Tests sous Drupal 8

Drupal 8 et PHPUnit

L'ÉQUIPETECH

Fabien Clément L'ÉQUIPETECH

- Contributeur Drupal depuis + de 10 ans.
- Core contributeur Drupal 8.
- Core contributeur Drupal Commerce 1.x et 2.x.
- Contributeur de modules.
- Lead developer pendant 3 ans chez Commerce Guys.
- Directeur technique associé L'Équipe.tech

GozOo Goz

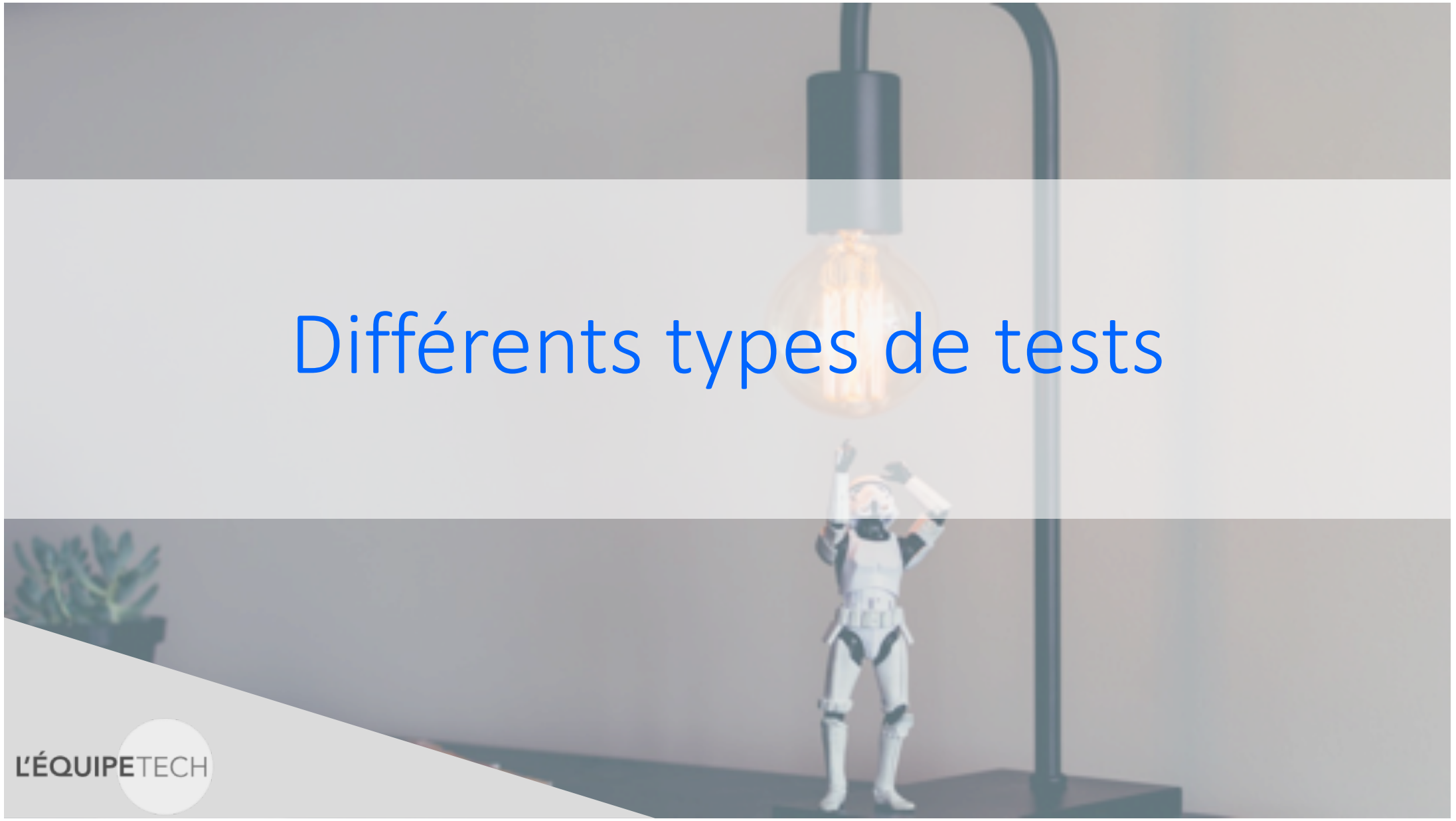
L'ÉQUIPETECH

Sommaire

L'ÉQUIPETECH

Sommaire

Différents types de tests



4 types de tests

- **Unitaire (Unit)** : Test avec un minimum de dépendances.
- **Noyau (Kernel)** : Test avec le chargement du noyau et un minimum d'extensions activés.
- **Fonctionnel (Functional)** : Tests avec une instance complète de Drupal.
- **Fonctionnel Javascript (FunctionalJavascript)** : Tests avec une instance complète de Drupal et l'utilisation de Webdriver (chromedriver) pour réaliser des tests sur des fonctionnalités Javascript.

