Formation PHPUnit et Tests sous Symfony

Romain Bohdanowicz
Twitter : @bioub
http://www.formation.tech/

Sommaire

- Introduction
- PHPUnit
- Assertions
- Types de tests
- Doubles
- Symfony

formation.tech

Introduction

Présentation

- ## Romain Bohdanowicz
  Ingénieur EFREI 2008, spécialité en Ingénierie Logicielle

- ## Expérience
  Formateur/Développeur Freelance depuis 2006
  Plus de 5000 heures de formation

- ## Langages
  Expert : HTML / CSS / JavaScript / PHP / Java
  Notions : C / C++ / Objective-C / C# / Python / Bash / Batch

- ## Certifications
  PHP 5 / PHP 5.3 / PHP 5.5 / Zend Framework 1

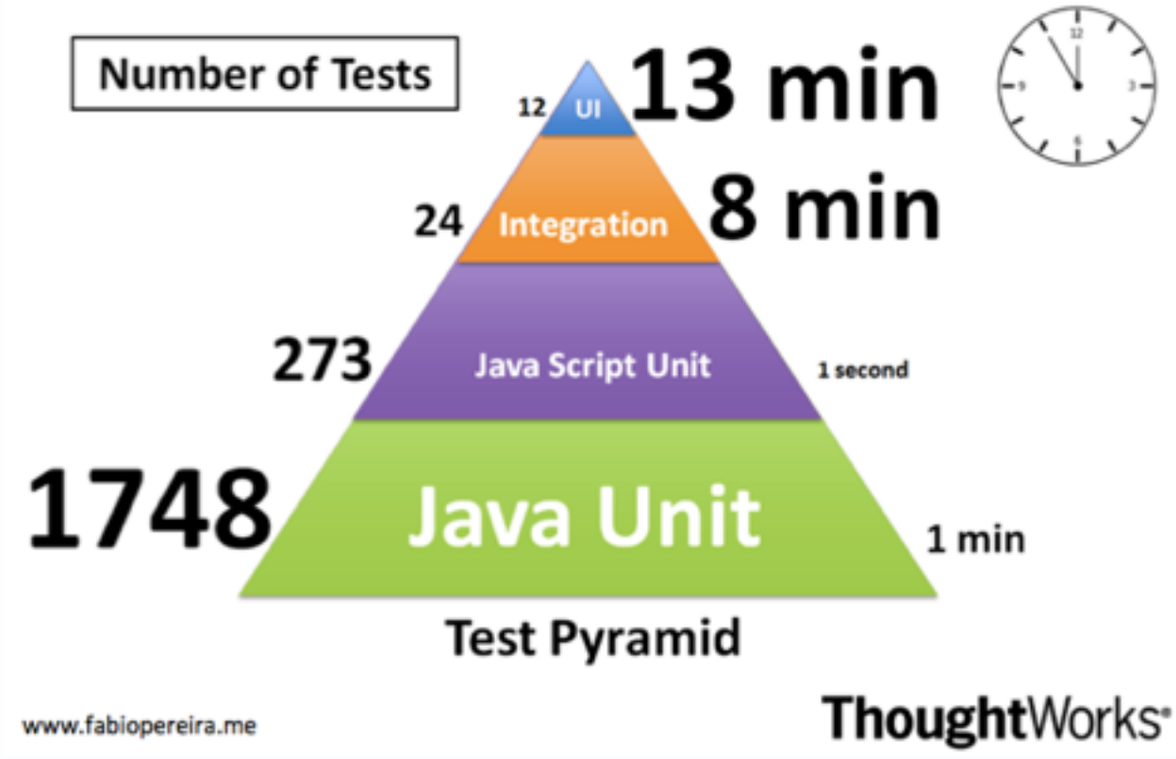
- ## Et vous ?
  Langages ? Expérience ? Utilité de cette formation ?

Tests automatisés

- Vérification manuelle
  - Ecrire une recette de tests et demander à une personne de la rejouer à des étapes clés (nouvelle version)
  - Ecrire le test sous la forme de code, et vérifier visuellement que les résultats attendus soit les bons
- Tests automatisés
  - Le test est codé, la vérification se fait dans un rapport
- Historique
  - sUnit en 1994 (SmallTalk), JUnit en 1997 (Java)
  - Les frameworks s’inspirant de jUnit sont catégorisés xUnit (PHPUnit, CUnit…)

Pyramide des Tests

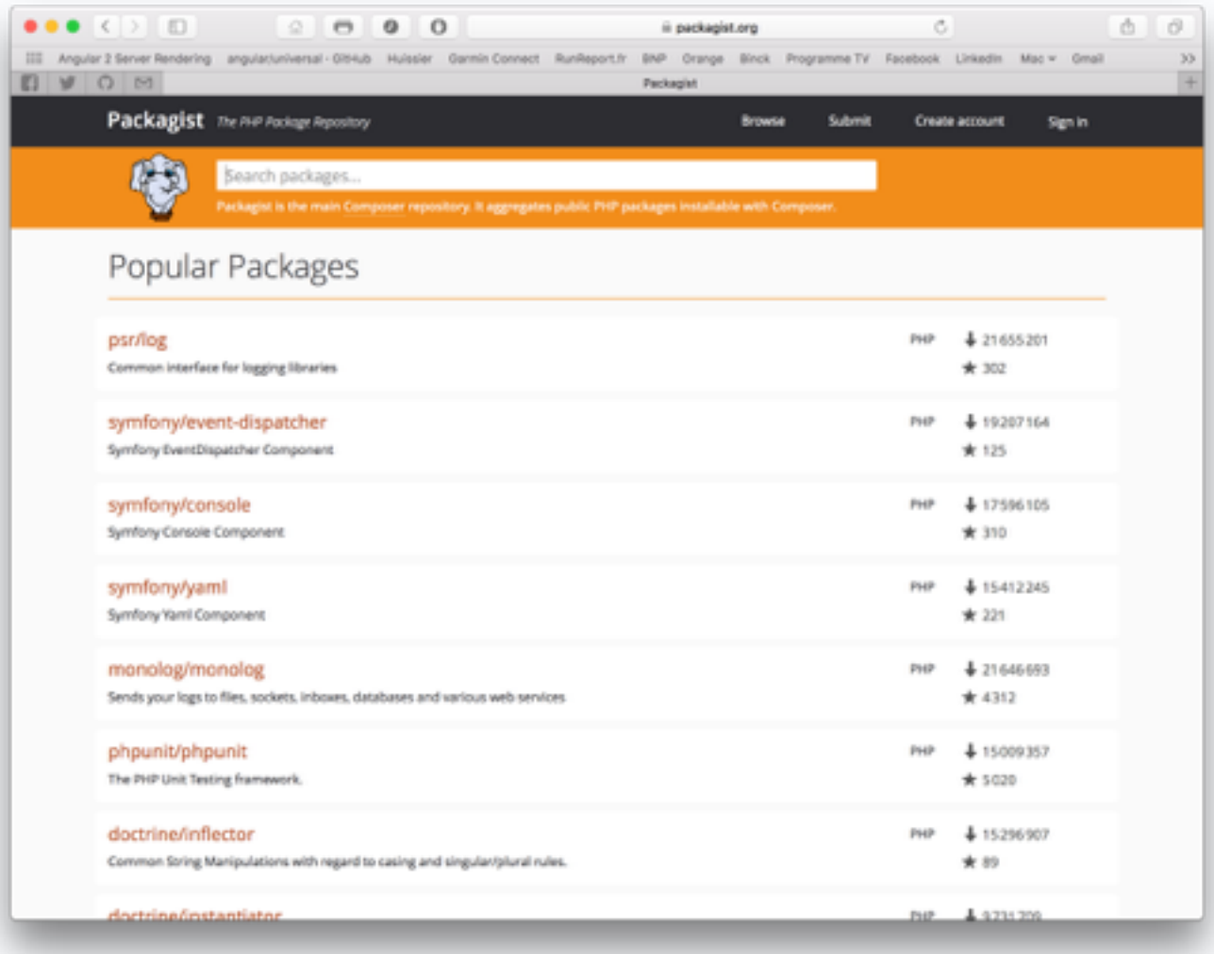
- Types de tests
  - **Unitaire** : tests des méthodes d'une classe
  - **Intégration** : teste l'intégration entre plusieurs classes
  - **Fonctionnels** : teste l'application du point de vue du client (HTTP dans le cas du web)
  - **End-to-End (E2E)** : teste l'application dans le client (y compris JavaScript, CSS…)



