => 'apc')),  
            array('doctrine.orm.cache.array.class',     array('type' => 'array')),  
            array('doctrine.orm.cache.xcache.class',    array('type' => 'xcache')),  
            array('doctrine.orm.cache.wincache.class',  array('type' => 'wincache')),  
            array('doctrine.orm.cache.zenddata.class',  array('type' => 'zenddata')),  
        );  
    }  
}
```

Data providers using PHP generators

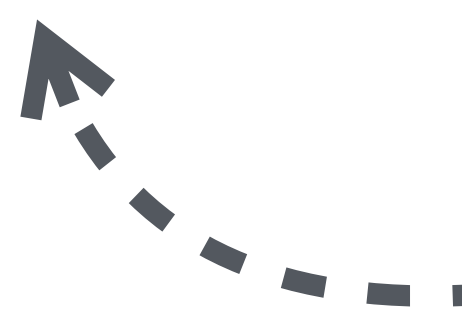
NEW PHP 5.5

```
public function dataProvider()
{
    public function providerBasicDrivers()
    {
        yield ['doctrine.orm.cache.apc.class',      ['type' => 'apc']];
        yield ['doctrine.orm.cache.array.class',    ['type' => 'array']];
        yield ['doctrine.orm.cache.xcache.class',   ['type' => 'xcache']];
        yield ['doctrine.orm.cache.wincache.class', ['type' => 'wincache']];
        yield ['doctrine.orm.cache.zenddata.class', ['type' => 'zenddata']];
    }
}
```


Data providers using PHP generators

NEW PHP 5.5

```
public function dataProvider()
{
    public function providerBasicDrivers()
    {
        yield ['doctrine.orm.cache.apc.class',      ['type' => 'apc']];
        yield ['doctrine.orm.cache.array.class',     ['type' => 'array']];
        yield ['doctrine.orm.cache.xcache.class',    ['type' => 'xcache']];
        yield ['doctrine.orm.cache.wincache.class',  ['type' => 'wincache']];
        yield ['doctrine.orm.cache.zenddata.class',  ['type' => 'zenddata']];
    }
}
```

 it reduces memory consumption

Better setUp and tearDown

I learned this from



Sebastian
Bergmann

More information

<https://phpunit.de/manual/current/en/appendixes.annotations.html#appendixes.annotations.before>

Common use of setUp() method in tests

```
class IntegrationTest extends \PHPUnit_Framework_TestCase
{
    private $rootDir;
    private $fs;

    public function setUp()
    {
        $this->rootDir = realpath(__DIR__.'../../../../../');
        $this->fs = new Filesystem();

        if (!$this->fs->exists($this->rootDir.'/symfony.phar')) {
            throw new \RuntimeException("...");
        }
    }
}
```

Common use of setUp() method in tests

```
class IntegrationTest extends
```

```
{  
    private $rootDir;  
    private $fs;
```

```
    public function setUp()  
    {
```

```
        $this->rootDir = realpath(__DIR__.'../../../../../');  
        $this->fs = new Filesystem();
```

```
        if (!$this->fs->exists($this->rootDir.'/symfony.phar')) {  
            throw new RuntimeException("...");  
        }
```

```
    }
```

```
}
```

```
public function setUp()
```

```
public function tearDown()
```

```
public static function setUpBeforeClass()
```

```
public static function tearDownAfterClass()
```

Better alternative: @before annotation

```
class IntegrationTest extends \PHPUnit_Framework_TestCase
{
    private $rootDir;
    private $fs;

    /** @before */
    public function initialize()
    {
        $this->rootDir = realpath(__DIR__.'../../../../../');
        $this->fs = new Filesystem();
    }

    /** @before */
    public function checkPharFile() {
        if (!$this->fs->exists($this->rootDir.'/symfony.phar')) {
            throw new \RuntimeException("...");
        }
    }
}
```

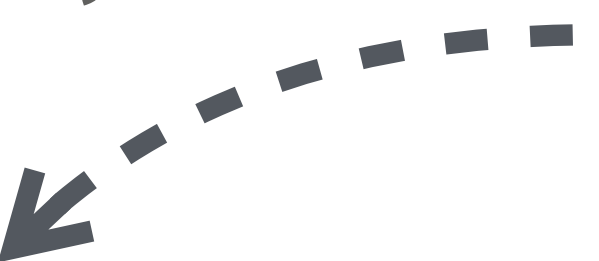
Better alternative: @before annotation