Quel(s) outil(s) sous Drupal 8

L'ÉQUIPETECH

Historiquement < Drupal 8

Jusqu'à Drupal 7, utilisation de Simpletest :

- Un outil créé par la communauté Drupal
- Spécifique à Drupal
- Peu performant (rapidité), manque d'optimisation
- Doit être maintenu par la communauté

Drupal 8 : PHPUnit

A partir de Drupal 8, support de PHPUnit.
Dépréciation de Simpletest, il est recommandé d'utiliser PHPUnit.

Selon la version de PHP utilisée :
< PHP 7.2 : PHPUnit 4.8 (la plus utilisée dans la communauté)
>= PHP 7.2 : PHPUnit 6.5

Documentation : https://phpunit.readthedocs.io/fr/latest/

+ Tests fonctionnels javascript

Pour les besoins en tests fonctionnels javascript :

- Chrome ou chromium
- Chromedriver (webdriver)

```
$ chromedriver --port=4444
```

Lancer chromedriver pour pouvoir tester le javascript

Structure d’un test

L’ÉQUIPETECH

Où écrire ses tests ?

Les tests doivent se placer dans des endroits spécifiques.

Pour les modules contrib / custom :

`/modules/[custom/mon-module]/tests/src/[TypeDeTest]`

Exemple pour un test de type noyau (kernel) dans le module monmodule qui se trouve dans /module/custom :

`/modules/custom/monmodule/tests/src/Kernel`

Exemple pour un test de type Fonctionnel (Functional) dans le module monmodule :

`/modules/custom/monmodule/tests/src/Functional`

Comment nommer les namespaces ?

Le namespace de ces tests doit reprendre le namespace suivant :

```
\Drupal\Tests\[nom-module]\[TypeDeTest]
```

Exemple pour un test de type noyau (kernel) dans le module monmodule :

```
\Drupal\Tests\monmodule\Kernel
```

Exemple pour un test de type Fonctionnel (Functional) dans le module monmodule :

```
\Drupal\Tests\monmodule\Functional
```

Utilisation de code réutilisable

Dans le cas où du code peut être réutilisé dans plusieurs tests, utiliser les traits.

Les traits sont à placer dans :

`/modules/[custom/mon-module]/tests/src/Traits`

Et le namespace pour les traits sont :

`\Drupal\Tests\[nom-module]\Traits`

Nommer ses classes de tests

Les classes de tests doivent obligatoirement se terminer par le mot Test.

Le nom des classes doivent refléter la nature du test.

```
class FaireLeCafeTest extends BrowserTestBase {
  public function testQueLaMachineSAllume() {
    // Vos tests ici.
  }
}
```

Utilisation des metadatas PHPDoc

Les Metadata du tests sont définis grâce à la documentation PHPDoc dans la classe. Exemples :

Fichier: modules/ice_cream/tests/src/Unit/IceCreamTest.php

- **Nom**: Généré depuis le nom de la classe.
- **Description**: Générée depuis la ligne de résumé de PHPDoc. Toutes les classes de test DOIVENT avoir une ligne de résumé PHPDoc. Autrement, préciser dans PHPDoc un @coversDefaultClass qui défini la classe que l'on souhaite tester. Voir PHPUnit @coversDefaultClass annotation documentation
- **Groupe**: Généré depuis l'annotation PHPDoc @group. Tous les tests doivent préciser au moins un @group qui correspond au nom machine du module d'origine.
- **Dépendances**: Générée depuis l'annotation PHPDoc @requires module $name. Voir PHPUnit @requires annotation documentation. Si les tests requièrent l'installation de modules contrib complémentaires, voir the Managing dependencies Composer page.

```php
namespace Drupal\Tests\ice_cream\Unit;

/**
 * Tests generation of ice cream.
 *
 * @group ice_cream
 */
class IceCreamTest extends UnitTestCase {
  ...
}
```

```php
/**
 * @coversDefaultClass \Drupal\ice_cream\IceCream
 * @group ice_cream
 */
class IceCreamTest extends UnitTestCase {
  ...
}
```

Exemple pour @coversDefaultClass

Méthodes d'une classe de test

```php
<?php

namespace Drupal\Tests\aggregator\Unit\Menu;

use Drupal\Tests\Core\Menu\LocalTaskIntegrationTestBase;

/**
 * Tests existence of aggregator local tasks.
 *
 * @group aggregator
 */
class AggregatorLocalTasksTest extends LocalTaskIntegrationTestBase {

  /**
   * {@inheritdoc}
   */
  protected function setUp() {
    $this->directoryList = ['aggregator' => 'core/modules/aggregator'];
    parent::setUp();
  }

  ...
```

SetUp() : exécuté à l'initialisation du test. Permet de définir l'environnement à monter pour les tests.

Le test a effectuer est une méthode commençant par le mot test.

```php
  /**
   * Tests local task existence.
   *
   * @dataProvider getAggregatorAdminRoutes
   */
  public function testAggregatorAdminLocalTasks($route) {
    $this->assertLocalTasks($route, [
      0 => ['aggregator.admin_overview', 'aggregator.admin_settings'],
    ]);
  }

  /**
   * Provides a list of routes to test.
   */
  public function getAggregatorAdminRoutes() {
    return [
      ['aggregator.admin_overview'],
      ['aggregator.admin_settings'],
    ];
  }

}
```

getAggregatorAdminRoutes() : Méthode définissant la donnée à injecter dans testAggregatorAdminLocalTAsks() comme défini par @dataProvider.