PHPUnit

PHPUnit - Introduction

- Créé en 2001 par Sebastian Bergmann
- Framework de tests de référence en PHP
  Utilisé, même étendu par Symfony et Zend Framework
- Documentation
  https://phpunit.de/documentation.html
- Open Source
  Licence BSD Modifiée
- Concurrents :
  atoum (FR), Behat (BDD), SimpleTest



PHPUnit - Installation globale

- PHAR
  - Dernière Version
    https://phar.phpunit.de/phpunit.phar
  - Version spécifique
    https://phar.phpunit.de/phpunit-X.Y.Z.phar

- Composer
  - Dernière Version
    `composer global require phpunit/phpunit`
  - Version spécifique
    `composer global require phpunit/phpunit:5.0.*`
  - Penser à ajouter le répertoire bin global au PATH, sur UNIX :
    `~/.composer/vendor/bin`

PHPUnit - Installation locale

- Composer
  - Dernière Version
    `composer require phpunit/phpunit --dev`
  - Version spécifique
    `composer require phpunit/phpunit:5.0.* --dev`
  - Ou en éditant directement le fichier composer.json puis composer update

```
{
    "require-dev" : {
        "phpunit/phpunit": "5.1.*"
    }
}
```

  - Exécution depuis la racine du projet :
    `./vendor/bin/phpunit`

PHPUnit - Structure d’un test

- Conventions
  - Un test PHPUnit est une méthode dont le nom commence par test : testMaFonction()
  - Cette méthode se trouve dans une classe dont le nom se termine par Test et qui hérite de \PHPUnit_Framework_TestCase ou \PHPUnit\Framework\TestCase
- Bonnes pratiques
  - Ne pas hésiter à être le plus verbeux possible dans le nom des méthodes
  - L’arborescence du répertoire test correspond au répertoire src (ex : src/MonNamespace/MaClasse.php -> tests/MonNamespaceTest/MaClasseTest.php)

PHPUnit - Exemple

```php
<?php

namespace FormationTechTest\Entity;

use FormationTech\Entity\CompteBancaire;

class CompteBancaireTest extends \PHPUnit_Framework_TestCase
{
    public function testCrediter()
    {
        $compte = new CompteBancaire(0);
        $compte->crediter(1000);
        $this->assertEquals(1000, $compte->getSolde());

        $compte->crediter(500);
        $this->assertEquals(1500, $compte->getSolde());
    }
}
```

```
MBP-de-Romain:PrepaFormationPHPUnit romain$ ./vendor/bin/phpunit tests/Entity/CompteBancaireTest.php --colors
PHPUnit 5.1.3 by Sebastian Bergmann and contributors.

.                                                                   1 / 1 (100%)

Time: 39 ms, Memory: 1.50Mb

OK (1 test, 2 assertions)
```

PHPUnit - Appels automatiques

- PHPUnit peut appeler des méthodes avant et après chaque test
  - setUp
  - tearDown
- Avant ou après chaque classe (méthodes statiques)
  - setUpBeforeClass
  - tearDownAfterClass

PHPUnit - Ligne de commande

```
MBP-de-Romain:PrepaFormationPHPUnit romain$ ./vendor/bin/phpunit -h
PHPUnit 5.1.3 by Sebastian Bergmann and contributors.

Usage: phpunit [options] UnitTest [UnitTest.php]
       phpunit [options] <directory>

Code Coverage Options:

  --coverage-clover <file>  Generate code coverage report in Clover XML format.
  --coverage-crap4j <file>  Generate code coverage report in Crap4J XML format.
  --coverage-html <dir>     Generate code coverage report in HTML format.
  --coverage-php <file>     Export PHP_CodeCoverage object to file.
  --coverage-text=<file>    Generate code coverage report in text format.
                            Default: Standard output.
  --coverage-xml <dir>      Generate code coverage report in PHPUnit XML format.
  --whitelist <dir>         Whitelist <dir> for code coverage analysis.

Logging Options:

  --log-junit <file>        Log test execution in JUnit XML format to file.
  --log-tap <file>          Log test execution in TAP format to file.
  --log-teamcity <file>     Log test execution in TeamCity format to file.
  --log-json <file>         Log test execution in JSON format.
  --testdox-html <file>     Write agile documentation in HTML format to file.
  --testdox-text <file>     Write agile documentation in Text format to file.
  --reverse-list            Print defects in reverse order
```

PHPUnit - Ligne de commande

```
Test Selection Options:

  --filter <pattern>         Filter which tests to run.
  --testsuite <pattern>      Filter which testsuite to run.
  --group ...                Only runs tests from the specified group(s).
  --exclude-group ...        Exclude tests from the specified group(s).
  --list-groups              List available test groups.
  --test-suffix ...          Only search for test in files with specified
                             suffix(es). Default: Test.php,.phpt

Configuration Options:

  --bootstrap <file>         A "bootstrap" PHP file that is run before the tests.
  -c|--configuration <file> Read configuration from XML file.
  --no-configuration         Ignore default configuration file (phpunit.xml).
  --no-coverage              Ignore code coverage configuration.
  --include-path <path(s)>  Prepend PHP's include_path with given path(s).
  -d key[=value]             Sets a php.ini value.

Miscellaneous Options:

  -h|--help                  Prints this usage information.
  --version                  Prints the version and exits.
  --atleast-version <min>   Checks that version is greater than min and exits.
```