```
class IntegrationTest extends \PHPUnit_Framework_TestCase
{
    private $rootDir;
    private $fs;

    /** @before */
    public function initialize()
    {
        $this->rootDir = realpath(__DIR__.'../../../../../');
        $this->fs = new Filesystem();
    }

    /** @before */
    public function checkPharFile() {
        if (!$this->fs->exists($this->rootDir.'/symfony.phar')) {
            throw new \RuntimeException("...");
        }
    }
}
```

you can define any number of **@before** methods



Better alternative: @before annotation

```
class IntegrationTest extends \PHPUnit_Framework_TestCase
{
```

```
    private $rootDir;
    private $fs;
```

```
    /** @before */
```

```
    public function initialize()
    {
```

```
        $this->rootDir = realpath(__DIR__.'../../../../../');
        $this->fs = new Filesystem();
    }
```

```
    /** @before */
```

```
    public function checkPharFile() {
```

```
        if (!$this->fs->exists($this->rootDir.'/symfony.phar')) {
            throw new \RuntimeException("...");
        }
    }
```

```
}
```

```
}
```

```
}
```

you can define any number
of **@before** methods



order matters (first
method is executed first)



Alternatives to setUp and tearDown

```
class IntegrationTest extends \PHPUnit_Framework_TestCase
{
    /** @before */
    public function initialize() { }

    /** @after */
    public function clean() { }

    /** @beforeClass */
    public static function initialize() { }

    /** @afterClass */
    public static function initialize() { }
}
```


Avoid risky tests

I learned this from



Sebastian
Bergmann

More information

<https://phpunit.de/manual/current/en/risky-tests.html>

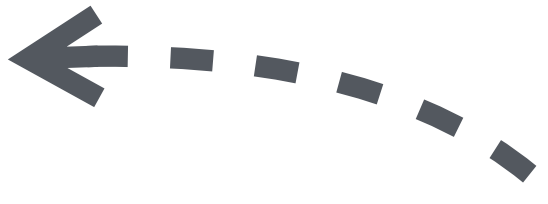
Common PHPUnit configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<phpunit xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://schema.phpunit.de/4.1/phpunit.xsd"
    backupGlobals="false"
    colors="true"
    bootstrap="vendor/autoload.php"
>
    ...
</phpunit>
```

Common PHPUnit configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<phpunit xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://schema.phpunit.de/4.1/phpunit.xsd"
  backupGlobals="false"
  colors="true"
  bootstrap="vendor/autoload.php"
  beStrictAboutTestsThatDoNotTestAnything="true"
  checkForUnintentionallyCoveredCode="true"
  beStrictAboutOutputDuringTests="true"
  beStrictAboutTestSize="true"
  beStrictAboutChangesToGlobalState="true"
>
...
</phpunit>
```

this strict configuration
will force you to create
better tests



mini-tip

Deprecation stack traces

I learned this from



Nicolas
Grekas

More information

<https://github.com/symfony/symfony/pull/16709>

Use the PHPUnit Bridge to detect deprecations

```
$ composer require --dev symfony/phpunit-bridge
```

```
$ phpunit -c app
```

```
PHPUnit by Sebastian Bergmann and contributors.
```

```
.....
```

```
Time: 3.7 seconds, Memory: 75.00Mb
```

```
OK (17 tests, 21 assertions)
```

```
Remaining deprecation notices (368)
```

When a deprecation occurs...

```
2. /Users/javier? (bash)
$ phpunit -c app
PHPUnit 4.6.6 by Sebastian Bergmann and contributors.

Configuration read from /Users/javier/sf/symfony-demo/app/phpunit.xml.dist

.....