L'ÉQUIPETECH

Fonctions globales simulées (Mocking)

La bonne pratique est d'injecter le service dont on a besoin plutôt que d'utiliser une fonction globale.

Cependant, si la classe a vraiment besoin d'utiliser une fonction globale, utiliser l'exemple suivant pour que le test fonctionne (cas de la fonction drupal_set_message()) :

Cet exemple provient du module aggregator qui utilise drupal_set_message. Note que cela ne fonctionne pas s'il existe un namespace avec le même nom : \drupal_set_message.

```
namespace Drupal\Tests\aggregator\Unit\Plugin;

class AggregatorPluginSettingsBaseTest extends UnitTestCase {}

namespace Drupal\Core\Form;

if (!function_exists('drupal_set_message')) {
  function drupal_set_message() {}
}
```

Simuler les valeurs de settings.php

- Dans les tests Unit et Kernel, simuler l'objet Settings, et s'assurer que le code qui se repose sur ces valeurs de settings.php l'utilise.
- Dans les tests Functional, utiliser :

```
$settings['settings']['my_settings_property'] = (object) [
'value' => 'my_value',
'required' => TRUE,
];
$this->writeSettings($settings);
```

Utiliser les simulations (mocks)

Exemple avec un test de modification (alter) de PhpTransliteration :

Mocks : Méthode à tester

```php
<?php

namespace Drupal\Core\Transliteration;

use Drupal\Component\Transliteration\PhpTransliteration as
BaseTransliteration;
use Drupal\Core\Extension\ModuleHandlerInterface;

/**
 * Enhances PhpTransliteration with an alter hook.
 *
 * @ingroup transliteration
 * @see hook_transliteration_overrides_alter()
 */
class PhpTransliteration extends BaseTransliteration {

  /**
   * The module handler to execute the transliteration_overrides alter
hook.
   *
   * @var \Drupal\Core\Extension\ModuleHandlerInterface
   */
  protected $moduleHandler;
```

```php
  /**
   * Constructs a PhpTransliteration object.
   *
   * @param string $data_directory
   *   The directory where data files reside. If NULL, defaults to
subdirectory
   *   'data' underneath the directory where the class's PHP file resides.
   * @param \Drupal\Core\Extension\ModuleHandlerInterface $module_handler
   *   The module handler to execute the transliteration_overrides alter hook.
   */
  public function __construct($data_directory, ModuleHandlerInterface
$module_handler) {
    parent::__construct($data_directory);

    $this->moduleHandler = $module_handler;
  }

  /**
   * Overrides
\Drupal\Component\Transliteration\PhpTransliteration::readLanguageOverrides().
   *
   * Allows modules to alter the language-specific $overrides array by
invoking
   * hook_transliteration_overrides_alter().
   */
  protected function readLanguageOverrides($langcode) {
    parent::readLanguageOverrides($langcode);

    // Let modules alter the language-specific overrides.
    $this->moduleHandler->alter('transliteration_overrides', $this-
>languageOverrides[$langcode], $langcode);
  }

}
```

L'injection de dépendance doit être simulée

Structure d'un test

L'ÉQUIPETECH

```php
<?php

namespace Drupal\Tests\Core\Transliteration;

use Drupal\Component\Utility\Random;
use Drupal\Core\Transliteration\PhpTransliteration;
use Drupal\Tests\UnitTestCase;

/**
 * Tests Transliteration component functionality.
 *
 * @group Transliteration
 *
 * @coversClass \Drupal\Core\Transliteration\PhpTransl
 */
class PhpTransliterationTest extends UnitTestCase {

  /**
   * Tests the PhpTransliteration with an alter hook.
   *
   * @param string $langcode
   *   The langcode of the string.
   * @param string $original
   *   The string which was not transliterated yet.
   * @param string $expected
   *   The string expected after the transliteration.
   * @param string|null $printable
   *   (optional) An alternative version of the original string which is
   *   printable in the output.
   *
   * @dataProvider providerTestPhpTransliterationWithAlter
   */
  public function testPhpTransliterationWithAlter($langcode, $original, $expected,
$printable = NULL) {
    if ($printable === NULL) {
      $printable = $original;
    }

    // Test each case both with a new instance of the tr sliteration class,
    // and with one that builds as it goes.
    $module_handler = $this-
>createMock('Drupal\Core\Extension\ModuleHandlerInterface');
    $module_handler->expects($this->any())
      ->method('alter')
      ->will($this->returnCallback(function ($hook, &$overrides, $langcode) {
        if ($langcode == 'zz') {
          // The default transliteration of Ä is A, but change it to Z for testing.
          $overrides[0xC4] = 'Z';
          // Also provide transliterations of two 5-byte characters from
          // http://wikipedia.org/wiki/Gothic_alphabet.
          $overrides[0x10330] = 'A';
          $overrides[0x10338] = 'Th';
        }
      }));
    $transliteration = new PhpTransliteration(NULL, $module_handler);

    $actual = $transliteration->transliterate($original, $langcode);
    $this->assertSame($expected, $actual, "'$printable' transliteration to
'$actual' is identical to '$expected' for language '$langcode' in service
instance.");
  }

  /**
   * Provides test data for testPhpTransliterationWithAlter.
   *
   * @return array
   */
  public function providerTestPhpTransliterationWithAlter() {
    $random_generator = new Random();
    $random = $random_generator->string(10);
    // Make some strings with two, three, and four-byte characters for testing.
    // Note that the 3-byte character is overridden by the 'kg' language.
    $two_byte = 'Ä Ö Ü Å Ø äöüåøhello';
    // These are two Gothic alphabet letters. See
    // http://wikipedia.org/wiki/Gothic_alphabet
    // They are not in our tables, but should at least give us '?' (unknown).
    $five_byte = html_entity_decode('&#x10330;&#x10338;', ENT_NOQUOTES, 'UTF-8');
    // Five-byte characters do not work in MySQL, so make a printable version.
    $five_byte_printable = '&#x10330;&#x10338;';

    $cases = [
      // Test the language override hook in the test module, which changes
      // the transliteration of Ä to Z and provides for the 5-byte characters.
      ['zz', $two_byte, 'Z O U A O aouaohello'],
      ['zz', $random, $random],
      ['zz', $five_byte, 'ATh', $five_byte_printable],
    ];

    return $cases;
  }

}
```