PHPUnit - Ligne de commande

```
Test Execution Options:

  --report-useless-tests    Be strict about tests that do not test anything.
  --strict-coverage         Be strict about unintentionally covered code.
  --strict-global-state     Be strict about changes to global state
  --disallow-test-output    Be strict about output during tests.
  --disallow-resource-usage Be strict about resource usage during small tests.
  --enforce-time-limit      Enforce time limit based on test size.
  --disallow-todo-tests     Disallow @todo-annotated tests.

  --process-isolation       Run each test in a separate PHP process.
  --no-globals-backup       Do not backup and restore $GLOBALS for each test.
  --static-backup           Backup and restore static attributes for each test.

  --colors=<flag>           Use colors in output ("never", "auto" or "always").
  --columns <n>             Number of columns to use for progress output.
  --columns max             Use maximum number of columns for progress output.
  --stderr                  Write to STDERR instead of STDOUT.
  --stop-on-error           Stop execution upon first error.
  --stop-on-failure         Stop execution upon first error or failure.
  --stop-on-warning         Stop execution upon first warning.
  --stop-on-risky           Stop execution upon first risky test.
  --stop-on-skipped         Stop execution upon first skipped test.
  --stop-on-incomplete      Stop execution upon first incomplete test.
  -v|--verbose              Output more verbose information.
  --debug                   Display debugging information during test execution.

  --loader <loader>         TestSuiteLoader implementation to use.
  --repeat <times>          Runs the test(s) repeatedly.
  --tap                     Report test execution progress in TAP format.
  --teamcity                Report test execution progress in TeamCity format.
  --testdox                 Report test execution progress in TestDox format.
  --printer <printer>       TestListener implementation to use.
```

PHPUnit - phpunit.xml

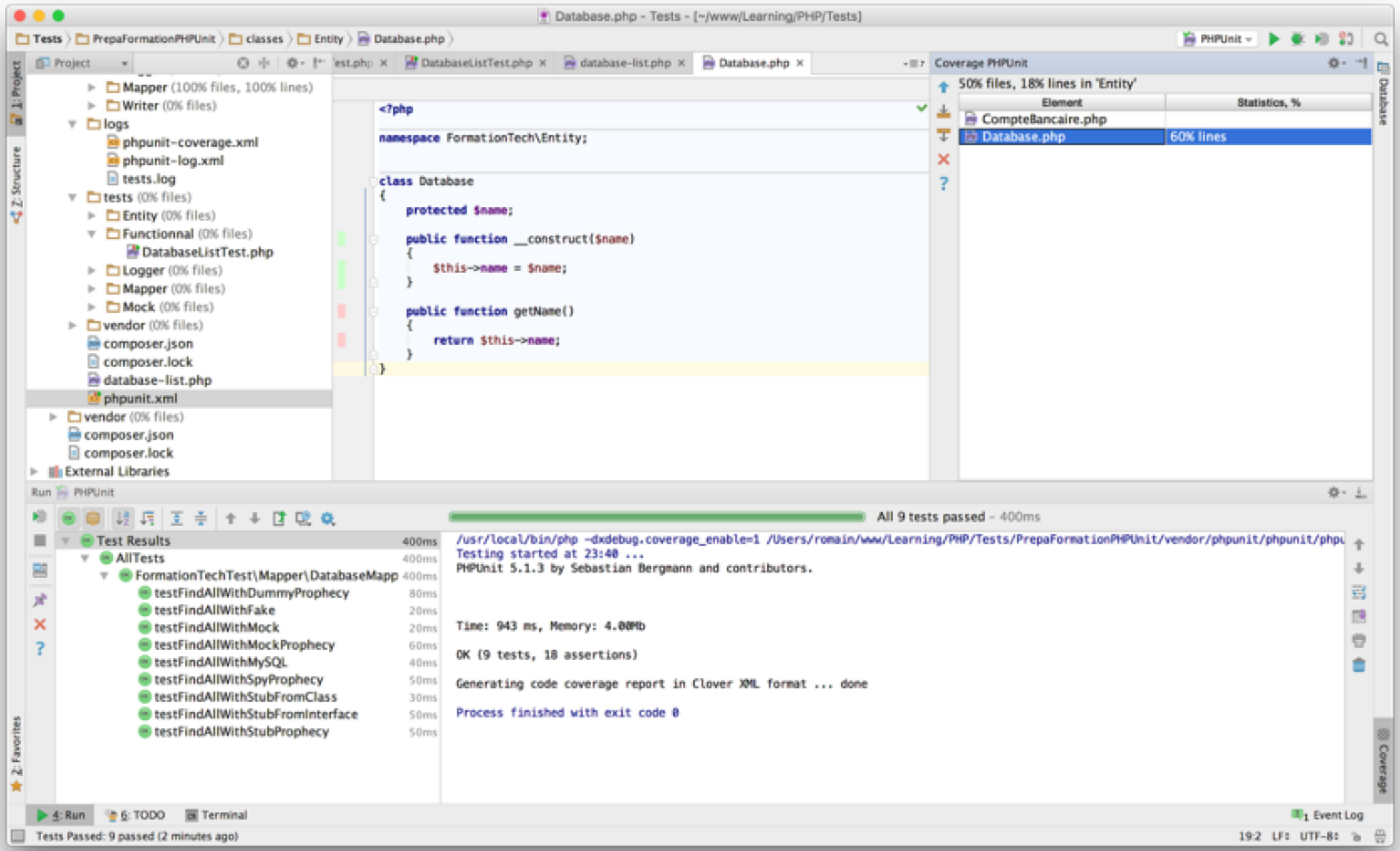
```xml
<?xml version="1.0" encoding="UTF-8"?>
<phpunit colors="true">

    <testsuites>
        <testsuite name="AllTests">
            <directory>tests/Mapper</directory>
        </testsuite>
    </testsuites>

    <filter>
        <blacklist>
            <directory suffix=".php"></directory>
            <file></file>
            <exclude>
                <directory suffix=".php"></directory>
                <file></file>
            </exclude>
        </blacklist>
        <whitelist processUncoveredFilesFromWhitelist="true">
            <directory suffix=".php">classes</directory>
            <file></file>
            <exclude>
                <directory suffix=".php"></directory>
                <file></file>
            </exclude>
        </whitelist>
    </filter>

    <logging>
        <log type="coverage-clover" target="logs/phpunit-coverage.xml"/>
        <log type="junit" target="logs/phpunit-log.xml" logIncompleteSkipped="false"/>
    </logging>

</phpunit>
```

