What's Changed? Only running PHPUnit Tests for Changed Classes

May 14, 2017 • @IcyApril

Last night I was working on adding some features to a pretty large PHP project, the project had incredibly thorough Unit Testing. Whilst having too few Unit Tests is common issue for developers, in this case I was experiencing the exact opposite challenge; there were too many tests.

The project had hundreds of thousands of Unit Tests (in some cases they really constituted Integration Tests) and it would take upwards of 40 minutes to run the entire test suite. After running a single change it simply wasn't feasible to run the entire test suite to check everything was all working. I needed a way to only test the classes and run the unit test files which had changed during development, when I actually want to merge the code in that's when my CI build system would run all the Unit Tests across all the required platforms.

The project directory was largely standardized around the pds/skeleton standard. The `src` directory contained classes (e.g. `Class.php`) which in turn had an associated unit test in the `tests` folder (e.g. `ClassTest.php`). The directory structure in the `src` directory and the `tests` folder are identical (if a class exists in `src/Core/File.php` the associated unit test exists in `tests/Core/FileTest.php`). For example:

```
├── LICENCE
├── README.md
├── composer.json
├── composer.lock
├── phpunit.xml
├── src
│   ├── Config.php
│   ├── File.php
│   └── Tail.php
├── tests
│   ├── ConfigTest.php
│   ├── FileTest.php
│   └── TailTest.php
└── vendor
    ├── ...
```

PHPUnit does have a way to group Unit Tests, you can organize different test suites into different folders and can use the `--filter` argument in PHPUnit to run specific tests. There is even an `@group` annotation you can run tests by in PHPUnit. Unfortunately neither of these solve my problem; firstly the existing codebase doesn't group tests logically and even if it did my changes to the codebase may not necessarily be aligned to a particular grouping.

In PHPUnit it is possible to run Unit Tests individually simply by running `./vendor/bin/phpunit tests/SomeTest.php` but this is fairly tedious to do manually. You have to manually build a list of every Unit Test which has changed and every Unit Test which is associated to a changed class.

What's Changed?

Out of frustration I took a few minutes out of my overnight coding session to build a tool called "What's Changed". It's a very small plugin you can add to Composer that will only test files which have recently changed according to Git. Files which are tested are those which are changed in the working tree since your last commit and those modified in the previous commit.

Constraints

- Only works with files ending with `.php`
- Assumes your classes are in src and tests are mapped in the exact file structure in the tests folder
- If a Unit Test in the `tests` folder is changed, that test is re-run
- If a class in the `src` folder is changed, the associated (if existent) is run

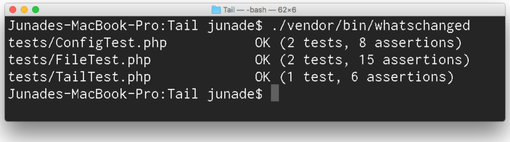
How to Use

You must have a compliant directory structure and

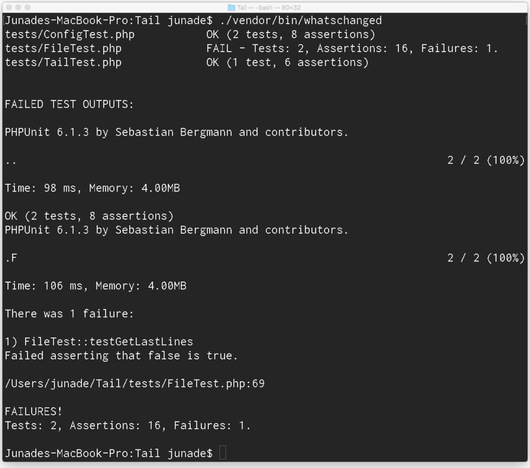
1. Pull in via Composer: `composer require --dev icyapril/whats-changed`
2. Run: `./vendor/bin/whatschanged`
3. Magic!

What it looks like

Successful Test



Test with Failures



What's Next?

This is very much an MVP version of what's possible here. There a few options as to where this experiment can go next.

- Static Code Analysis to work out collateral changes where the Single Responsibility Principle isn't followed
- Ability to customized operation to a different directory structure of a project
- Needs a code clean-up (quick and dirty and I don't write my best at 5AM)

You can check out the source on GitHub and check the project out on Packagist.

mjsa@junade.com
IcyApril
IcyApril
Junade Ali is a British computer scientist with specialist knowledge of computer security, distributed systems and software design. (more)