Instanciation de la classe, appel de la méthode à tester et test du résultat

Préparation de l'interface à simuler

L'ÉQUIPETECH

Structure d'un test

```php
<?php

// Mock the field type manager and place it in the container.
$field_type_manager = $this->createMock('Drupal\Core\Field\FieldTypePluginManagerInterface');

$this->fieldType = $this->randomMachineName();
$this->fieldTypeDefinition = [
  'id' => $this->fieldType,
  'storage_settings' => [
    'some_setting' => 'value 1',
  ],
  'field_settings' => [
    'some_instance_setting' => 'value 2',
  ],
];

$field_type_manager->expects($this->any())
  ->method('getDefinitions')
  ->will($this->returnValue([$this->fieldType => $this->fieldTypeDefinition]));
$field_type_manager->expects($this->any())
  ->method('getDefinition')
  ->with($this->fieldType)
  ->will($this->returnValue($this->fieldTypeDefinition));
$field_type_manager->expects($this->any())
  ->method('getDefaultStorageSettings')
  ->with($this->fieldType)
  ->will($this->returnValue($this->fieldTypeDefinition['storage_settings']));
$field_type_manager->expects($this->any())
  ->method('getDefaultFieldSettings')
  ->with($this->fieldType)
  ->will($this->returnValue($this->fieldTypeDefinition['field_settings']));
```

Définition de la méthode à simuler

Définition des valeurs tests

expects() : défini les attentes avec en paramètre le type de matcher

method() : la méthode à simuler

with() : paramètres attendus par la méthode à simuler

will() : le résultat de la simulation

Un autre exemple de simulation avec méthodes possibles.

Mocks : Simulations complexes

Dans le cas où la simulation devient trop complexe

Lors d'un test d'une méthode unique, le constructeur et la méthode peuvent déclencher trop d'appels de méthode et de service Drupal. Le test va nécessité énormément de simulation. Si aucune autre possibilité n'est envisageable, tuer chaque méthode sauf celle qui est testée :

```
$methods =
get_class_methods('Drupal\comment\Entity\Comment');
unset($methods[array_search('preSave', $methods)]);
$comment = $this-
>getMockBuilder('Drupal\comment\Entity\Comment')
->disableOriginalConstructor()
->setMethods($methods)
->getMock();
```

Ceci créera une version de la classe d'entité Comment où chaque méthode retourne NULL sauf preSave(). Si NULL n'est pas adéquat, il est possible de modifier le résultat attendu :

```
$comment->expects($this->once())
->method('isNew')
->will($this->returnValue(TRUE));
```

Tests Noyau (Kernel) : initialisation

```php
<?php

namespace Drupal\Tests\content_moderation\Kernel;

(...)

/**
 * Tests with node access enabled.
 *
 * @group content_moderation
 */
class NodeAccessTest extends KernelTestBase {

  use NodeCreationTrait;
  use UserCreationTrait;
  use ContentModerationTestTrait;

  /**
   * The moderation information service.
   *
   * @var \Drupal\content_moderation\ModerationInformationInterface
   */
  protected $moderationInformation;

  /**
   * {@inheritdoc}
   */
  public static $modules = [
    'content_moderation',
    'filter',
    'node',
    'node_access_test',
    'system',
    'user',
    'workflows',
  ];
```

Définition des modules à installer

Création de la structure de la base de données en fonction des besoins.

```php
  /**
   * {@inheritdoc}
   */
  protected function setUp() {
    parent::setUp();

    $this->installEntitySchema('content_moderation_state');
    $this->installEntitySchema('node');
    $this->installEntitySchema('user');
    $this->installEntitySchema('workflow');
    $this->installConfig(['content_moderation', 'filter']);
    $this->installSchema('system', ['sequences']);
    $this->installSchema('node', ['node_access']);

    // Add a moderated node type.
    $node_type = NodeType::create([
      'type' => 'page',
      'label' => 'Page',
    ]);
    $node_type->save();
    $workflow = $this->createEditorialWorkflow();
    $workflow->getTypePlugin()->addEntityTypeAndBundle('node', 'page');
    $workflow->save();

    $this->moderationInformation =
\Drupal::service('content_moderation.moderation_information');
  }
```