PHPUnit - bootstrap

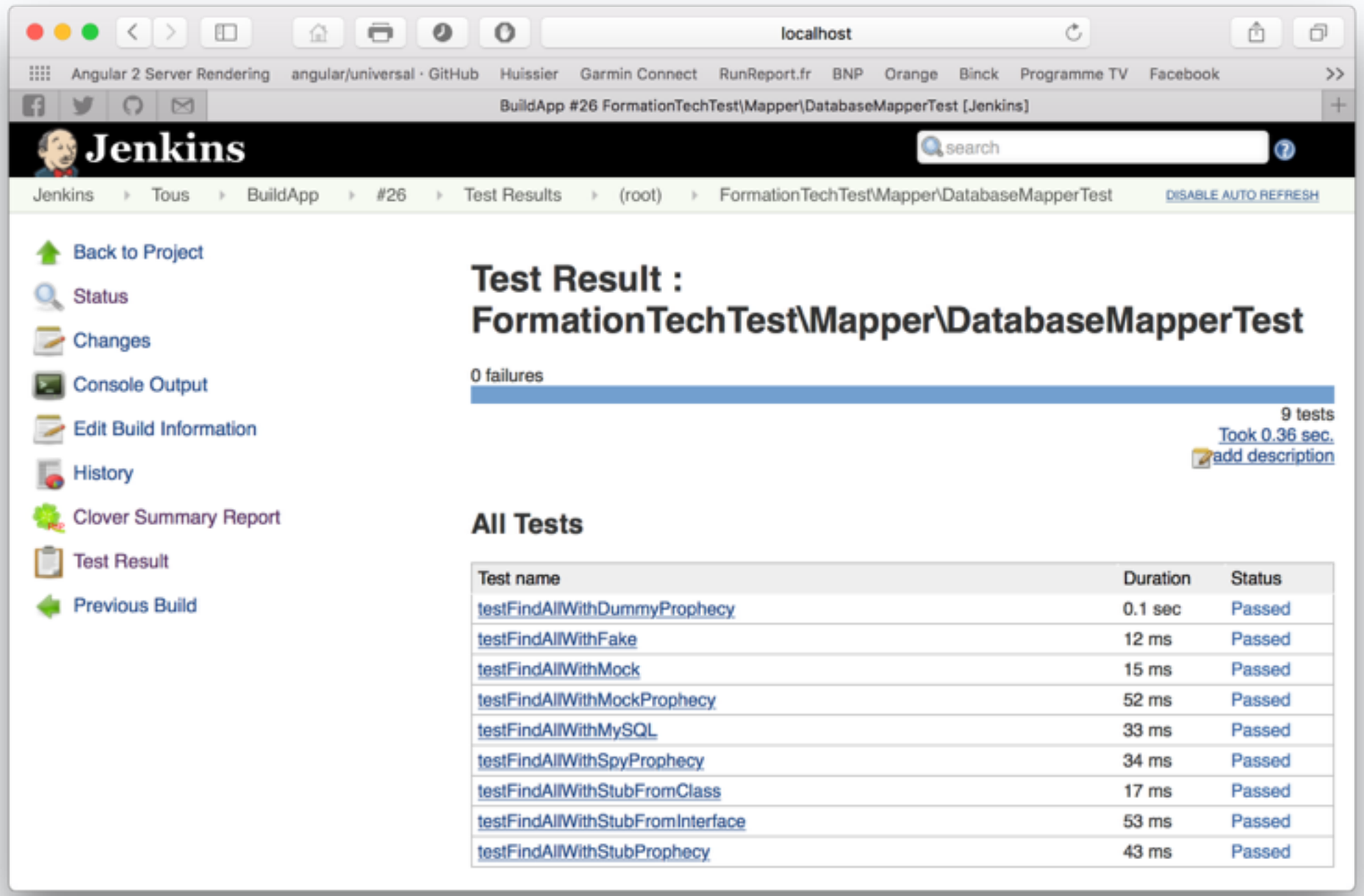
- Un fichier de bootstrap peut être exécuté au démarrage de PHPUnit
- Intérêts :
  - Autochargement de classe (sauf si phpunit a été installé avec Composer et que l'autoloader est celui de composer)
  - Modification du include_path
  - Chargement de fichiers de configuration

PHPUnit - IDE

- PHPStorm



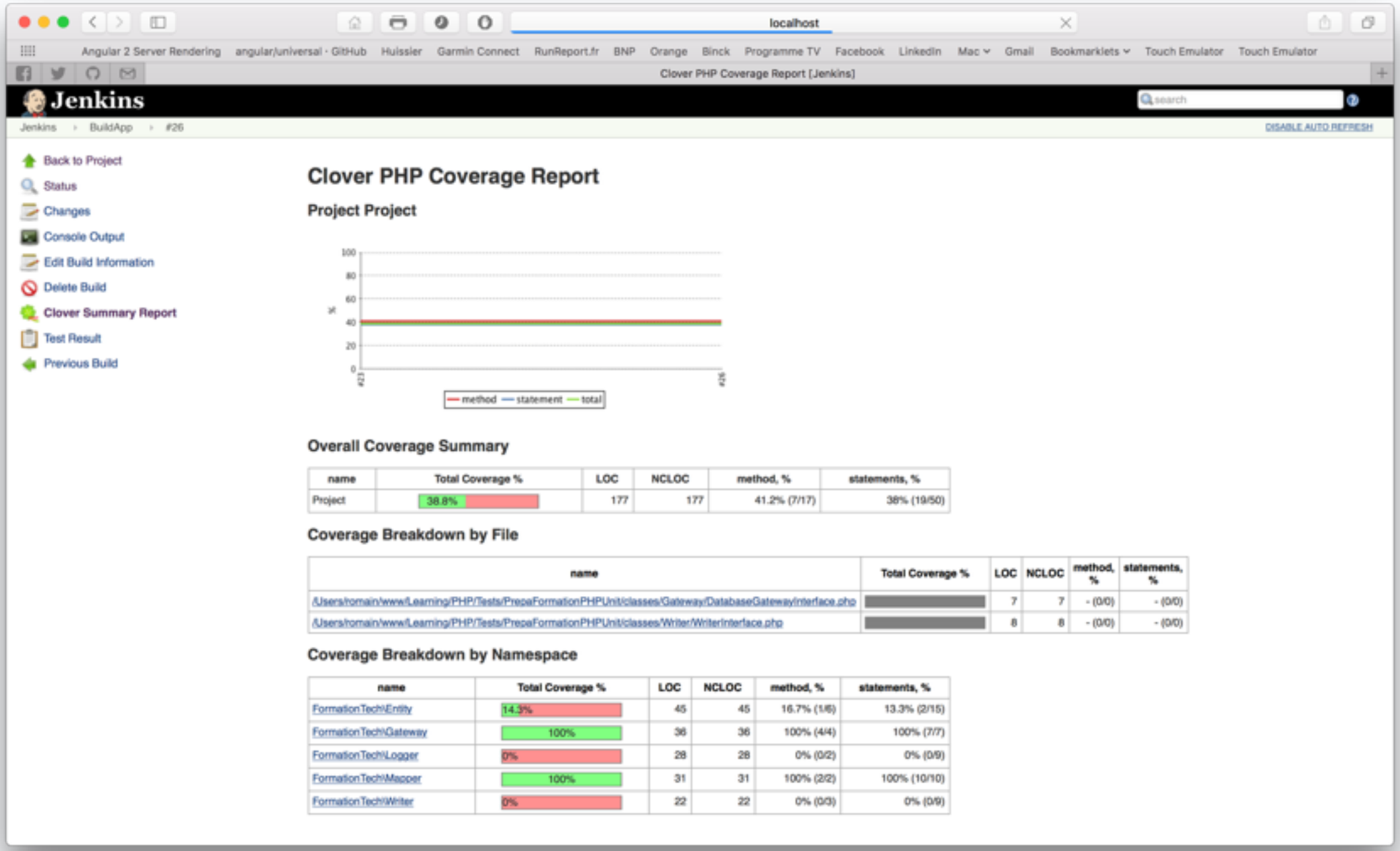
PHPUnit - Intégration continue



- JUnit Plugin

PHPUnit - Intégration continue

- Clover PHP plugin



formation.tech

Assertions

Assertions - Introduction

- Dans un framework xUnit, les assertions sont les méthodes qui vérifient qu'un résultat espéré corresponde au résultat attendu
- Le test échoue et s'arrête à la première assertion qui n'est pas vérifiée
- Bonnes pratiques :
  - Plusieurs assertions par test
  - Utiliser la méthode d'assertion la plus précise possible pour avoir un message d'erreur clair :
    Ex : `assertEmpty($tableau)`
    plutôt que `assertEquals(0, count($tableau))`
  - Si possible ajouter un message personnalisé

