Mocks, Stubs, Tests

Marc Weistroff

Alexandre Salomé

SensioLabs

C'qui Marc ?

Marc Weistroff @futurecat

SensioLabs

C’qui lui ?

Alexandre Salomé

(Mock de Marc)

SensioLabs

L’instant quizz

Qui a déjà écrit des tests ?

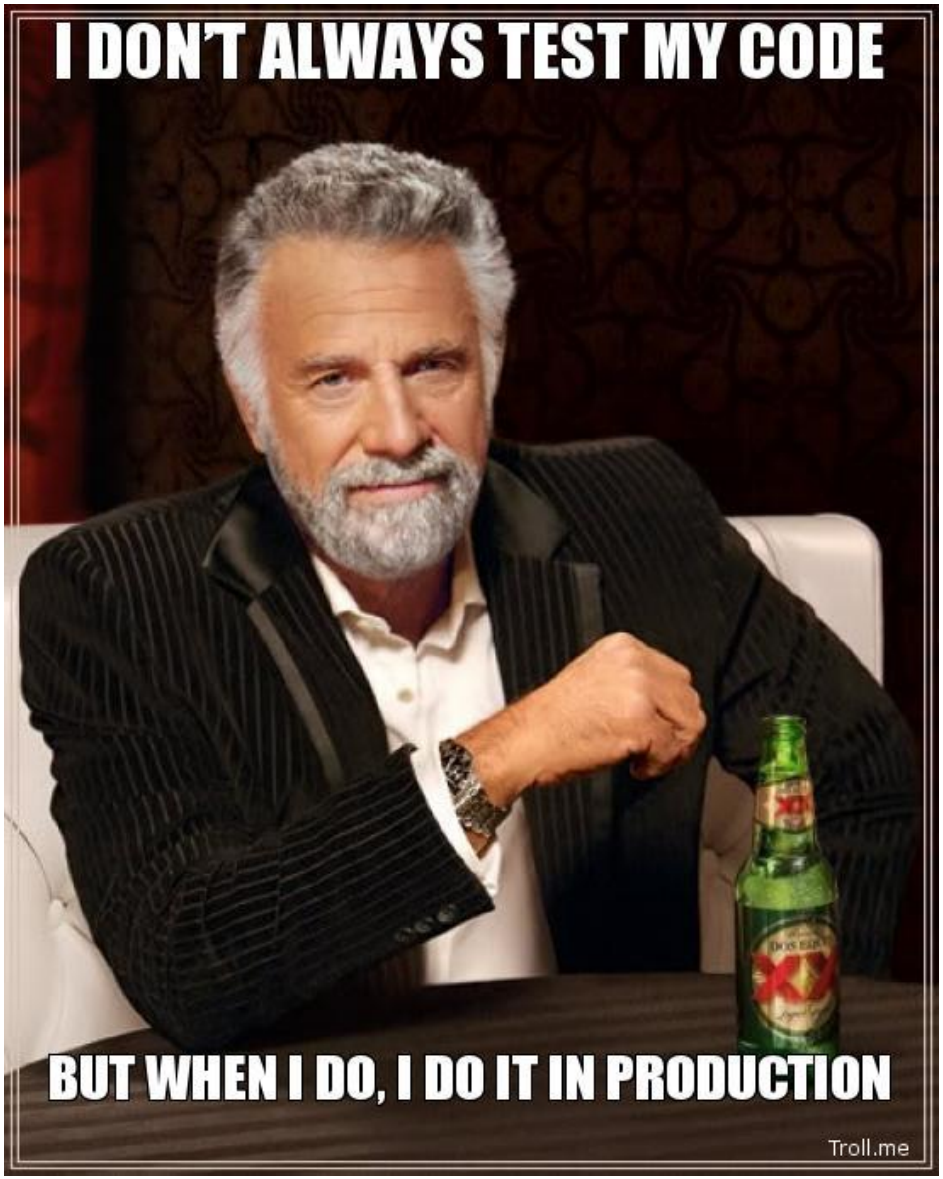
Qui a déjà utilisé PHPUnit ?

Qui a déjà mocké une classe ?

Séance de gym terminée.

Merci

TESTS



SensioLabs

Pourquoi tester?

- Assurance
  - Stabilité
  - Protection
  - Prévention des effets de bord
- Liberté
  - Confiance
  - Prévoyance - Santé
- Documentation

SensioLabs

Dans quels cas ne pas tester ?

