axelerant

PHPUnit

Write your first PHP Unit test

axelerant

Swarad Mokal

Drupal Developer

@swarad07

Writing tests

The get your hands dirty approach …

In computer programming, **unit testing is a software testing method by which individual units of source code**, sets of one or more computer program modules together with associated **control data**, usage procedures, and operating procedures, **are tested to determine whether they are fit for use**

- Wikipedia

Unit testing

Is good for your health!

In the absence of tests

- Write step-by-step procedures or checklists.
- Manually run them everytime we have to test if something works.
- Maintain an updated documentation.

Issues:

- Manual overheads.
- Human errors.
- Difficult to run regression for old test cases.
- Not an optimal way to identify problem
- Time consuming and people dependent.

A better alternative

- Write code to test your code - Enter PHPUnit!
- Find problems early
- Force better code design
- Follow TDD
- Facilitates changes to the code
- Can serve as a live documentation.

15-35% ↑

A Microsoft study found development time increased for product teams which wrote unit tests as part of development.

(https://link.springer.com/article/10.1007%2Fs10664-008-9062-z)

40-90% ⬇

The same study found that the number of bugs for the same projects went down.

(https://link.springer.com/article/10.1007%2Fs10664-008-9062-z)

Drupal 7.x - SimpleTest

Drupal 8.x - PHPUnit

Installation

```
$ composer install --dev
$ cd core
$ cp phpunit.xml.dist phpunit.xml
```

*For running browser test you also need phatomjs installed

Running tests

```
$ cd core
$ ../vendor/bin/phpunit
```

```
$ phantomjs vendor/jcalderonzumba/gastonjs/src/Client/main.js 8510 1024 768
```

*Run phantomjs

Unit tests

Base Class to extend: UnitTestCase

Directory: [module]/tests/src/Unit

To test functions, methods, classes.

Pros

- Easy to write
- Fast
- No system setup needed
- Finds problems in code quickly
- Test code not user interface

Cons

- Just verifies a single unit. Cannot confirm if whole system works.
- Rewrite on every core refactoring

Kernel tests

Base Class to extend: KernelTestBase or EntityKernelTesBase

Directory: [module]/tests/src/Kernel

Kernel tests are integration tests that test on components. You can install modules. You have access to database.

Pros

- Verify components work
- Finds problems in code somewhat quickly

Cons

- System setup needed.
- Slower execution
- Still no guarantee that system actually works.

Browser tests

Base Class to extend: BrowserTestBase

Directory: [module]/tests/src/Functional

Test system similar to how end user would use it, often termed as Functional Testing. Browser tests are essentially web tests.

Pros

- We have access to a browser and database.
- Verify system works after code changes

Cons

- System setup needed.
- Slower execution
- No javascript support in the browser.

Javascript tests

Base Class to extend: JavascriptTestBase

Directory: [module]/tests/src/FunctionalJavascript

These are essentially Browser tests with javascript support in the browser. As we have javascript support we can test end user's browser behaviour via code.

Pros

- Javascript support.
- Verify system works after code changes.

Cons

- System setup needed.
- Slower execution
- Hard to get to the origin of a bug.

Conclusion

- You need combination of tests, there is no such thing as 100% coverage.
- Better code design and testable code a must for tests.
- Writing Unit tests demand time & effort.
- While, writing tests is not hard, writing good tests in not easy.
- Treat your tests just like your code, don’t write and forget.
- Think about TDD, you will more often end up writing well structured code that can be tested.

Final thoughts and references

- Mocking and Prophecy is a thing to learn
- PHPUnit extensions available for selenium, dbunit.
- There is a phpunit_example in examples module
- Good tutorials and documentation here: https://www.drupal.org/docs/8/phpunit
- More examples and documentation here: https://phpunit.de/
- You can run phpunit test cases as part of your CI.
- Behat tests are a good alternatives to complex Functional Test scenarios.
- Remember, tests are good! PHPUnit in D8 is awesome.