Création du type de contenu pour les tests

Structure d'un test

Tests Noyau (Kernel) : initialisation

```
    // Création du schéma des entités en base de données.
    $this->installEntitySchema('content_moderation_state');
    $this->installEntitySchema('node');
    $this->installEntitySchema('user');
    $this->installEntitySchema('workflow');


    // Création en base de données des configurations présentes
dans les répertoire config/ des modules listés.
    $this->installConfig(['content_moderation', 'filter']);


    // Création des schémas listés pour les modules.
    $this->installSchema('system', ['sequences']);
    $this->installSchema('node', ['node_access']);
```

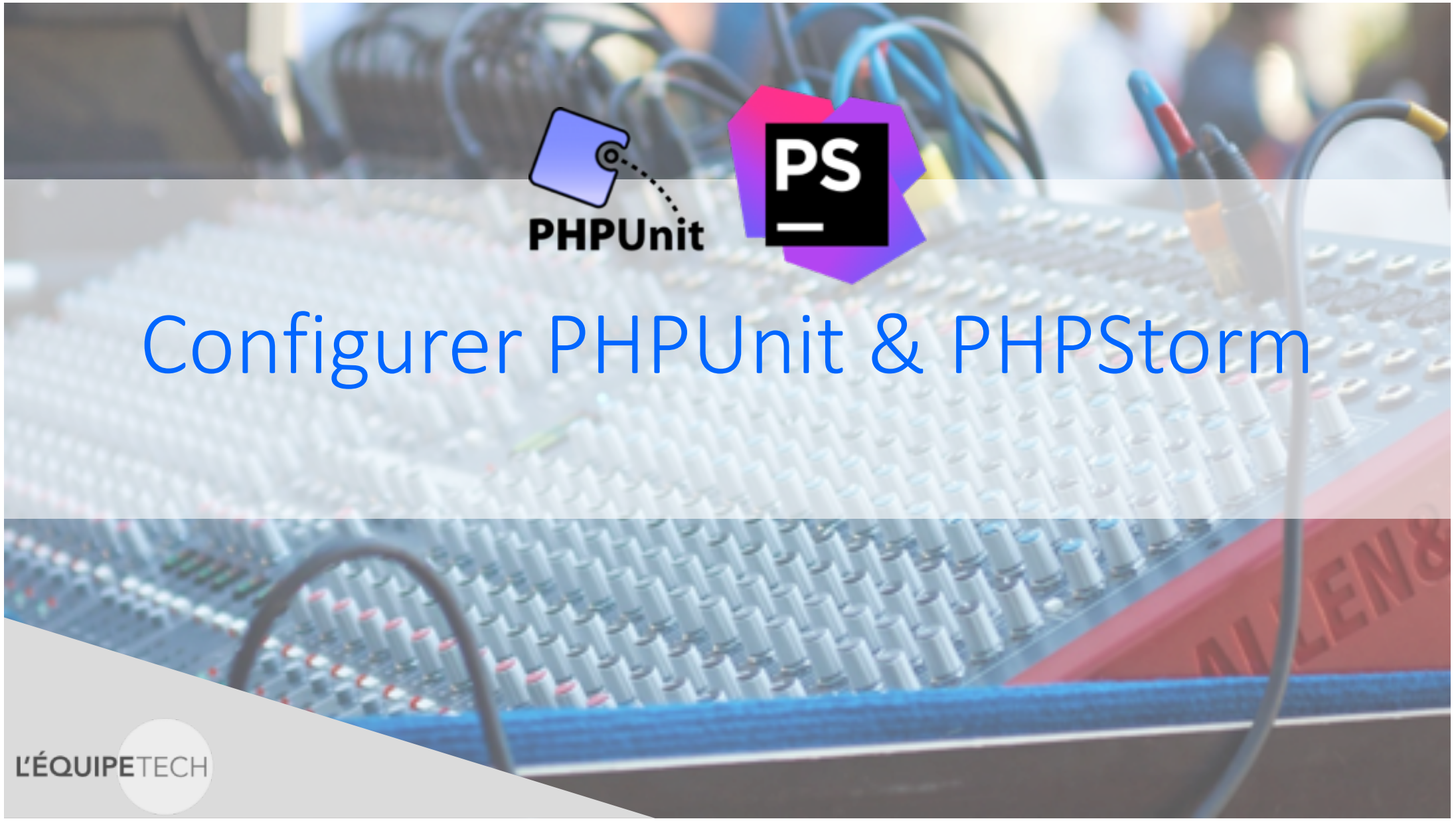
Tests Noyau (Kernel) : test

```
/**
 * Tests for moderation information methods with node access.
 */
public function testModerationInformation() {
  // Create an admin user.
  $user = $this->createUser([], NULL, TRUE);
  \Drupal::currentUser()->setAccount($user);

  // Create a node.
  $node = $this->createNode(['type' => 'page']);
  $this->assertEquals($node->getRevisionId(), $this-
>moderationInformation->getDefaultRevisionId('node', $node->id()));
  $this->assertEquals($node->getRevisionId(), $this-
>moderationInformation->getLatestRevisionId('node', $node->id()));

  // Create a non-admin user.
  $user = $this->createUser();
  \Drupal::currentUser()->setAccount($user);
  $this->assertEquals($node->getRevisionId(), $this-
>moderationInformation->getDefaultRevisionId('node', $node->id()));
  $this->assertEquals($node->getRevisionId(), $this-
>moderationInformation->getLatestRevisionId('node', $node->id()));
}
```

