Unit testing MediaWiki with PHPUnit

WIKIMEDIA
FOUNDATION

About me

- Engineer on Growth team
- Code health metrics project
- Prague Hackathon

Wikimedia Foundation project

Growth

GROWTH

Active Projects [edit source]

Below is a summary of the active CHG projects.

Project	Purpose
DevEd	To develop educational resources for software engineers, software test engineers, software engineers in test
Code Review	To understand the code review challenges that we are facing and propose a course of action.

WIKIMEDIA
FOUNDATION

Agenda

- 📚 Theory
- 🏃🏾 Running tests
- ✏️ Writing tests
- !? Next steps

WIKIMEDIA
FOUNDATION

Does your code work the way you think it does?

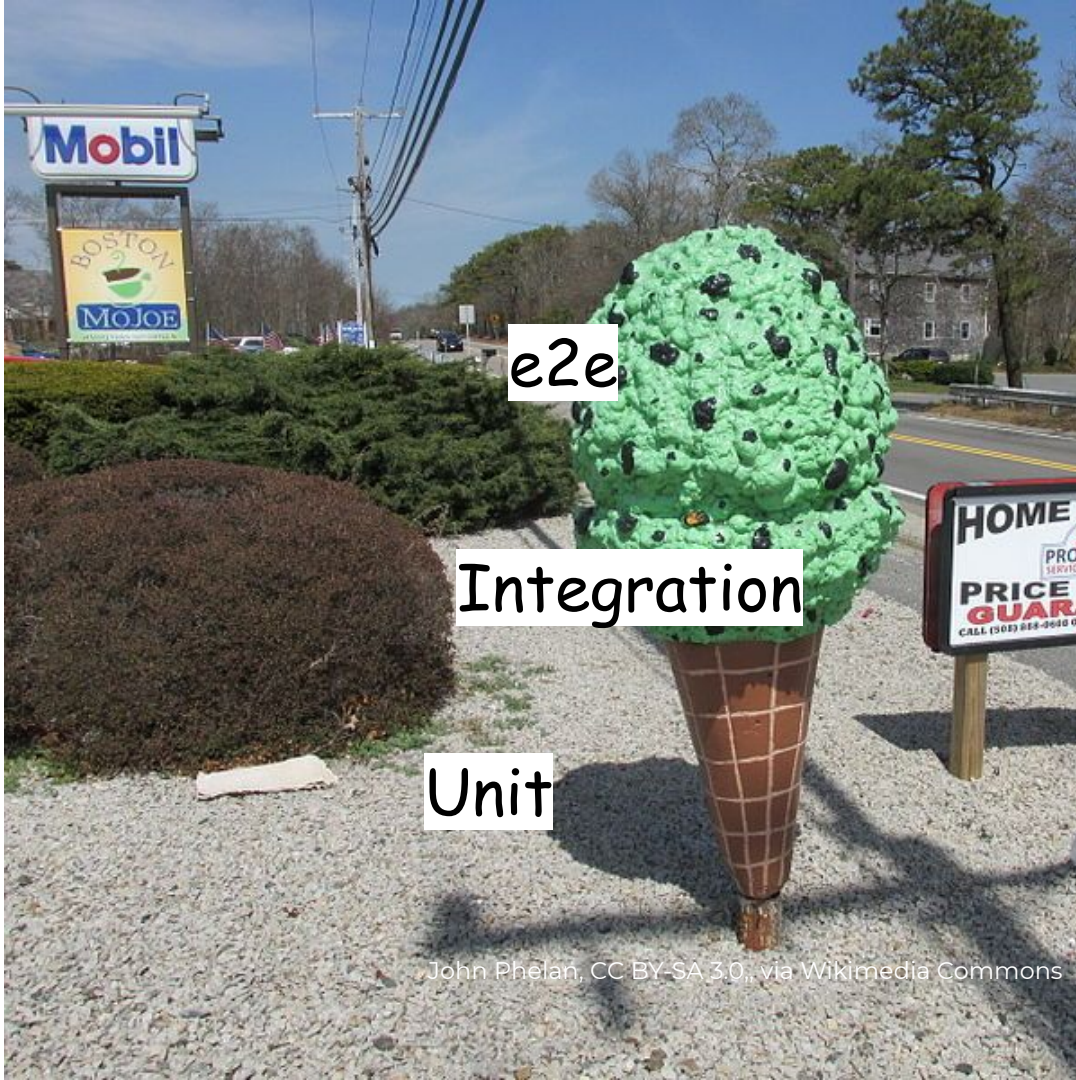
WIKIMEDIA
FOUNDATION

Can you and/or another human understand your code?

Unit testing as means towards correctness & comprehension

WIKIMEDIA
FOUNDATION

Types of tests



Ways to test MediaWiki

- Manual (end-to-end)
- Selenium (end-to-end)
- QUnit (unit / integration)
- api-testing (end-to-end)
- PHPUnit (unit / integration)

WIKIMEDIA
FOUNDATION

Terminology
a.k.a.
Naming things is hard

WIKIMEDIA
FOUNDATION

PHPUnit

“PHPUnit is a programmer-oriented testing framework for PHP.”