Assertions - Basiques

- assertContains
- assertEquals
- assertFalse
- assertGreaterThan
- assertGreaterThanOrEqual
- assertInfinite
- assertInternalType
- assertLessThan
- assertLessThanOrEqual
- assertNan
- assertRegExp
- assertSame
- assertStringEndsWith
- assertStringMatchesFormat
- assertStringStartsWith
- assertThat
- assertTrue

Assertions - Tableaux

- assertArrayHasKey
- assertArraySubset
- assertCount
- assertContains
- assertContainsOnly
- assertContainsOnlyInstancesOf
- assertEmpty

Assertions - Fichiers et Formats

- Fichiers
  - assertFileEquals
  - assertFileExists
  - assertStringEqualsFile
  - assertStringMatchesFormatFile
- XML
  - assertEqualXMLStructure
  - assertXmlFileEqualsXmlFile
  - assertXmlStringEqualsXmlFile
  - assertXmlStringEqualsXmlString
- JSON
  - assertJsonFileEqualsJsonFile
  - assertJsonStringEqualsJsonFile
  - assertJsonStringEqualsJsonString

Assertions - Classes et Objets

- assertClassHasAttribute
- assertClassHasStaticAttribute
- assertInstanceOf
- assertObjectHasAttribute
- assertNull

formation.tech

Types de tests

Types de tests - Test Unitaire

```
PrepaFormationPHPUnit
  classes
    Entity
      CompteBancaire.php
  tests
    Entity
      CompteBancaireTest.php
  vendor
  composer.json
  composer.lock
```

```php
<?php

namespace FormationTech\Entity;

class CompteBancaire
{
    protected $solde;

    public function __construct($solde = 0)
    {
        $this->solde = (double) $solde;
    }
    public function getSolde()
    {
        return $this->solde;
    }

    public function debiter($montant)
    {
        $this->solde -= (double) $montant;
    }

    public function crediter($montant)
    {
        $this->solde += (double) $montant;
    }
}
```

```php
<?php

namespace FormationTechTest\Entity;

use FormationTech\Entity\CompteBancaire;

class CompteBancaireTest extends
\PHPUnit_Framework_TestCase
{
    public function testCrediter()
    {
        $compte = new CompteBancaire(0);
        $compte->crediter(1000);
        $this->assertEquals(1000, $compte->getSolde());

        $compte->crediter(500);
        $this->assertEquals(1500, $compte->getSolde());
    }
}
```

```
MBP-de-Romain:PrepaFormationPHPUnit romain$ ./vendor/bin/phpunit tests/Entity/CompteBancaireTest.php --colors
PHPUnit 5.1.3 by Sebastian Bergmann and contributors.

.                                                                   1 / 1 (100%)

Time: 39 ms, Memory: 1.50Mb

OK (1 test, 2 assertions)
```

Types de tests - Test d’intégration

```php
<?php

namespace FormationTech\Logger;

use FormationTech\Writer\WriterInterface;
use Psr\Log\LoggerInterface;
use Psr\Log\LoggerTrait;

class Logger implements LoggerInterface
{
    use LoggerTrait;

    protected $writer;

    public function __construct(WriterInterface $writer)
    {
        $this->writer = $writer;
    }

    public function log($level, $message, array $context = array())
    {
        $datetime = date('Y-m-d H:i:s');
        $logMessage = "[$level] - $datetime - $message";

        $this->writer->write($logMessage);
    }
}
```

```php
<?php

namespace FormationTech\Writer;

class FileWriter implements WriterInterface
{
    protected $fic;

    public function __construct($filePath)
    {
        $this->fic = fopen($filePath, 'a');
    }

    public function write($message)
    {
        fwrite($this->fic, "$message\n");
    }

    public function __destruct()
    {
        fclose($this->fic);
    }
}
```

- Exemple de communication entre 2 classes :
  - Logger dépend de Writer (WriterInterface) et est compatible PSR-4
  - FileWriter implémente WriterInterface et sa méthode write

Types de tests - Test d'intégration

```php
<?php

namespace FormationTechTest\Logger;

use FormationTech\Logger\Logger;
use FormationTech\Writer\FileWriter;
use Psr\Log\LogLevel;

class LoggerTest extends \PHPUnit_Framework_TestCase
{
    public function testLogWithFileWriter()
    {
        $testFile = __DIR__ . '/../../tests.log';
        $fw = new FileWriter($testFile);
        $logger = new Logger($fw);

        $logger->log(LogLevel::NOTICE, 'Un message');
        $content = file_get_contents($testFile);

        $this->assertRegExp('/\[notice\] - \d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2} - Un message\n/',
$content);
    }
}
```

```
MBP-de-Romain:PrepaFormationPHPUnit romain$ ./vendor/bin/phpunit tests/Logger/LoggerTest.php --colors
PHPUnit 5.1.3 by Sebastian Bergmann and contributors.

.                                                                   1 / 1 (100%)

Time: 38 ms, Memory: 1.50Mb

OK (1 test, 1 assertion)
```

Types de tests - Test fonctionnel

```php
<?php
require_once __DIR__ . '/vendor/autoload.php';

$pdo = new \PDO('mysql:host=localhost', 'root', '');
$gateway = new \FormationTech\Gateway\DatabaseGateway($pdo);
$dbList = $gateway->listDbs();
?>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Database list</title>
    </head>
    <body>
        <h2>Database list</h2>
        <ul>
            <?php foreach ($dbList as $db) : ?>
            <li><?=htmlspecialchars($db)?></li>
            <?php endforeach; ?>
        </ul>
    </body>
</html>
```

- Démarrage du PHP Built-in Server

  `php -S localhost:8080`

Types de tests - Test fonctionnel

```php
<?php

namespace FormationTechTest\Functionnal;

use Goutte\Client;

class DatabaseListTest extends \PHPUnit_Framework_TestCase
{
    public function testListDbs()
    {
        $client = new Client();
        $crawler = $client->request('GET', 'http://localhost:8080/database-list.php');

        $this->assertEquals(200, $client->getResponse()->getStatus());
        $this->assertEquals('Database list', $crawler->filter('h2')->text());
        $this->assertCount(13, $crawler->filter('ul > li'));
    }
}
```

```
MBP-de-Romain:PrepaFormationPHPUnit romain$ ./vendor/bin/phpunit tests/Logger/LoggerTest.php --colors
PHPUnit 5.1.3 by Sebastian Bergmann and contributors.

.                                                                   1 / 1 (100%)

Time: 38 ms, Memory: 1.50Mb

OK (1 test, 1 assertion)
```

formation.tech

Doubles

Double - Introduction

- Le code PHP fait souvent appel à des composants externes :
  - Accès aux entrées/sorties
  - Accès à une base de données
  - Accès à un Service Web
- Certaines classes ne peuvent être testées de manières unitaires car elles dépendent d’autres classes.
- Solutions : les Doubles