Création de l'utilisateur pour tester son accès

Création d'un contenu et vérification de ses informations

Création d'un utilisateur anonyme et vérification des informations récupérées suivant ses droits



Configurer PHPUnit & PHPStorm

Configuration dans Drupal

Dans le répertoire de fichier de drupal, créer le dossier :

`sites/default/simpletest`

Copier le fichier phpunit.xml.dist en phpunit.xml

`cp core/phpunit.xml.dist core/phpunit.xml`

Modifier le fichier phpunit.xml selon sa configuration :

```
<!-- Example SIMPLETEST_BASE_URL value: http://localhost -->
<env name="SIMPLETEST_BASE_URL" value="http://localhost"/>
<!-- Example SIMPLETEST_DB value: mysql://username:password@localhost/databasename#table_prefix -->
<env name="SIMPLETEST_DB" value="mysql://username:password@localhost/databasename#table_prefix "/>
<!-- Example BROWSERTEST_OUTPUT_DIRECTORY value: /path/to/webroot/sites/simpletest/browser_output -->
<env name="BROWSERTEST_OUTPUT_DIRECTORY" value="/chemin/vers/drupal/sites/default/simpletest/browser_output"/>
```

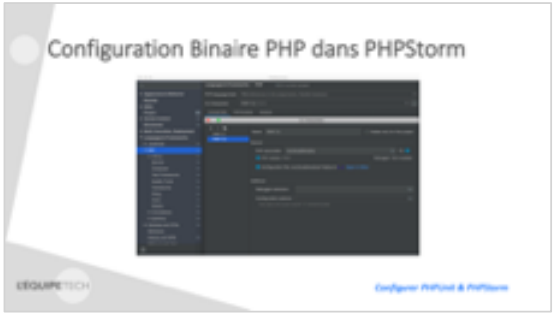
Configuration dans Drupal

Installer les paquets sous-traitant via composer pour obtenir phpunit.

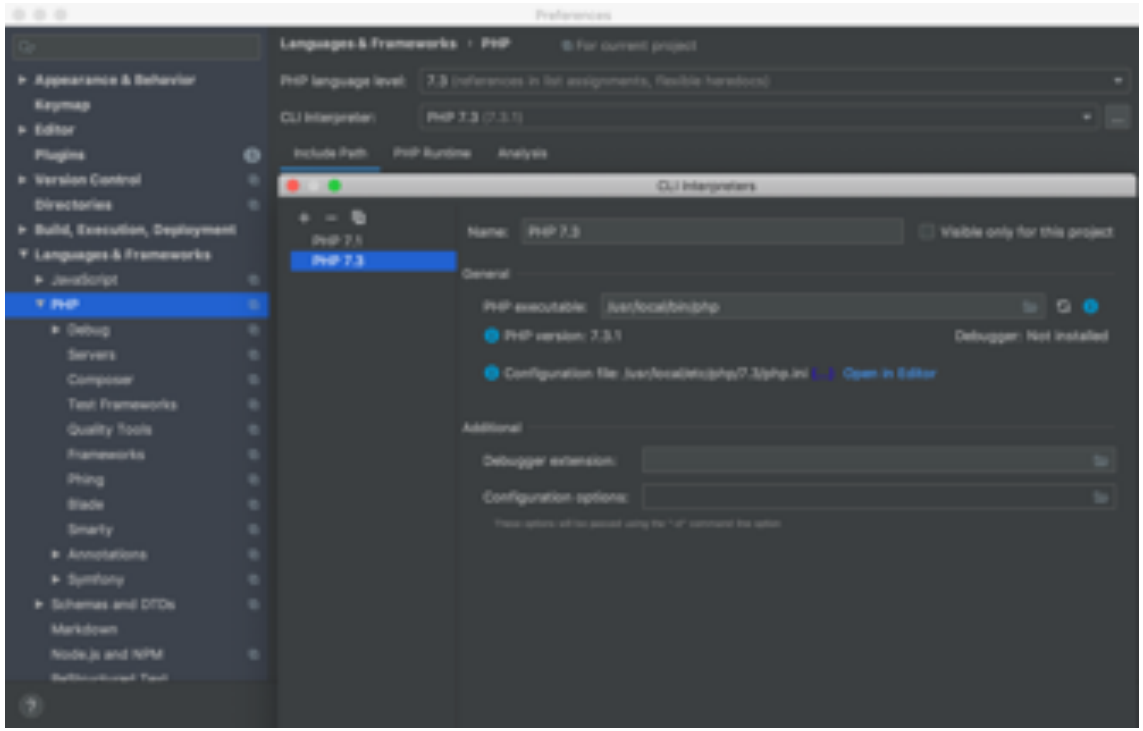
```
composer install
```

Configuration Phpstorm :

- Aller dans le menu PHPStorm > Preferences, chercher PHP et y ajouter son binaire PHP.

