**UnitTesting**

Simply put, a unit test is a single function whose job is to verify another piece of code (a unit) is working correctly. They should be written by the developer who first writes the unit of code under test as they write it. This allows them to verify the code does what they expect it to and to find bugs with the code before QA do.

A good unit test should be simple, small, focused and repeatable. It should set up all the test data it needs and remove all the data after it has finished to prevent

**Role in preventing bugs**

**Role in refactoring**

Unit tests become very useful when code is being refactored provided the

**TDD**

**Examples**

**PHP Testing Frameworks**

Not specifically my area but there are a few libraries that I have been able to find from a quick google, SimpleTest and PHPUnit.

Both have documentation, although PHPUnit’s seems more comprehensive to me. Take a look and decide what fits you needs better:

<https://phpunit.de/index.html>

<http://www.simpletest.org/>

**Test Coverage Tools**

Increasing the total number of lines covered by unit tests isn’t the whole story, as mentioned in previous sections, but it is a good place to start. To make this effective you need to be able to find out how much of your code is covered by unit tests. There are many tools that can do this and it looks (again, from a quick google) that PHPUnit has build in facilities to calculate code coverage metrics (<https://phpunit.de/manual/current/en/code-coverage-analysis.html>). Once you can monitor your metrics, you can set about improving them and increasing Unit Test coverage always makes a great Sprint Kaizen.