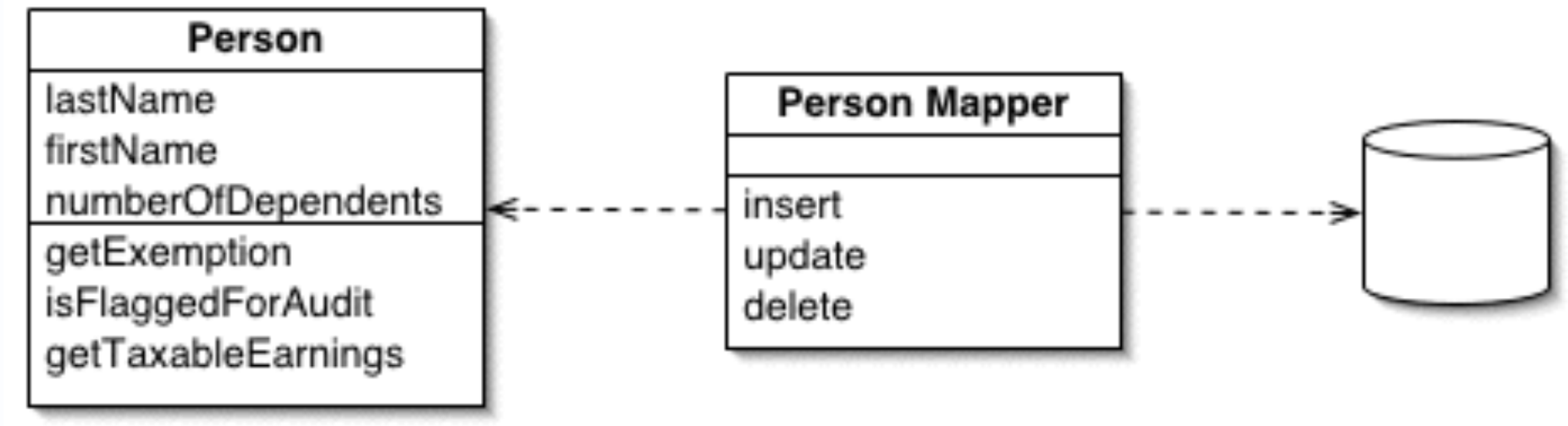
  Objets ou fonctions qui ressemblent et se comportent comme le composant qu’ils imitent, mais qui sont en réalité des versions simplifiée qui permettent de faciliter l’écriture du test.

Double - Introduction

- 5 types de Doubles :
  - Fake (une classe créée par l'utilisateur qui fera les opérations en mémoire)
  - Dummy (une classe générée dont les méthodes ne font rien)
  - Stub (une classe générée dont les méthodes ont le même comportement)
  - Mock (Stub + vérification que les méthodes soient bien appelée)
  - Spy (Dummy + vérification que les méthodes soient bien appelée à postériori)
- Bonnes pratiques :
  - Injection de Dépendance (pas de composition)
  - Registre ou Container d'injection de Dépendance

Double - Classes d’exemple

- Un DataMapper
  http://martinfowler.com/eaaCatalog/dataMapper.html



Double - Classes d’exemple

- Entité

```php
<?php

namespace FormationTech\Entity;

class Database
{
    protected $name;

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

    public function getName()
    {
        return $this->name;
    }
}
```

Double - Classes d’exemple

- Gateway

```php
<?php

namespace FormationTech\Gateway;

class DatabaseGateway implements DatabaseGatewayInterface
{
    protected $pdo;

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

    public function listDbs()
    {
        $stmt = $this->pdo->query('SHOW DATABASES');

        return $stmt->fetchAll(\PDO::FETCH_COLUMN);
    }
}
```

```php
<?php

namespace FormationTech\Gateway;

interface DatabaseGatewayInterface
{
    public function listDbs();
}
```

Double - Classes d’exemple

- Mapper (classe à tester unitairement)

```php
<?php

namespace FormationTech\Mapper;

use FormationTech\Entity\Database;
use FormationTech\Gateway\DatabaseGatewayInterface;

class DatabaseMapper
{
    protected $gateway;

    public function __construct(DatabaseGatewayInterface $gateway)
    {
        $this->gateway = $gateway;
    }

    public function findAll()
    {
        $dbsArray = $this->gateway->listDbs();
        $dbsObj = [];

        if (!$dbsArray) {
            return $dbsObj;
        }

        foreach ($dbsArray as $dbName) {
            $dbsObj[] = new Database($dbName);
        }

        return $dbsObj;
    }
}
```

Double - Sans Double

- Test sans double

```php
<?php

namespace FormationTechTest\Mapper;

use FormationTech\Entity\Database;
use FormationTech\Gateway\DatabaseGateway;
use FormationTech\Mapper\DatabaseMapper;

class DatabaseMapperTest extends \PHPUnit_Framework_TestCase
{
   public function testFindAllWithMySQL()
    {
        $pdo = new \PDO('mysql:host=localhost', 'root', '');
        $gateway = new DatabaseGateway($pdo);
        $mapper = new DatabaseMapper($gateway);

        $dbs = $mapper->findAll();

        $this->assertCount(13, $dbs);
        $this->assertContainsOnlyInstancesOf(Database::class, $dbs);
    }
}
```

- Problème : changement dans la base de données ?
- Solution : fixture dans un setUp ? double ?

Double - Fake

- Fake

```php
<?php

namespace FormationTech\Gateway;

class DatabaseGatewayFake implements DatabaseGatewayInterface
{
    protected $dbs;

    public function __construct(Array $dbs)
    {
        $this->dbs = $dbs;
    }

    public function listDbs()
    {
        return $this->dbs;
    }
}
```

```php
<?php

namespace FormationTechTest\Mapper;

use FormationTech\Entity\Database;
use FormationTech\Gateway\DatabaseGatewayFake;
use FormationTech\Mapper\DatabaseMapper;

class DatabaseMapperTest extends \PHPUnit_Framework_TestCase
{
    public function testFindAllWithFake()
    {
        $gateway = new DatabaseGatewayFake(['db1', 'db2', 'db3']);
        $mapper = new DatabaseMapper($gateway);

        $dbs = $mapper->findAll();

        $this->assertCount(3, $dbs);
        $this->assertContainsOnlyInstancesOf(Database::class, $dbs);
    }
}
```

Double - Prophecy

- Sebastian Bergman à propos de l'API de Mock de PHPUnit : https://thephp.cc/news/2015/02/phpunit-4-5-and-prophecy
- L'ancien API continue d'exister pour rester compatible avec les anciens tests
- PHPUnit depuis la version 4.5 intègre un framework de test moderne : Prophecy
- Documentation https://github.com/phpspec/prophecy

Double - Prophecy Dummy

```php
<?php

namespace FormationTechTest\Mapper;

use FormationTech\Entity\Database;
use FormationTech\Gateway\DatabaseGateway;
use FormationTech\Mapper\DatabaseMapper;

class DatabaseMapperTest extends \PHPUnit_Framework_TestCase
{

    // ...

    public function testFindAllWithDummyProphecy()
    {
        $dummy = $this->prophesize(DatabaseGateway::class);

        $mapper = new DatabaseMapper($dummy->reveal());

        $dbs = $mapper->findAll();

        $this->assertEmpty($dbs);
    }

    // ...

}
```