Time: 3.7 seconds, Memory: 75.00Mb

OK (17 tests, 21 assertions)

Remaining deprecation notices (368)

The class "Symfony\Bundle\AsseticBundle\Config\AsseticResource" is performing resource checking through
ResourceInterface::isFresh(), which is deprecated since 2.8 and will be removed in 3.0: 368x
    128x in DefaultControllerTest::testPublicUrls from AppBundle\Tests\Controller
    128x in DefaultControllerTest::testSecureUrls from AppBundle\Tests\Controller
    32x in BlogControllerTest::testIndex from AppBundle\Tests\Controller
    32x in BlogControllerTest::testIndex from AppBundle\Tests\Controller\Admin
    32x in BlogControllerTest::testAdministratorUsersCanAccessToTheBackend from AppBundle\Tests\Controller\Admin
    16x in BlogControllerTest::testRegularUsersCannotAccessToTheBackend from AppBundle\Tests\Controller\Admin
```

This information is useful, but incomplete. Often you need the stack trace of the code that triggered the deprecation.



On-demand deprecation stack traces


// BEFORE

```
$ phpunit -c app
```

// AFTER

```
$ SYMFONY_DEPRECATIONS_HELPER=/Blog::index/ \  
  phpunit -c app
```

A regular expression that
is matched against
className::methodName



On-demand deprecation stack traces

```
$ SYMFONY_DEPRECATIONS_HELPER=/.*/ phpunit -c app
```

The class "Symfony\Bundle\AsseticBundle\Config\AsseticResource" is performing resource checking through ResourceInterface::isFresh(), which is deprecated since 2.8 and will be removed in 3.0

It stops at the first
deprecation and shows
its stack trace

Stack trace:

```
#0 vendor/symfony/symfony/src/Symfony/Component/Config/Resource/BCResourceInterfaceChecker.php(32):  
    trigger_error()  
#1 app/bootstrap.php.cache(3061): Symfony\Component\Config\Resource\BCResourceInterfaceChecker->isFresh()  
#2 vendor/symfony/symfony/src/Symfony/Component/Config/ResourceCheckerConfigCacheFactory.php(45):  
    Symfony\Component\Config\ResourceCheckerConfigCache->isFresh()  
#3 vendor/symfony/symfony/src/Symfony/Component/Routing/Router.php(302): Symfony\Component\Config\ResourceCheckerConfigCacheFactory->cache()  
#4 vendor/symfony/symfony/src/Symfony/Component/Routing/Router.php(250): Symfony\Component\Routing\Router->getMatcher()  
#5 vendor/symfony/symfony/src/Symfony/Component/HttpKernel/EventListener/RouterListener.php(154): Symfony\Component\Routing\Router->matchRequest()  
#6 [internal function]: Symfony\Component\HttpKernel\EventListener\RouterListener->onKernelRequest()  
#7 vendor/symfony/symfony/src/Symfony/Component/EventDispatcher/Debug/WrappedListener.php(61): call_user_func()  
#8 [internal function]: Symfony\Component\EventDispatcher\Debug\WrappedListener->__invoke()  
#9 vendor/symfony/symfony/src/Symfony/Component/EventDispatcher/EventDispatcher.php(181): call_user_func()  
#10 vendor/symfony/symfony/src/Symfony/Component/EventDispatcher/EventDispatcher.php(46): Symfony\Component\EventDispatcher\EventDispatcher->doDispatch()  
#11 vendor/symfony/symfony/src/Symfony/Component/EventDispatcher/Debug/TraceableEventDispatcher.php(132): Symfony\Component\EventDispatcher\EventDispatcher->  
    dispatch()  
#12 app/bootstrap.php.cache(3178): Symfony\Component\EventDispatcher\Debug\TraceableEventDispatcher->dispatch()  
#13 app/bootstrap.php.cache(3151): Symfony\Component\HttpKernel\HttpKernel->handleRaw()  
#14 app/bootstrap.php.cache(3302): Symfony\Component\HttpKernel\HttpKernel->handle()  
#15 app/bootstrap.php.cache(2498): Symfony\Component\HttpKernel\DependencyInjection\ContainerAwareHttpKernel->handle()  
#16 vendor/symfony/symfony/src/Symfony/Component/HttpKernel/Client.php(79): Symfony\Component\HttpKernel\Kernel->handle()  
#17 vendor/symfony/symfony/src/Symfony/Bundle/FrameworkBundle/Client.php(131): Symfony\Component\HttpKernel\Client->doRequest()  
#18 vendor/symfony/symfony/src/Symfony/Component/BrowserKit/Client.php(317): Symfony\Bundle\FrameworkBundle\Client->doRequest()  
#19 src/AppBundle/Tests/Controller/Admin/BlogControllerTest.php(42): Symfony\Component\BrowserKit\Client->request()  
#20 [internal function]: AppBundle\Tests\Controller\Admin\BlogControllerTest->testRegularUsersCannotAccessToTheBackend()  
#21 {main}
```

Services

Smoke testing for services

I learned this from



Benjamin
Eberlei

More information

http://www.whitewashing.de/2015/06/12/the_containertest.html

**«Smoke Tests are an
early warning for when
things go wrong»**

Automatic "smoke testing" for services