- Prototyping

Dans quels cas ne pas tester ?

- Prototyping
- ? ? ? ? ?

Dans quels cas ne pas tester ?

- Prototyping
- C'est tout

Quel format pour mes tests ?

- Objectif : être capable de dire « ça marche »
- Le format dépend de ce qu'on teste
  - Librairie PHP
    - Tests unitaires
  - Application Web
    - Tests fonctionnels
  - Une voiture qu'on veut acheter
    - On roule avec

Le format des tests dépend du sujet

SensioLabs

Une bonne suite de tests, c'est quoi ?

" Une bonne suite de test,
c'est une suite qui me servira"

Moi

Quand je casserais quelque chose

Elle ne me ralentira pas dans mon travail

Le code coverage sera de 100%

SensioLabs

Une bonne suite de tests, c'est quoi ?

" Une bonne suite de test,
c'est une suite qui me servira"

Moi

Quand je casserais quelque chose

Elle ne me ralentira pas dans mon travail

~~Le code coverage sera de 100%~~

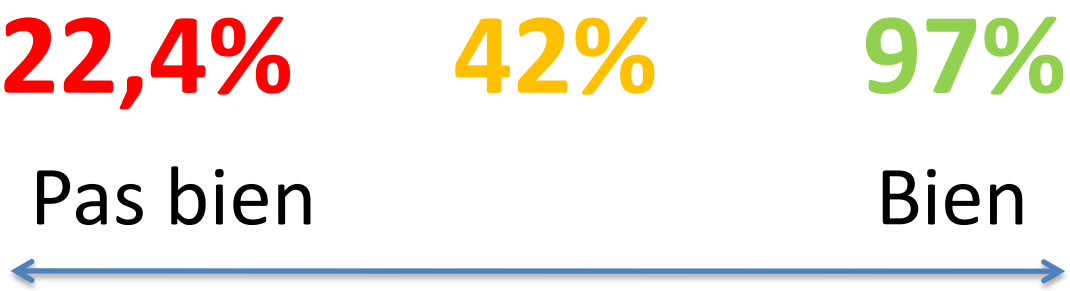
SensioLabs

Sortez couverts !

CODE COVERAGE

SensioLabs

Un indicateur



Un **mauvais** indicateur

SensioLabs

Un **mauvais** indicateur

- Focalisation sur code coverage = **erreur**

Un **mauvais** indicateur

- Focalisation sur code coverage = **erreur**
- La méthode n'est pas correcte
- Nous éloigne de nos envies premières

Un **mauvais** indicateur

- Focalisation sur code coverage = **erreur**
- La méthode n'est pas correcte
- Nous éloigne de nos envies premières
  - Coverage à 100% ne veut pas dire "tout est testé"

Un **mauvais** indicateur

- Focalisation sur code coverage = **erreur**
- La méthode n'est pas correcte
- Nous éloigne de nos envies premières
  - Coverage à 100% ne veut pas dire "tout est testé"
  - Ca veut dire "tout le code a été exécuté une fois"

Un **mauvais** indicateur

```
class PHPTour
{
    public $totalCost;
    public $attendees;

    public function getPlacePrice()
    {
        return $this->totalCost / $this->attendees;
    }
}
```

SensioLabs

Un **mauvais** indicateur

```
class PHPTourTest extends PHPUnit_Framework_TestCase
{
    public function testForCoverage()
    {
        $tour = new PHPTour();
        $tour->totalCost = 40000;
        $tour->attendees = 100;

        $this->assertEquals(400, $tour->getPlacePrice());
   }
}
```

SensioLabs

Un **mauvais** indicateur

```
class PHPTourTest extends PHPUnit_Framework_TestCase
{
    public function testForCoverage()
    {
        our =  Tour
        our-> talC  = 0000
        our-  tend  = 00;

        s->a  qual  tour- etPl  ice());
    }
}
```

100%

SensioLabs

Un **mauvais** indicateur

```
$tour->attendees = 0;
```

SensioLabs

Un **mauvais** indicateur

```
$tour->attendees = 0;
```

Division par zéro

SensioLabs

Un **mauvais** indicateur

Se concentrer sur le coverage

fait oublier l'objectif premier

=

Tester son code

Un **mauvais** indicateur

- Dire « il faut plus de X % de coverage »
  - CA SERT A RIEN !
- Certaines parties ne méritent pas un test

```php
<?php

class CodeCoverage
{
    private $field1;
    // …
    private $fieldN;
    // …
    private $field30;

    // Powerpoint coding standards
    public function setFieldN($n) { $this->fieldn = $n; }
    public function getFieldN() { return $this->fieldN; }
}
```

