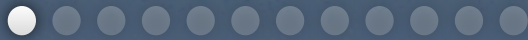
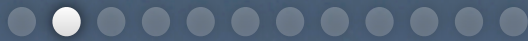


Testing with PHPUnit

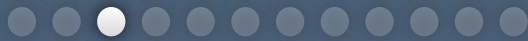


```
use Caution;  
while ( MediaWiki && PHPUnit ) {  
    // TODO: Write tests  
}
```



Yeah, yeah, we know we need more tests – but it's more than just having them around
Unit testing, like any other superpower, can be used for good or evil
Let's talk about how to do this the right way

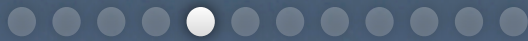
Unit Tests



Integration Tests



Keep 'em separated



Everything in between is evil, keep a safe distance between them

Assertions



- You need at least one assertion (or the test is useless)
- But too many will spoil the soup (tests shouldn't be too busy)
- Use the most specialized assertion available

Test names

“test” + NameOfSystemUnderTest

Like: testWfMsg or testFileUploads

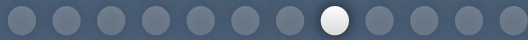


- Always start with the word “test”
- End in a TitleCase description of the system under test
- Symbol name for unit tests
- Behavior description for integration tests

Assertion messages

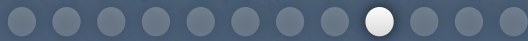
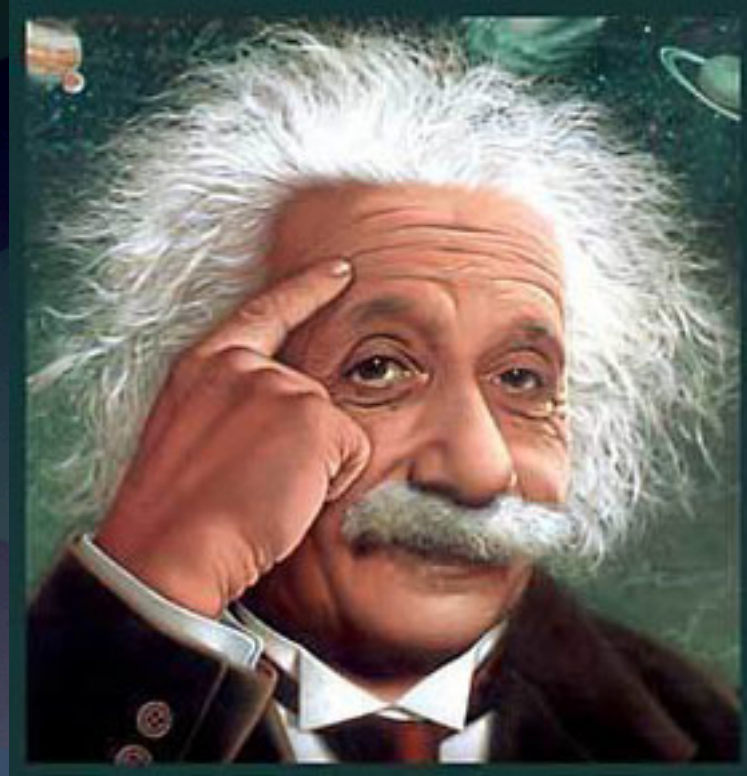
“Sentence case description”

Like: “Returns string of message key with angle brackets if message does not exist”



- Complete sentence, punctuation not necessary
- Should be very descriptive, tell a story