```
public function testContainerServices()  
{  
    $client = static::createClient();  
    foreach ($client->getContainer()->getServiceIds() as $serviceId) {  
        $service = $client->getContainer()->get($serviceId);  
        $this->assertNotNull($service);  
    }  
}
```

Automatic "smoke testing" for services

```
public function testContainerServices()  
{  
    $client = static::createClient();  
    foreach ($client->getContainer()->getServiceIds() as $serviceId) {  
        $service = $client->getContainer()->get($serviceId);  
        $this->assertNotNull($service);  
    }  
}
```

get all services defined
by the container



Automatic "smoke testing" for services

```
public function testContainerServices()  
{  
    $client = static::createClient();  
    foreach ($client->getContainer()->getServiceIds() as $serviceId) {  
        $service = $client->getContainer()->get($serviceId);  
        $this->assertNotNull($service);  
    }  
}
```

get all services defined
by the container



try to instantiate all of
them to look for errors



Automatic "smoke testing" for services

```
public function testContainerServices()  
{  
    $client = static::createClient();  
    foreach ($client->getContainer()->getServiceIds() as $serviceId) {  
        try {  
            $service = $client->getContainer()->get($serviceId);  
            $this->assertNotNull($service);  
        } catch(InactiveScopeException $e) { }  
    }  
}
```

 you may need this **try ... catch** to avoid
issues with request related services

Manual "smoke testing" for important services

```
public static function dataServices()
{
    return [
        ['app.datetimepicker', 'AppBundle\Form\Type\DateTimePickerType'],
        ['app.redirect_listener', 'AppBundle\EventListener\RedirectToPreferredLocaleListener'],
    ];
}

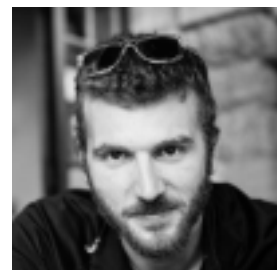
/** @dataProvider dataServices */
public function testImportantServices($id, $class)
{
    $client = static::createClient();
    $service = $client->getContainer()->get($id);

    $this->assertInstanceOf($class, $service);
}
```

check that the service is instantiable and that the returned class is the right one

Services benchmark

I learned this from



Alexandre
Salomé

More information

<https://speakerdeck.com/alexandresalome/french-symfony2-et-performances>

Service instantiation should not take too long

```
public function testContainerServices()  
{  
    $client = static::createClient();  
    foreach ($client->getContainer()->getServiceIds() as $serviceId) {  
        try {  
            ➡ $startedAt = microtime(true);  
            $service = $client->getContainer()->get($serviceId);  
            ➡ $elapsed = (microtime(true) - $startedAt) * 1000;  
            $this->assertLessThan(50, $elapsed);  
        } catch(InactiveScopeException $e) {  
        }  
    }  
}
```

Service instantiation should not take too long

```
public function testContainerServices()  
{  
    $client = static::createClient();  
    foreach ($client->getContainer()->getServiceIds() as $serviceId) {  
        try {  
            ➡ $startedAt = microtime(true);  
            $service = $client->getContainer()->get($serviceId);  
            ➡ $elapsed = (microtime(true) - $startedAt) * 1000;  
            $this->assertLessThan(50, $elapsed);  
        } catch(InactiveScopeException $e) {  
        }  
    }  
}
```

services should be
created in **50ms** or less

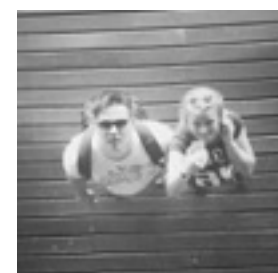
Most expensive services in Symfony Standard

Service	Time (ms)
cache_warmer	42.95
doctrine.orm.default_entity_manager	14.05
web_profiler.controller.router	11.88
swiftmailer.mailer.default	6.65
profiler	5.45
annotation_reader	4.53
doctrine.dbal.default_connection	4.22
form.type_extension.form.validator	4.13
routing.loader	4.02
form.type.choice	3.10

mini-tip

Simpler command interactions

I learned this from



Nikita
Gusakov

More information

[https://github.com/symfony/swiftmailer-bundle/blob/master/Command/
NewEmailCommand.php](https://github.com/symfony/swiftmailer-bundle/blob/master/Command/NewEmailCommand.php)

Common lifecycle of a command