SensioLabs

```php
<?php

class CodeCoverage
{
    // …
    public function compute()
    {
        $troll = $field1 * $field2 / (($field3 * $field4) * $field5) *
$field6 / $field7 * $field8;
        $troll2 = $field9 * $field10 / cos($field11);
        $troll3 = $field12 * tan($field13);

        return $troll * log($troll2) / sin($troll3) * $field14  /*
(;;)(^_^)(;;) */;
    }
}
```

SensioLabs

```php
<?php

class CodeCoverageTest extends PHPUnit_Framework_TestCase
{
    public function testSetField1AndGetField1ReturnsCorrectValue()
    {
        $cc = new CodeCoverage();
        $cc->setField1('fuu');
        $this->assertEquals('fuu', $cc->getField1());
    }

    // repeat for each fields
}
```

```
$ phpunit –coverage-html <dir>
```

Résultats et métriques

- Code coverage = 98.13%
- Valeur ajoutée du test = 0%
- Efficacité du test = 0%
- Risque pour le projet = 100%
- Risque de se faire virer en cas de bug = 100%
- Chance de mieux écrire ses tests après = ?

La bonne métrique ?

- Le coverage est un OUTIL, pas une finalité

La bonne métrique ?

- Le coverage est un OUTIL, pas une finalité
- Savoir l'interpréter
  - Pas de coverage = pas de test
  - N% de coverage = pas d'interprétation, lire les tests

La bonne métrique ?

- Le coverage est un OUTIL, pas une finalité
- Savoir l'interpréter
  - Pas de coverage = pas de test
  - N% de coverage = pas d'interprétation, lire les tests
  - 100% = OK, il y a un problème

QUALITE IMPLEMENTATION, DESIGN, ARCHITECTURE

SensioLabs

Ça pue ou ça roxx ?

```
<?php

// Insert your code here
```

SensioLabs

Ça pue ou ça roxx ?

```php
<?php

class Car
{
    public function __construct()
    {
        $this->engine = new Engine();
     }
}
```

SensioLabs

Ça pue ou ça roxx ?

```php
<?php

class Car
{
    public function __construct()
    {
        $this->engine = new Engine();
     }
}
```

SensioLabs

Ça pue ou ça roxx ?

```php
<?php

class Car
{
    public function __construct(EngineInterface $engine)
    {
        $this->engine = $engine;
     }
}
```

SensioLabs

Ça pue ou ça roxx ?

```php
<?php

class Car
{
    public function __construct(EngineInterface $engine)
    {
        $this->engine = $engine;
     }
}
```

SensioLabs

Ça pue ou ça roxx ?

```php
<?php

class SausageShop
{
    public function serve()
    {
        $sausage =
            SausageFactory::getNewSausage();
        $bread =
            BreadFactory::getNewBread();

        return
            $bread->upgradeWith($sausage);
    }
}
```

SensioLabs

Ça pue ou ça roxx ?

```php
<?php

class SausageShop
{
    public function serve()
    {
        $sausage =
            SausageFactory::getNewSausage();
        $bread =
            BreadFactory::getNewBread();

        return
            $bread->upgradeWith($sausage);
    }
}
```

SensioLabs

Ça pue ou ça roxx ?

```php
<?php

class SausageShop
{
    public function __construct(
     SausageFactory $sausageFactory,
     BreadFactory $breadFactory )
    {
        $this->sausageFactory = $sausageFactory;
        $this->breadFactory = $breadFactory;
    }

    public function serve()
    {
       $sausage =
          $this->sausageFactory->getNewSausage();
       $bread =
          $this->breadFactory->getNewBread();

       return $bread->upgradeWith($sausage);
    }
}
```

SensioLabs

Ça pue ou ça roxx ?

```php
<?php

class SausageShop
{
    public function __construct(
     SausageFactory $sausageFactory,
     BreadFactory $breadFactory )
    {
        $this->sausageFactory = $sausageFactory;
        $this->breadFactory = $breadFactory;
    }

    public function serve()
    {
       $sausage =
          $this->sausageFactory->getNewSausage();
       $bread =
          $this->breadFactory->getNewBread();

       return $bread->upgradeWith($sausage);
    }
}
```