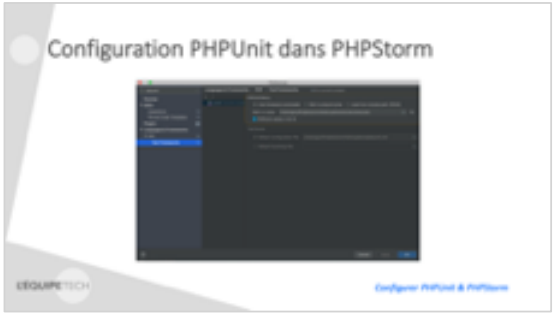


Configuration Binaire PHP dans PHPStorm

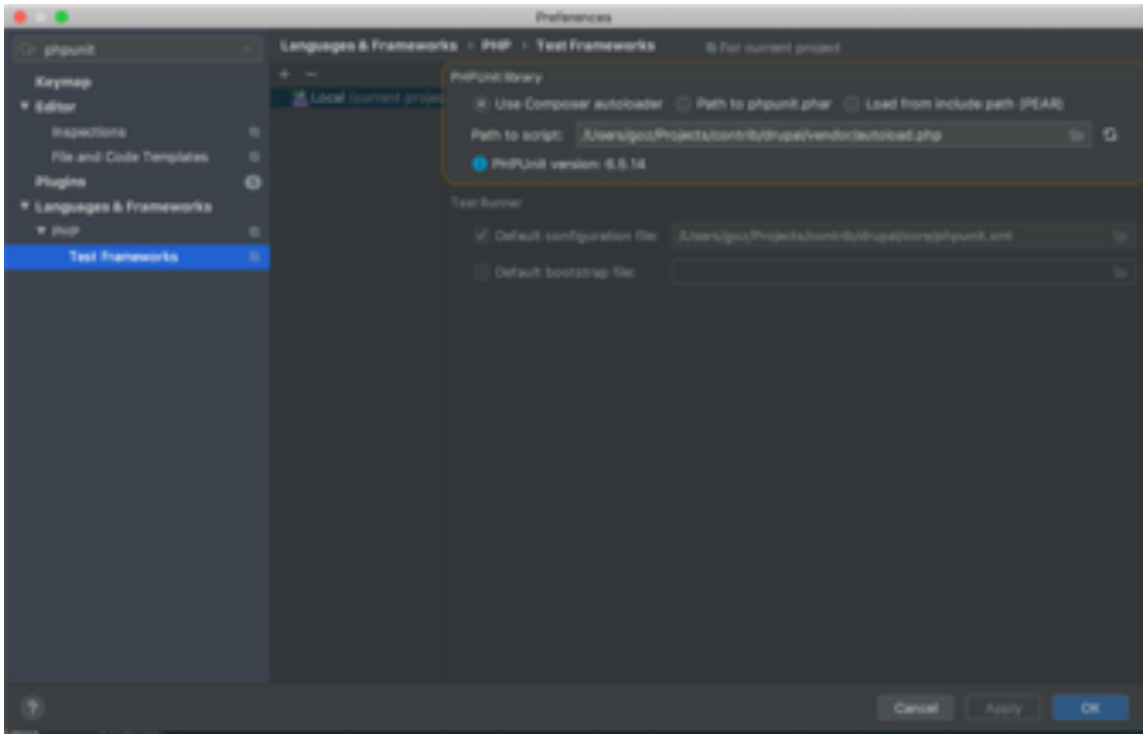


Configuration PHPUnit dans PHPStorm

Chercher « phpunit » et dans le menu Test Frameworks :