Double - Prophecy Stub

```php
<?php

namespace FormationTechTest\Mapper;

use FormationTech\Entity\Database;
use FormationTech\Gateway\DatabaseGateway;
use FormationTech\Mapper\DatabaseMapper;

class DatabaseMapperTest extends \PHPUnit_Framework_TestCase
{

    // ...

    public function testFindAllWithStubProphecy()
    {
        $stub = $this->prophesize(DatabaseGateway::class);

        $stub->listDbs()->willReturn(['db1', 'db2', 'db3', 'db4']);

        $mapper = new DatabaseMapper($stub->reveal());

        $dbs = $mapper->findAll();

        $this->assertCount(4, $dbs);
        $this->assertContainsOnlyInstancesOf(Database::class, $dbs);
    }

    // ...

}
```

Double - Prophecy Mock

```php
<?php

namespace FormationTechTest\Mapper;

use FormationTech\Entity\Database;
use FormationTech\Gateway\DatabaseGateway;
use FormationTech\Mapper\DatabaseMapper;

class DatabaseMapperTest extends \PHPUnit_Framework_TestCase
{

    // ...

    public function testFindAllWithMockProphecy()
    {
        $mock = $this->prophesize(DatabaseGateway::class);

        $mock->listDbs()->willReturn(['db1', 'db2'])->shouldBeCalledTimes(1);

        $mapper = new DatabaseMapper($mock->reveal());

        $dbs = $mapper->findAll();

        $this->assertCount(2, $dbs);
        $this->assertContainsOnlyInstancesOf(Database::class, $dbs);
    }

    // ...

}
```

Double - Prophecy Spy

```php
<?php

namespace FormationTechTest\Mapper;


use FormationTech\Entity\Database;
use FormationTech\Gateway\DatabaseGateway;
use FormationTech\Mapper\DatabaseMapper;

class DatabaseMapperTest extends \PHPUnit_Framework_TestCase
{

    // ...

    public function testFindAllWithSpyProphecy()
    {
        $mock = $this->prophesize(DatabaseGateway::class);

        $mapper = new DatabaseMapper($mock->reveal());

        $dbs = $mapper->findAll();

        $this->assertEmpty($dbs);

        $mock->listDbs()->shouldHaveBeenCalledTimes(1);
    }

    // ...

}
```

Double - Autres frameworks

- Mockery
  https://github.com/padraic/mockery
  http://docs.mockery.io/en/latest/

- Phake
  https://github.com/mlively/Phake
  http://phake.readthedocs.org/en/2.1/

Symfony

Composition

Une composition est un type d'association forte entre 2 objet. La destruction d'un objet entrainerait la destruction de l'objet associé.
Exemple : Un objet Tasse est composée de Café

```php
<?php

namespace ExpressoComposition;

class Cafe
{
    protected $variete;
    protected $provenance;

    public function __construct($provenance, $variete)
    {
        $this->provenance = $provenance;
        $this->variete = $variete;
    }
}
```

```php
<?php

namespace ExpressoComposition;

class Tasse
{
    protected $contenu;

    public function __construct() {
        $this->contenu = new Cafe("Arabica", "Mexique");
    }
}
```

```php
<?php

require_once 'autoload.php';

$tasseDeCafe = new \ExpressoComposition\Tasse();
```

Mauvaise Pratique

La composition est désormais considéré comme une mauvaise pratique. Premièrement la classe Tasse n'est très réutilisable, elle ne peut contenir que du café. De plus il n'est pas possible d'écrire d'écrire un test unitaire de Tasse, puisqu'il faudrait en même temps tester Café.
Globalement il faut essayer de proscrire l'utilisation de new il l'intérieur d'une classe (à l'exception des Values Objects (DateTime, ArrayObject, etc…)

Solution

La solution est simple, pour éviter le new dans cette classe nous allons injecter la dépendance.

```php
<?php

namespace ExpressoInjection;

interface Liquide {

}
```

```php
<?php

namespace ExpressoInjection;

class Cafe implements Liquide
{
    protected $variete;
    protected $provenance;

    public function __construct($provenance, $variete)
    {
        $this->provenance = $provenance;
        $this->variete = $variete;
    }
}
```

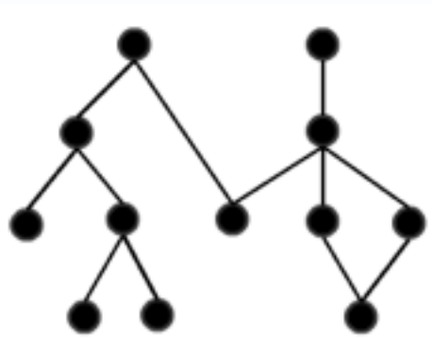
```php
<?php

namespace ExpressoInjection;

class Tasse
{
    protected $contenu;

    public function __construct(Liquide $contenu) {
        $this->contenu = $contenu;
    }
}
```

```php
<?php

require_once 'autoload.php';

$cafe = new \ExpressoInjection\Cafe();
$tasseDeCafe = new \ExpressoInjection\Tasse($cafe);
```

- La classe Tasse peut désormais recevoir n’importe quel contenu qui implémente l’interface Liquide.

- ## Conteneur d’injection de dépendance (DIC)

  Le problème lorsqu’on injecte les dépendances est qu’on peut parfois se retrouver avec des dépendances complexes :



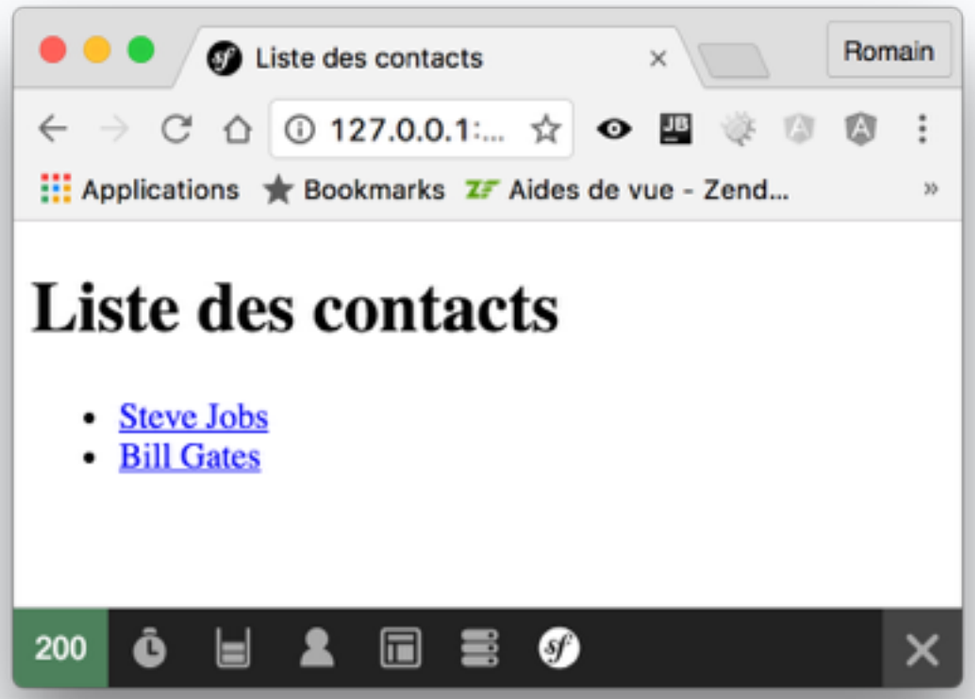
- Dans ce cas il devient utile d’utiliser un conteneur d’injection de dépendance qu’on aura configuré au préalable. En PHP il existe quelques DIC connus :
  - Pimple par Fabien Potencier
  - Dice par Tom Butler
  - PHP-DI par Matthieu Napoli
  - Symfony\Container intégré Symfony2
  - Zend\Di & Zend\ServiceManager intégrés ZF 2