SensioLabs

Ça pue ou ça roxx ?

```
<?php

class Customer
{
    public function orderBeer()
    {
        $url = 'http://waiter/bring-me-a-
beer/erdinger';
        return file_get_contents($url);
    }
}
```

SensioLabs

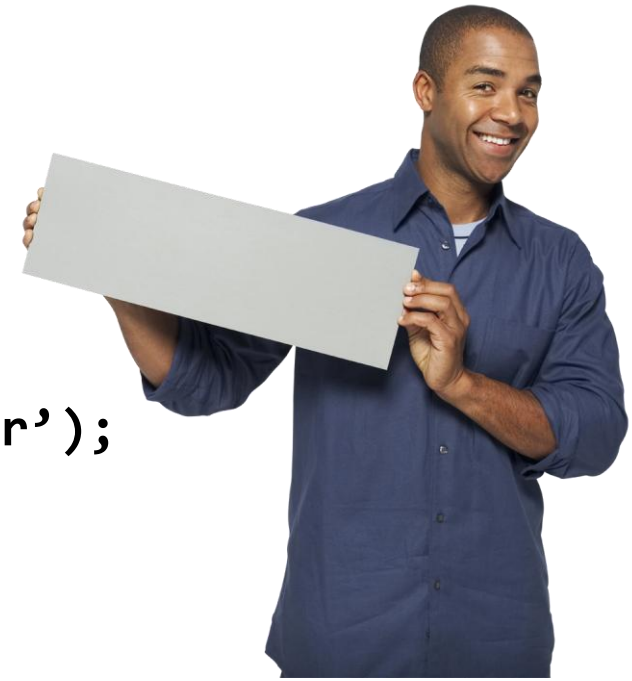
Ça pue ou ça roxx ?

```php
<?php

class Customer
{
    public function orderBeer()
    {
        $url = 'http://waiter/bring-me-a-
beer/erdinger';
        return file_get_contents($url);
    }
}
```

SensioLabs

Ça pue ou ça roxx ?

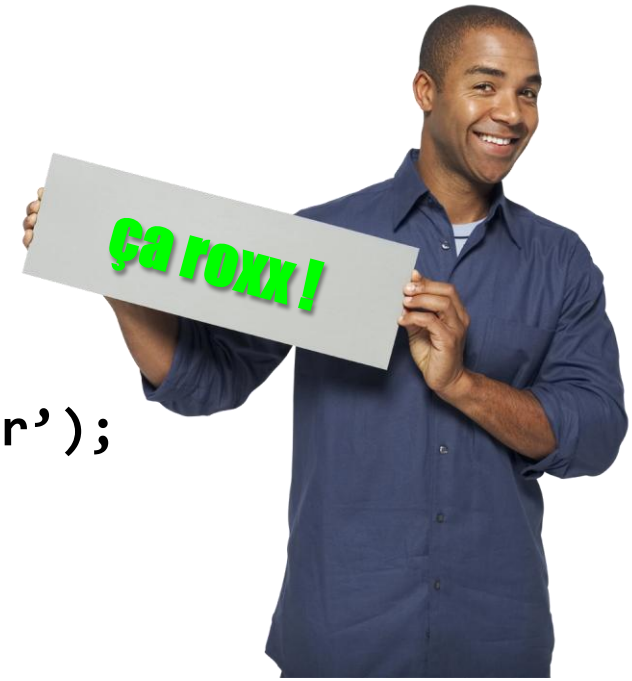
```php
<?php

class Customer
{
    public function orderBeer(
          WaiterApi $api)
    {
        return $api->bring('erdinger');
    }
}
```



SensioLabs

Ça pue ou ça roxx ?

```php
<?php

class Customer
{
    public function orderBeer(
          WaiterApi $api)
    {
        return $api->bring('erdinger');
    }
}
```