- https://phpunit.de/

WIKIMEDIA
FOUNDATION

xUnit

From Wikipedia, the free encyclopedia

> *For the particular .NET testing framework, see xUnit.net.*
>
> *For the unit of measurement, see x unit.*

xUnit is the collective name for several unit testing frameworks that derive their structure and functionality from Smalltalk's SUnit. *SUnit*, designed by Kent Beck in 1998, was written in a highly structured object-oriented style, which lent easily to contemporary languages such as Java and C#. Following its introduction in Smalltalk the framework was ported to Java by Kent Beck and Erich Gamma and gained wide popularity, eventually gaining ground in the majority of programming languages in current use. The names of many of these frameworks

What is a test?

You have some code that says it does something.

A test is a series of actions that *exercise* the code with various inputs to *assert* that it works the way you think it should.

WIKIMEDIA
FOUNDATION

Unit testable

- No dependencies on application state
- Does not “reach out”
- No database connection, network calls, file system access
- Given set of inputs output can be known **deterministically**

Joebeone at the English language Wikipedia, CC BY 2.5, via Wikimedia Commons

Live demo 2020 edition!

Joebeone at the English language Wikipedia, CC BY 2.5, via Wikimedia Commons

Not unit testable

- Call to global function `validateResult()`
- Access to global `$wgDemocraticNorms`
  `Exist`

Examples in MediaWiki

- Usages of global `$wg{someVariable}`
- Calls to global functions from `includes/GlobalFunctions.php`, e.g. `wfGetDB()`
- Calls to `MediaWikiServices::getInstance()`, `RequestContext::getMain()`, etc

WIKIMEDIA FOUNDATION

You can still test it, of course

```php
<?php
// phpcs:disable MediaWiki.Commenting.FunctionAnnotations.UnrecognizedAnnotati

use ...

/**
 * @since 1.18
 *
 * Extend this class if you are testing classes which access global variables,
 * or a storage backend.
 *
 * Consider using MediaWikiUnitTestCase and mocking dependencies if your code
 * injection and does not access any globals.
 *
 * @stable for subclassing
 */
abstract class MediaWikiIntegrationTestCase extends PHPUnit\Framework\TestCase
    use MediaWikiCoversValidator;
    use MediaWikiGroupValidator;
    use MediaWikiTestCaseTrait;

    /**
     * The original service locator. This is overridden during setUp().
     *
     * @var MediaWikiServices|null
     */
    private static $originalServices;

    /**
     * Cached service wirings of the original service locator, to work around
     * @var callable[]
     */
    private static $originalServiceWirings = [];

    /**
     * The local service locator, created during setUp().
     * @var MediaWikiServices
     */
    private $localServices;
```

WIKIMEDIA
FOUNDATION

Refactor for unit testing

WIKIMEDIA
FOUNDATION

Dependency injection

💡 The core idea of *passing dependencies in* instead of *reaching outside* is known as dependency injection.

Instead of using `global $wgFoo`, `wfFoo()`, `RequestContext::getMain()` or `MediaWikiServices::getInstance()`, set up your classes to inject dependencies into them.

This helps with **readability** and **refactoring** too!

WIKIMEDIA
FOUNDATION

Unit

- Test small functions in isolation
- No dependency on global state
- Deterministic results
- Fast

Integration

- Interaction with other parts of code base
- Knows about and overrides global state
- Race conditions
- Slower

WIKIMEDIA
FOUNDATION

Running the tests

WIKIMEDIA
FOUNDATION

CLI

- ✅ `vendor/bin/phpunit`
- ✅ `composer phpunit:unit`
- ⚠️ `php test/phpunit/phpunit.php`

⚠️ https://phabricator.wikimedia.org/T90875

WIKIMEDIA
FOUNDATION

IDE

AuthenticationResponseTest.php

```php
<?php

namespace MediaWiki\Auth;

/**
 * @group AuthManager
 * @covers \MediaWiki\Auth\AuthenticationResponse
 */
class AuthenticationResponseTest extends \MediaWikiUnitTestCase {
    /**
     * @dataProvider provideConstructors
     * @param string $constructor
     * @param array $args
     * @param array|Exception $expect
     */
    public function testConstructors( $constructor, $args, $expect ) {
        if ( is_array( $expect ) ) {
            $res = new AuthenticationResponse();
            $res->messageType = 'warning';
            foreach ( $expect as $field => $value ) {
                $res->$field = $value;
            }
            $ret = AuthenticationResponse::$constructor( ...$args );
            $this->assertEquals( $res, $ret );
        } else {
            try {
                AuthenticationResponse::$constructor( ...$args );
                $this->fail( 'Expected exception not thrown' );
            } catch ( \Exception $ex ) {
                $this->assertEquals( $expect, $ex );
            }
        }
    }
```

WIKIMEDIA
FOUNDATION

Writing tests

- Directory structure
- Base class
- Traits
- Assertions
- @covers tags
- Mocks

When should you write tests?

WIKIMEDIA
FOUNDATION

Next steps

- `tests/phpunit/unit`
- #MediaWiki-Core-Testing in Phabricator
- Manual:PHP unit testing
- Code health developer education workshops (https://w.wiki/jps)
- #wikimedia-codehealth

WIKIMEDIA
FOUNDATION

Thank you!