- ## Test fonctionnels

  Symfony\Bundle\FrameworkBundle contient 2 classes pour faciliter les tests

- Symfony\Bundle\FrameworkBundle\Test\WebTestCase (et sa méthode createClient pour les tests fonctionnels)
- Symfony\Bundle\FrameworkBundle\Test\KernelTestCase (et sa méthode bootKernel pour les tests qui nécessitent un kernel)

```php
public function listAction()
{
    $repo = $this->getDoctrine()
                 ->getRepository('AppBundle:Contact');

    $contacts = $repo->findAll();

    return $this->render('list.html.twig', array(
        'contacts' => $contacts
    ));
}
```



Symfony

- Test fonctionnels

```php
<?php

namespace AppBundle\Tests\Controller;

use Symfony\Bundle\FrameworkBundle\Test\WebTestCase;

class ContactControllerTest extends WebTestCase
{
    public function testListAvecMysql()
    {
        $client = static::createClient();

        $crawler = $client->request('GET', '/contacts/');
        $this->assertEquals(200, $client->getResponse()->getStatusCode());

        $this->assertContains('Liste des contacts', $crawler->filter('h1')->text());

        $this->assertCount(2, $crawler->filter('h1 + ul > li'));
    }
}
```

- Problème : le tests dépend d'un composant extérieur (base de données)
- Solution 1 : Réinitialiser la base de données entre chaque test (dans une méthode setup)
- Solution 2 : Utiliser les mocks

- Test fonctionnels (avec mock)

```php
public function testListAvecMock()
{
    $client = static::createClient();

    $contacts = [
        (new Contact())->setId(1)->setPrenom('A')->setNom('B'),
        (new Contact())->setId(2)->setPrenom('C')->setNom('D'),
        (new Contact())->setId(3)->setPrenom('E')->setNom('F'),
    ];

    $mockRepo = $this->prophesize(ContactRepository::class);
    $mockRepo->findAll()->willReturn($contacts)->shouldBeCalledTimes(1);

    $mockRegistry = $this->prophesize(Registry::class);
    $mockRegistry->getConnectionNames()->shouldBeCalledTimes(1);
    $mockRegistry->getManagerNames()->shouldBeCalledTimes(1);
    $mockRegistry->getRepository('AppBundle:Contact')->willReturn($mockRepo->reveal())->shouldBeCalledTimes(1);

    $client->getContainer()->set('doctrine', $mockRegistry->reveal());

    $crawler = $client->request('GET', '/contacts/');

    $this->assertCount(3, $crawler->filter('h1 + ul > li'));
}
```

- Problème : le contrôleur dépend de la classe Registry pour obtenir le Repository, 2 mocks à créer
- Solution : avoir une dépendance directe dans le Conteneur (couche Service par exemple)

- Test fonctionnels (avec couche Service)

```php
public function testListAvecMockEtServiceLayer()
{
    $client = static::createClient();

    $contacts = [
        (new Contact())->setId(1)->setPrenom('A')->setNom('B'),
        (new Contact())->setId(2)->setPrenom('C')->setNom('D'),
        (new Contact())->setId(3)->setPrenom('E')->setNom('F'),
    ];

    $mockRepo = $this->prophesize(ContactManager::class);
    $mockRepo->findAll()->willReturn($contacts)->shouldBeCalledTimes(1);

    $client->getContainer()->set('app.manager.contact', $mockRepo->reveal());

    $crawler = $client->request('GET', '/contacts/list-avec-manager');

    $this->assertCount(3, $crawler->filter('h1 + ul > li'));
}
```

```yaml
# services.yml
services:
    app.manager.contact:
          class: AppBundle\Manager\ContactManager
          arguments: ["@doctrine.orm.entity_manager"]
```

Test de Commande

```php
<?php

namespace AppBundle\Command;

use Symfony\Bundle\FrameworkBundle\Command\ContainerAwareCommand;
use Symfony\Component\Console\Input\InputArgument;
use Symfony\Component\Console\Input\InputInterface;
use Symfony\Component\Console\Input\InputOption;
use Symfony\Component\Console\Output\OutputInterface;

class HelloWorldCommand extends ContainerAwareCommand
{
    protected function configure()
    {
        $this->setName('hello:world')
             ->setDescription('A Hello command')
             ->addArgument('name', InputArgument::OPTIONAL, 'Your name')
             ->addOption('upper', 'u', InputOption::VALUE_NONE, 'Capitalize answer');
    }

    protected function execute(InputInterface $input, OutputInterface $output)
    {
        $name = $input->getArgument('name');
        $message = ($name) ? "Hello $name :)" : "Hello !";

        if ($input->getOption('upper')) {
            $message = strtoupper($message);
        }

        $output->writeln($message);
    }
}
```

Test de Commande

```php
<?php

namespace Tests\AppBundle\Command;

use AppBundle\Command\HelloWorldCommand;
use Symfony\Bundle\FrameworkBundle\Console\Application;
use Symfony\Bundle\FrameworkBundle\Test\KernelTestCase;
use Symfony\Component\Console\Tester\CommandTester;

class HelloWorldCommandTest extends KernelTestCase
{
    public function testExecute()
    {
        $kernel = $this->createKernel();
        $kernel->boot();

        $application = new Application($kernel);
        $application->add(new HelloWorldCommand());

        $command = $application->find('hello:world');
        $commandTester = new CommandTester($command);
        $exitCode = $commandTester->execute(array(
            'command'  => $command->getName(),
            'name' => 'Romain',
            '-u' => true
        ));

        $output = $commandTester->getDisplay();
        $this->assertEquals(0, $exitCode, 'Returns 0 in case of success');
        $this->assertContains('HELLO ROMAIN :)', $output);
    }
}
```

Test de Formulaire

```php
<?php

namespace AppBundle\Form;

use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;

class ContactType extends AbstractType
{
    /**
     * @param FormBuilderInterface $builder
     * @param array $options
     */
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('prenom')
            ->add('nom')
        ;
    }

    /**
     * @param OptionsResolver $resolver
     */
    public function configureOptions(OptionsResolver $resolver)
    {
        $resolver->setDefaults(array(
            'data_class' => 'AppBundle\Entity\Contact'
        ));
    }
}
```

Symfony

```php
<?php

namespace AppBundle\Tests\Form;

use AppBundle\Entity\Contact;
use AppBundle\Form\ContactType;
use Symfony\Component\Form\Test\TypeTestCase;

class ContactTypeTest extends TypeTestCase
{
    public function testSubmitValidData()
    {
        $formData = array(
            'prenom' => 'Romain',
            'nom' => 'Bohdanowicz',
        );

        $form = $this->factory->create(ContactType::class);

        $contact = (new Contact())->setPrenom('Romain')->setNom('Bohdanowicz');

        // submit the data to the form directly
        $form->submit($formData);

        $this->assertTrue($form->isSynchronized());
        $this->assertEquals($contact, $form->getData());

        $view = $form->createView();
        $children = $view->children;

        foreach (array_keys($formData) as $key) {
            $this->assertArrayHasKey($key, $children);
        }
    }
}
```