SensioLabs

Design by contract

~~SINGLETONS~~

mauvais design

=

difficilement testable

difficilement testable

=

mauvais design

couplage fort

=

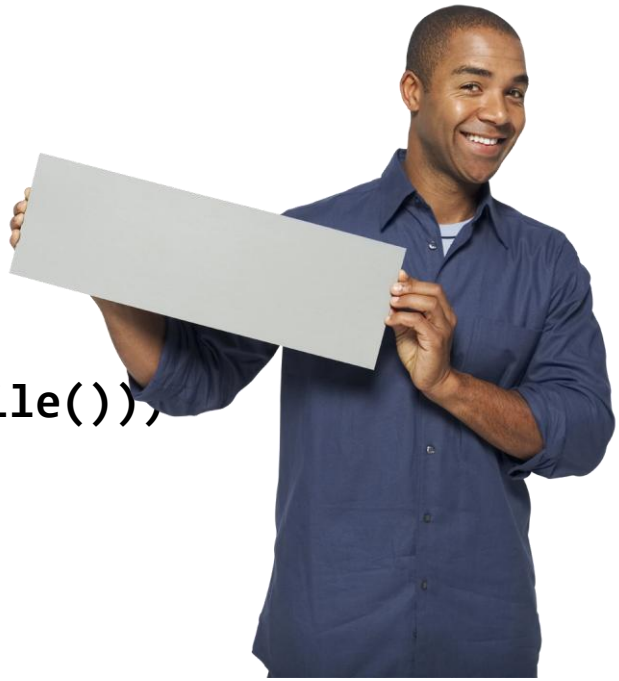
mauvais design

BONUS !!!!

SensioLabs

Ça pue ou ça roxx ?

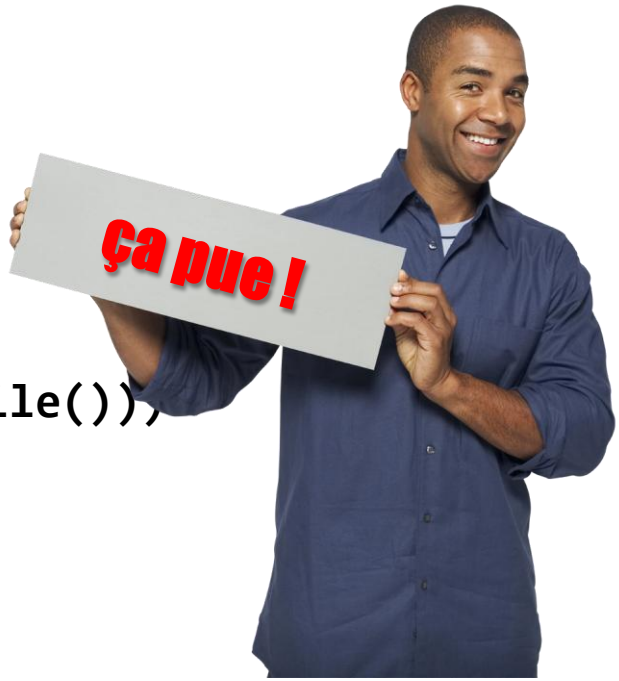
```php
<?php

abstract class BaseMedia
{
    const TYPE_IMAGE;
    abstract public function getFile();

    public function getType() {
      switch (get_extension($this->getFile()))
      {
         case 'png‘:
           return self::TYPE_IMAGE;
      }
}
```



SensioLabs

Ça pue ou ça roxx ?

```php
<?php

abstract class BaseMedia
{
    const TYPE_IMAGE;
    abstract public function getFile();

    public function getType() {
      switch (get_extension($this->getFile()))
      {
         case 'png‘:
           return self::TYPE_IMAGE;
      }
}
```



SensioLabs

Ça pue ou ça roxx ?

```php
<?php

Interface MediaInterface
{
    const TYPE_IMAGE;

    public function getType();
}
```

SensioLabs

AUTOMATISATION, INDÉPENDANCE

SensioLabs

Installation

- Je veux trouver un fichier **README** qui me dit comment utiliser votre code, sans comprendre comment il marche
- Si je ne peux pas lancer la suite de test de votre projet sur ma machine, je ne suis pas content

Isolation

- Les tests ne doivent pas se baser sur l'état de la machine sur laquelle ils tournent
- Un bon test est un test isolé
  - Pas d'appels réseaux
  - Pas de modification de l'état du système (BDD, fichiers) SANS restauration de l'état initial

Oui mais…

- Pas d'appels réseaux ?
- Indépendance ?

Les mocks/stubs à notre secours !