Configuration PHPUnit dans PHPStorm

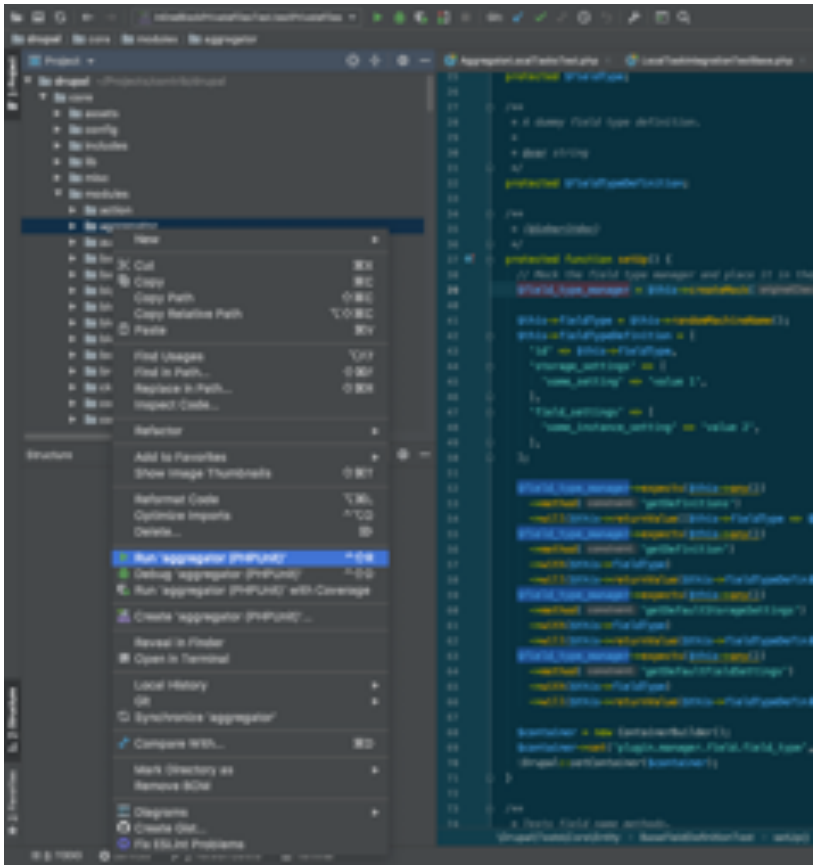


L'ÉQUIPETECH

Configuration PHPUnit dans PHPStorm

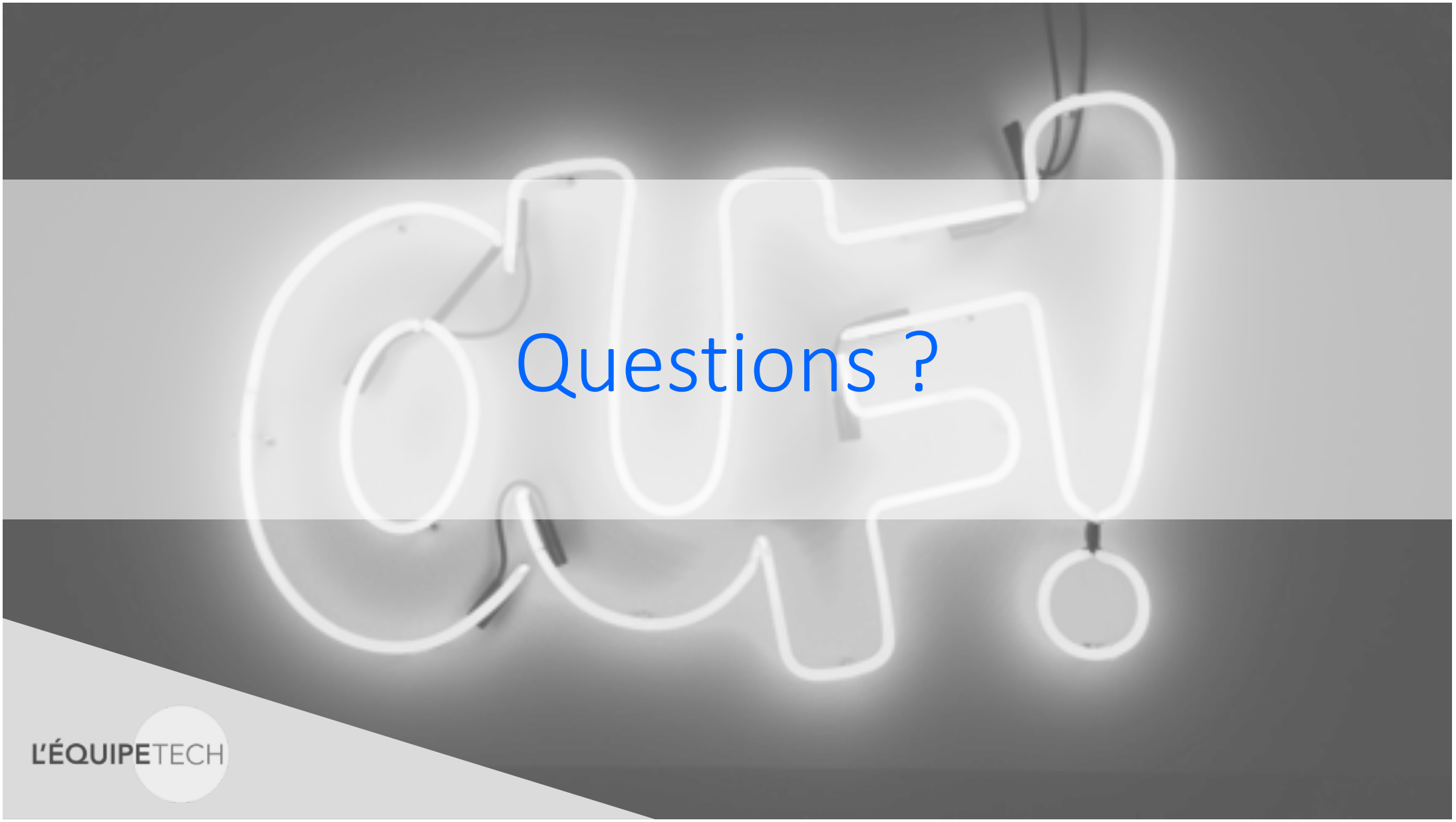
Pour lancer un test :

- Clic droit dans l'arborescence de fichiers du projet
- Run '(<repertoire> PHPUnit)'.



Configurer PHPUnit & PHPStorm

L'ÉQUIPETECH


Questions ?
L'ÉQUIPETECH