```
class MyCustomCommand extends ContainerAwareCommand
{
    protected function configure()
    {
    }

    protected function execute(InputInterface $input, OutputInterface $output)
    {
    }
}
```


Full lifecycle of a command

```
class MyCustomCommand extends ContainerAwareCommand
```

```
{
```

```
    protected function configure()  
    { }
```

here you can ask the user for
any missing option or argument

```
    protected function initialize(InputInterface $input, OutputInterface $output)  
    { }
```

```
    protected function interact(InputInterface $input, OutputInterface $output)  
    { }
```

```
    protected function execute(InputInterface $input, OutputInterface $output)  
    { }
```

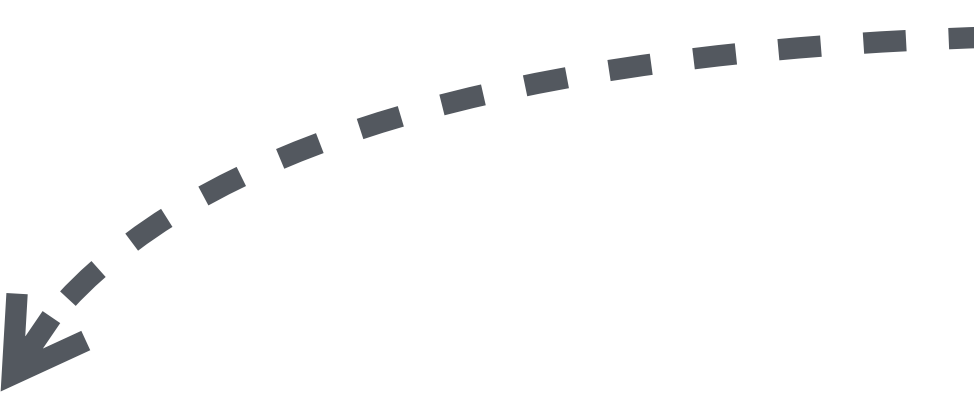
```
}
```

Ask for any missing command argument

```
class MyCommand extends ContainerAwareCommand
{
    // ...

    protected function interact(InputInterface $input, OutputInterface $output)
    {
        $dialog = $this->getHelper('dialog');
        foreach ($input->getArguments() as $argument => $value) {
            if ($value === null) {
                $input->setArgument($argument, $dialog->ask($output,
                    sprintf('<question>%s</question>: ', ucfirst($argument))
                ));
            }
        }
    }
}
```

the **interact()** method is usually very verbose. This is the minimal useful **interact()** method.



Console

Console Style Guide


Most commands are a mess

```
$description = $this->formatSection(  
    'container', sprintf('Information for service <info>%s</info>', $options['id']))  
    ."\\n".sprintf('<comment>Service Id</comment>'          %s', isset($options['id']) ? $options['id'] : '-')  
    ."\\n".sprintf('<comment>Class</comment>'              %s', get_class($service)  
);  
  
$description[] = sprintf('<comment>Scope</comment>'          %s', $definition->getScope(false));  
$description[] = sprintf('<comment>Public</comment>'         %s', $definition->isPublic() ? 'yes' : 'no');  
$description[] = sprintf('<comment>Synthetic</comment>'      %s', $definition->isSynthetic() ? 'yes' : 'no');  
$description[] = sprintf('<comment>Lazy</comment>'           %s', $definition->isLazy() ? 'yes' : 'no');  
  