MOCKS & STUBS

SensioLabs

Test doubles

- Les mocks et les stubs sont des « faux »
- On donne une liste d'attentes
- Ils permettent d'accélérer les tests
- Ils évitent de faire de vrais appels

Pré-requis

- Code à tester suffisamment découplé
  - Injection de dépendances
  - Separation of concerns
- Pas de découplage, pas de tests
- Pas de tests… pas de tests

Points communs et différences

Stub

- « Test double »
- Permet de remplacer une dépendance de la classe testée.
- Permet de retourner des valeurs préconfigurées pour certaines méthodes.
- Permet de contrôler la chaine d'exécution (ex: différents comportements suivant ce que le Stub renvoie)

Mock

- « Test double »
- Stub++
- Permet de vérifier des expectations
  - Méthode a bien été appelée
  - Paramètres de la méthode ont étés x et y

Comment mocker une classe ?

1. La pointer du doigt

2. Faire “HA HA!”

SensioLabs

Comment mocker une classe ?

- Dépendance injectée « mockée »

Comment mocker une classe ?

```php
<?php

class MyClassTest extends PHPUnit_Framework_TestCase
{
    public function testFoo()
    {
        $engine = $this->getMock('Engine');
        $car = new Car($engine);
    }
}
```

En fait…
c'est un stub

Tous les appels à Engine renverront « null »

SensioLabs

Quelle différence ?

- Pour PHPUnit, pas grand-chose
- D'un point de vue sémantique
  - Stub = objet qui est juste présent
  - Mock = objet avec des attentes
- Un mock va définir un contrat

Comment mocker une classe ?

```php
<?php

// ...

$engine = $this->getMock('Engine');
$engine
    ->expects($this->once())
    ->method('start')
    ->will($this->returnValue(true))
;
```

C'est un mock, un vrai

Un contrat est établi

SensioLabs

Interface fluide

- `$mock`
- `->expects(…)`
- `->method('name')`
- `->with(…)`
- `->will(…)`

Interface fluide

```
expects(PHPUnit_…_Matcher_Invocation)

    $this->exactly($n)
    $this->never()
    $this->any()
    $this->atLeastOnce()
```

SensioLabs

Interface fluide

```
with(PHPUnit_Framework_Constraint)

    $this->exactly($n)
    $this->arrayHasKey ('...')
    $this->any()
```

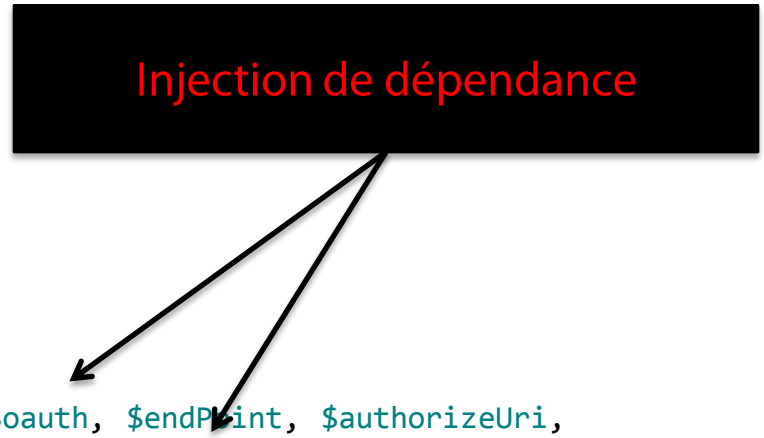
SensioLabs

Interface fluide

```
will(PHPUnit_Framework_MockObject_Stub)

    $this->returnValue($value)
    $this->returnArgument($n)
    $this->returnCallback($c)
    $this->throwException($e)
    $this->onConsecutiveCalls(1, 2, 3)
```