Inventing



Inventing

1



2

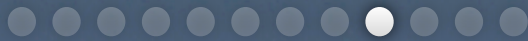


3

Write tests

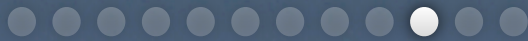
Write code

Run tests



1. Write tests as if you are done writing the code
2. Write the actual code
3. Run the tests to verify it works properly

Refactoring



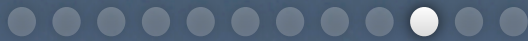
Refactoring



Change tests

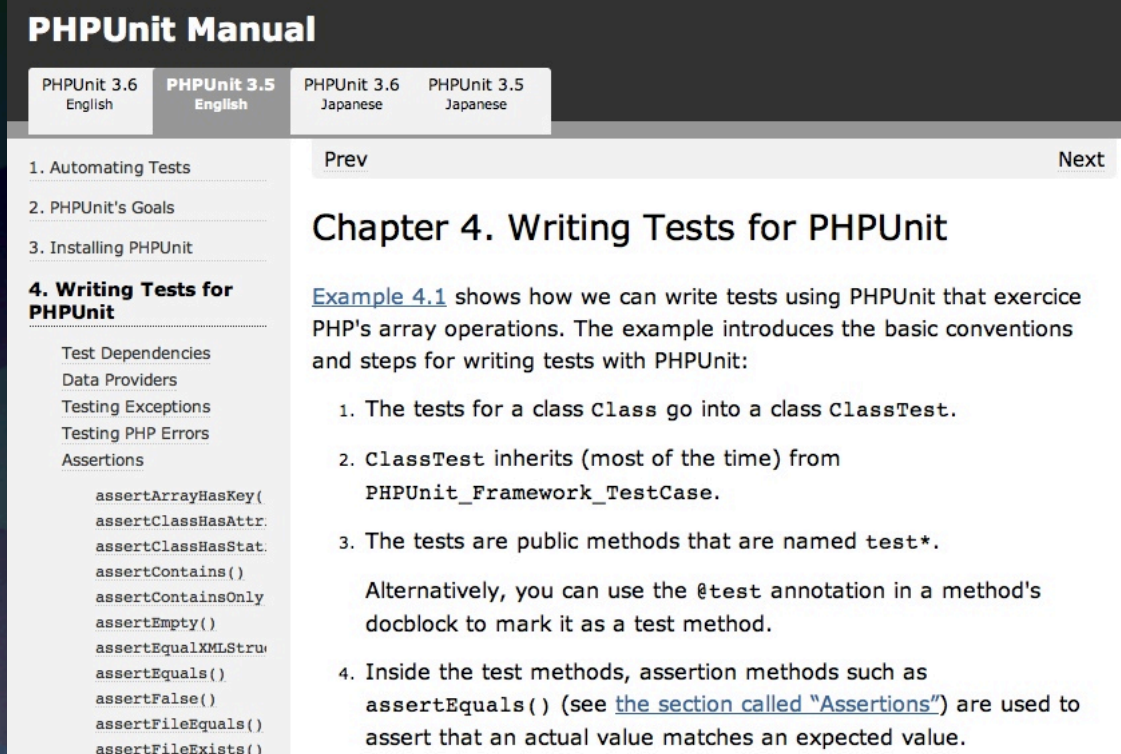
Change code

Run tests



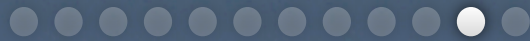
1. Modify the tests as if the changes have already been made
2. Make the actual changes
3. Run the tests to verify the changes were made correctly

RTFM!



The screenshot shows the PHPUnit Manual website. At the top, there are navigation tabs for different versions and languages: PHPUnit 3.6 English, PHPUnit 3.5 English, PHPUnit 3.6 Japanese, and PHPUnit 3.5 Japanese. The main content area is titled "Chapter 4. Writing Tests for PHPUnit". It includes a list of sections on the left: 1. Automating Tests, 2. PHPUnit's Goals, 3. Installing PHPUnit, 4. Writing Tests for PHPUnit (highlighted), Test Dependencies, Data Providers, Testing Exceptions, Testing PHP Errors, and Assertions. The main text of Chapter 4 explains that Example 4.1 shows how to write tests using PHPUnit that exercise PHP's array operations. It lists three steps: 1. Tests for a class `Class` go into a class `ClassTest`. 2. `ClassTest` inherits (most of the time) from `PHPUnit_Framework_TestCase`. 3. Tests are public methods named `test*`. It also mentions that alternatively, the `@test` annotation can be used in a method's docblock. Step 4 states that inside test methods, assertion methods like `assertEquals()` are used to verify values.

<http://www.phpunit.de/manual/current/en/>



Writing tests is like learning to cook, improvisation rarely yields a positive outcome
Read the manual and stick to the recipe – don't start getting fancy with the spices!

Happy testing!