$this->writeText($description, $options);
```

Most commands are a mess

```
$description = $this->formatSection(  
    'container', sprintf('Information for service <info>%s</info>', $options['id']))  
    ."\n".sprintf('<comment>Service Id</comment>'           %s', isset($options['id']) ? $options['id'] : '-')  
    ."\n".sprintf('<comment>Class</comment>'               %s', get_class($service)  
);  
  
$description[] = sprintf('<comment>Scope</comment>'        %s', $definition->getScope(false));  
$description[] = sprintf('<comment>Public</comment>'       %s', $definition->isPublic() ? 'yes' : 'no');  
$description[] = sprintf('<comment>Synthetic</comment>'    %s', $definition->isSynthetic() ? 'yes' : 'no');  
$description[] = sprintf('<comment>Lazy</comment>'         %s', $definition->isLazy() ? 'yes' : 'no');  
  
$this->writeText($description, $options);
```

you are mixing **content** with
presentation, like in the good
old days of HTML+CSS



**The new "Style Guide"
makes built-in commands
visually consistent.**

You can use it in your own commands too

```
use Symfony\Component\Console\Style\SymfonyStyle;
```

```
class MyCustomCommand
{
    // ...

    protected function execute(InputInterface $input, OutputInterface $output)
    {
        $io = new SymfonyStyle($input, $output);
        $io->...

        // ...
    }
}
```


Displaying a title (BEFORE)

```
// alternative 1
```

```
$formatter = $this->getHelperSet()->get('formatter');  
$formattedBlock = $formatter->formatBlock($title, '', true);  
$output->writeln($formattedBlock);
```

```
// alternative 2
```

```
$title = 'Lorem Ipsum Dolor Sit Amet';  
$output->writeln('<info>'.$title.'</info>');  
$output->writeln(str_repeat('=', strlen($title)));
```

```
// alternative 3
```

```
$output->writeln('');  
$output->writeln('Add User Command Interactive Wizard');  
$output->writeln('-----');
```

Displaying a title (AFTER)

```
use Symfony\Component\Console\Style\SymfonyStyle;
```

```
$io = new SymfonyStyle($input, $output);
```

```
$io->title('Lorem Ipsum Dolor Sit Amet');
```

Displaying a title (AFTER)

```
use Symfony\Component\Console\Style\SymfonyStyle;
```

```
$io = new SymfonyStyle($input, $output);
```

```
$io->title('Lorem Ipsum Dolor Sit Amet');
```



```
$ php app/console app:my-command
```

```
Lorem Ipsum Dolor Sit Amet
```

```
=====
```

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit. I  
turpis. Mauris dictum, leo et bibendum tempus, mi diam fac  
eget interdum nisi risus a massa. Suspendisse non sapien s  
consectetur convallis nec quis odio. Nulla facilisis sapie  
suscipit varius. Vivamus at tincidunt tellus. Suspendisse  
Suspendisse non odio iusto. Fusce imperdiet augue id hibe
```

Displaying a table (BEFORE)

```
$table = new Table($this->getOutput());  
$table->setStyle('compact');  
  
$table->setHeaders(['Parameter', 'Value']);  
  
$table->addRow(['...', '...']);  
$table->addRow(['...', '...']);  
$table->addRow(['...', '...']);  
  
$table->render();
```

Displaying a table (AFTER)

```
use Symfony\Component\Console\Style\SymfonyStyle;
```

```
$headers = ['Parameter', 'Value'];
```

```
$rows = [ ... ];
```

```
$io = new SymfonyStyle($input, $output);
```

```
$io->table($headers, $rows);
```

Displaying a table (AFTER)