CAS CONCRET

SensioLabs



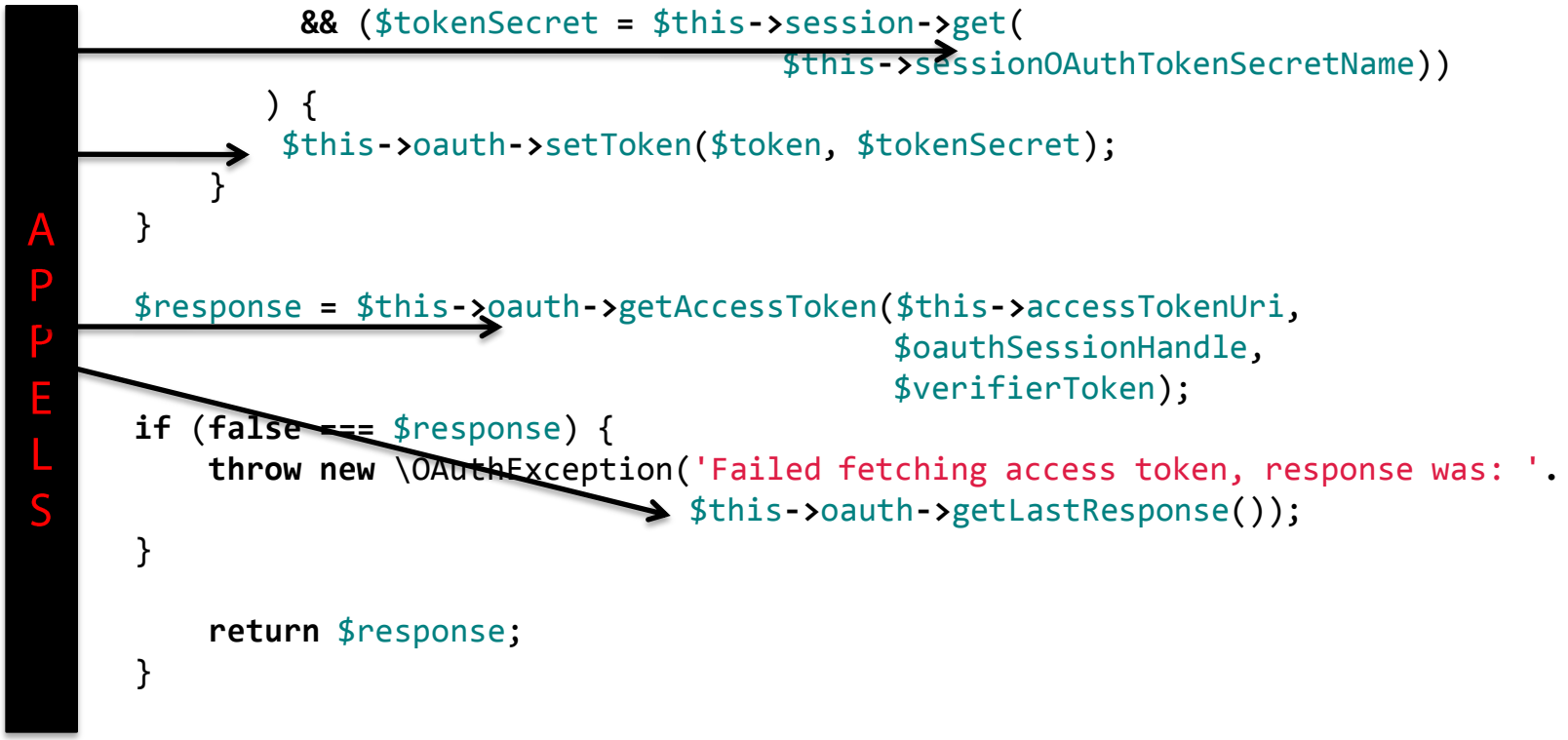

```php
<?php

namespace Sensio\Bundle\PlatformBundle\OAuth;

use Symfony\Component\HttpFoundation\Session;

class OAuthClient
{
    // … properties declaration …
    public function __construct($serviceName, \OAuth $oauth, $endPoint, $authorizeUri,
                    $requestTokenUri, $accessTokenUri, Session $session = null)
    {
        $this->serviceName = $serviceName;
        $this->oauth = $oauth;
        $this->endPoint = $endPoint;

        $this->authorizeUri    = $authorizeUri;
        $this->requestTokenUri = $requestTokenUri;
        $this->accessTokenUri  = $accessTokenUri;

        $this->session = $session;
        $this->sessionOAuthTokenName = sprintf('oauth/_%s.oauth_token',
                                               $this->serviceName);
        $this->sessionOAuthTokenSecretName = sprintf('oauth/_%s.oauth_token_secret',
                                                     $this->serviceName);
    }
    // … other methods …
```

SensioLabs

```php
public function getRequestToken($redirectUri = null)
{
    if ($this->session) {
        $this->session->remove($this->sessionOAuthTokenName);
        $this->session->remove($this->sessionOAuthTokenSecretName);
    }

    $requestToken = $this->oauth->getRequestToken(
                                    $this->requestTokenUri,
                                    $redirectUri);
    if ($requestToken === false) {
        throw new \OAuthException('Failed fetching request token,
                                  response was: ’
                                  .$this->oauth->getLastResponse());
    }

    if ($this->session) {
        $this->session->set($this->sessionOAuthTokenName,
                            $requestToken['oauth_token']);
        $this->session->set($this->sessionOAuthTokenSecretName,
                            $requestToken['oauth_token_secret']);
    }

    return $requestToken;
}
```