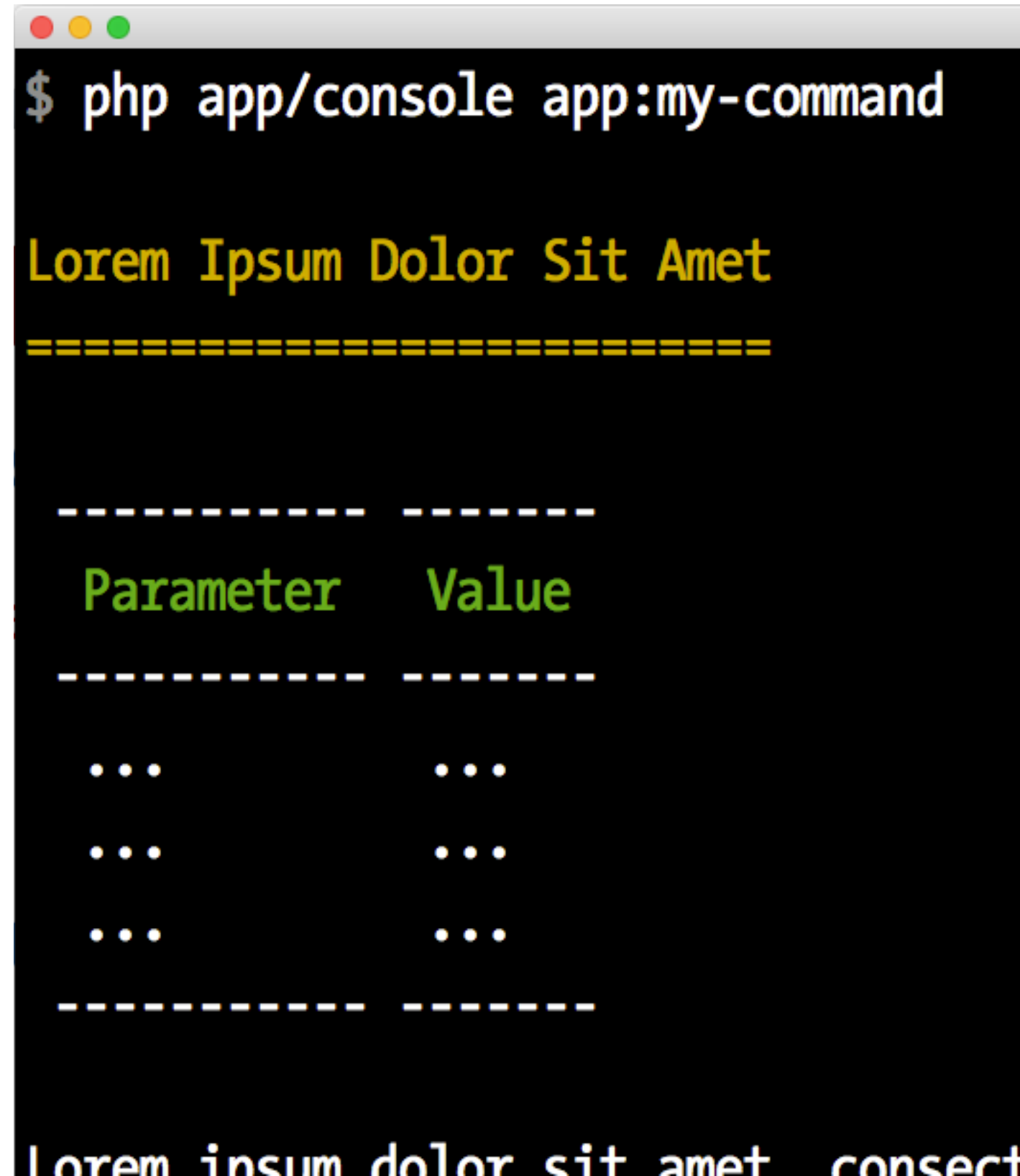
```
use Symfony\Component\Console\Style\SymfonyStyle;
```

```
$headers = ['Parameter', 'Value'];
```

```
$rows = [ ... ];
```

```
$io = new SymfonyStyle($input, $output);
```

```
$io->table($headers, $rows);
```



```
$ php app/console app:my-command

Lorem Ipsum Dolor Sit Amet
=====

-----
Parameter  Value
-----

...        ...
...        ...
...        ...
-----

Lorem ipsum dolor sit amet consectetur
```

All the common features are covered

```
$io = new SymfonyStyle($input, $output);
```

```
$io->title();                      $io->success();
```

```
$io->section();                   $io->error();
```

```
$io->text();                      $io->warning();
```

```
$io->listing();                  $io->ask();
```

```
$io->table();                    $io->askHidden();
```

```
$io->choice();                  $io->confirm();
```


Create consistent commands effortlessly

Lorem Ipsum Dolor Sit Amet
=====

// Duis aute irure dolor in reprehenderit in voluptate velit esse
// cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat

Name	Method	Scheme	Host	Path
-----	-----	-----	----	-----
admin_post_new	ANY	ANY	ANY	/admin/post/new
admin_post_show	GET	ANY	ANY	/admin/post/{id}
admin_post_edit	ANY	ANY	ANY	/admin/post/{id}/edit
admin_post_delete	DELETE	ANY	ANY	/admin/post/{id}
-----	-----	-----	----	-----

! [CAUTION] Lorem ipsum dolor sit amet, consectetur adipisicing elit,
! sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
! Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris.