APPELS

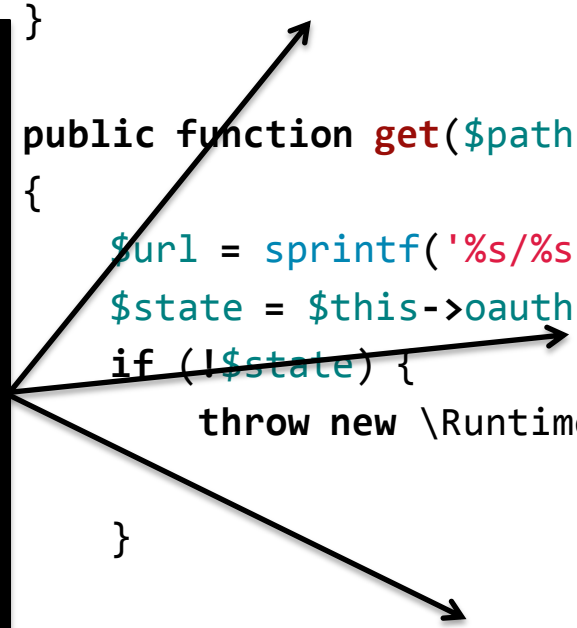
SensioLabs


APPELS
public function retrieveAccessToken($oauthSessionHandle = '', $verifierToken)
{
if ($this->session) {
if (($token = $this->session->get($this->sessionOAuthTokenName))
&& ($tokenSecret = $this->session->get(
$this->sessionOAuthTokenSecretName))
) {
$this->oauth->setToken($token, $tokenSecret);
}
}
$response = $this->oauth->getAccessToken($this->accessTokenUri,
$oauthSessionHandle,
$verifierToken);
if (false === $response) {
throw new \OAuthException('Failed fetching access token, response was: '.
$this->oauth->getLastResponse());
}
return $response;
}

SensioLabs

```php
public function setToken($token, $tokenSecret)
{
    $this->oauth->setToken($token, $tokenSecret);
}

public function get($path)
{
    $url = sprintf('%s/%s', $this->endPoint, ltrim($path, '/'));
    $state = $this->oauth->fetch($url);
    if (!$state) {
        throw new \RuntimeException(sprintf(Can\'t fetch resource %s',
                                            $url));
    }

    return $this->oauth->getLastResponse();
}
```

APPELS



SensioLabs

Et du coup…

- On peut mocker Session et OAuth

Par exemple

```
class myTest extends PHPUnit_Framework_TestCase {
   public function testGetRequestToken() {
       $oauth   = $this->getMock('OAuth');
       $oauth
           ->expects($this->once())
           ->method('getRequestToken')
           ->will($this->returnValue('ok'))
       ;

       $client = new OAuthClient(
           'foo', $oauth,
           'bar', 'bar', 'bar', 'bar', 'bar');

       $this->assertEquals('ok', $client->getRequestToken());
    }
}
```

SensioLabs

Ou encore

```
class myTest extends PHPUnit_Framework_TestCase {
   /** @expectedException LogicException */
   public function testGetRequestTokenWithError() {
       $oauth   = $this->getMock('OAuth');
       $oauth
           ->expects($this->once())
           ->method('getRequestToken')
           ->will($this->returnValue(false));

       $client = new OAuthClient('foo',
           $oauth,
           'bar', 'bar', 'bar', 'bar', 'bar');

       $client->getRequestToken();
    }
}
```

SensioLabs

Conclusion

- Bonnes pratiques de développement
  - Design by contract
  - Separation of concerns
- Les mocks et les stubs
  - Si le code est suffisamment découplé
  - Si on ne doit pas en créer 50
- Les tests permettent de voir si le code est bon
  - Tests simples = Code simple

SensioLabs

Questions?

Merci de vous ~~indigner~~ exprimer

http://joind.in/4348

MERCI !

Marc Weistroff

Alexandre Salomé

Sensio Labs

<prenom>.<nom>@gmail.com

@futurecat - @alexandresalome

SensioLabs