Consectetur Adipisicing Elit Sed Do Eiusmod

- * Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
- * Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo.
- * Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Bundle namespace:
> AppBundle <Enter>

Bundle name [AcmeAppBundle]:
> <Enter>

Configuration format (yaml, xml, php, annotation) [annotation]:
> <Enter>

Do you want to enable the bundle? (yes/no) [yes]:
> <Enter>

Configuration format (select one) [annotation]:
> yaml
> xml
> php
> annotation

! [NOTE] Duis aute irure dolor in reprehenderit in voluptate velit esse
! cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat.

[OK] Lorem ipsum dolor sit amet, consectetur adipisicing elit

[ERROR] Duis aute irure dolor in reprehenderit in voluptate velit esse.

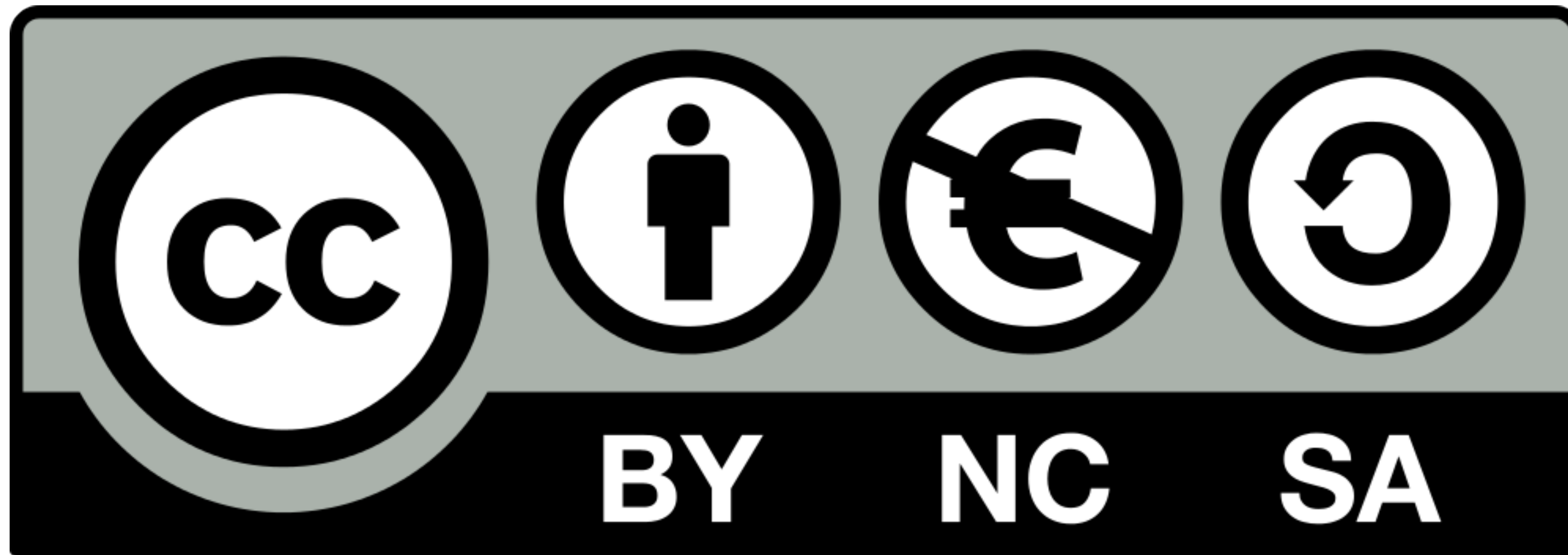
[WARNING] Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi.

Contact information

Contact information

javier.eguiluz@sensiolabs.com

License of this presentation



creativecommons.org/licenses/by-nc-sa/3.0

