


Wednesday, November 21. 2007

## PHPUnit: Independent Mock Object/Stub Framework

Sometimes I think my readers must get terribly bored with my one track entries - I seem to roll through phases: a few weeks of OpenID, then a few of PHPSpec, then a detour for several months on X 

. That's what you get for reading an open source developer's weblog - we can only really work on one thing at a time so there's a lack of variety.

Anyway, after that minor detour, this one's a brief notice that we've commenced work (as of a while ago really) on PHPUnit.


If you remember, I had one of those long winded blog posts about Mock Objects and Stubs in SimpleTest/PHPUnit and PHPT. Now that I've really launched into PHPSpec development, I'm finding a few spots here and there where my attention levels drop and I need a diversion. Which means PHPUnit is seeing activity - there's a whole branch written over a few days to get an API discussion going between developers.

The purpose of PHPUnit is to write an independent Mock Object/Stub framework - something you can use in any scenario or xDD approach - from a PHPUnit Unit Test, to a PHPSpec Spec, to a PHPT test file. SimpleTest isn't left out either - PHPUnit is self contained so it should travel well.

Once we get over the Mock vs Stub approach question to such a framework (Are Mocks complicated Stubs, or Stubs simplified Mocks?) we'll have a firm design and an API to develop with. I'm currently favouring an API in a simple readable form, e.g.


```
// setup
$mockPerson = PHPUnit::mock('Person');
$mockPerson->shouldReceive('hasName')->withNoArgs()->once()->andReturn(false);
// implemented use case
$mockPerson->hasName(); //FALSE
// verification of Mock expectations
$mockPerson->verify(); //TRUE
```


Not sure if it will stay that way, have some alternate API appended (less verbose), or end up as something different. I'll let you know.

A good feeling about this small project is that we have peers checking in. Sebastian Bergmann for PHPUnit, me for PHPSpec, and Travis Swicegood for PHPT and maybe SimpleTest. So at least it's not just me rumbling around enforcing my divine will 

If you want to look around the experiemental branch code, we're hosted over at <http://code.google.com/p/phpmock/>.

Posted by PÃ¡draic Brady in PHP General, PHP Security at 01:19

Simple first, then add complexity 

It's always easier to add features and much, much harder to remove them. Just look at some of the old code in PEAR1... 

I will get around to that email I owe you, soon



Anonymous on Nov 21 2007, 03:51

The way I see it a stub is a dumb actor, whereas a mock is also a critic, which means it can observe itself on top of acting abilities. It's a stub that can look at what it does, so I think you should start from stubs and go from there. Maybe implement a Mock as a sort of decorator/interceptor for the Stub?

regards Anonymous on Nov 21 2007, 07:38

PHPMock is indeed really interesting. A nice feature would be to provide a possibility to do filter the return values of the mocked methods. I've built a mock objects implementation for our mock object based XSS barrier (see <http://usrportage.de/archives/841-Preventing-cross-site-scripting-by-design.html> for details) which is pretty similiar to what you did. So if I could do something like:

```
PHPMock::setFilter(new MyMockFilter());
```

```
$mocked = PHPMock::mock('MyObject');
```

```
$mocked->myMethod() would trigger MyMockFilter::filter() with the return value as argument. What do you think about adding this?
```

Anonymous on Nov 21 2007, 07:58

@Travis: I'm shuddering already - and are you positive whatever you have is not catching across copper wires?



@gasper: That's "the argument" idir Travis and myself. I believe Mock Objects are the core purpose of the framework, the majority use case. Travis is the opposite. When you have two opposites on a project, you get the same thing as in physics - so it's hard to figure out where to compromise between two committed souls. At least we do agree both Mocks and Stubs will be supported from the start.

If you follow that logic - the way I've captured Stubs is as dumb Mocks. When you call verify() on a Stub, you automatically get a TRUE result (i.e. even Stubs are verified (illusory) just so integration to, say, PHPUnit just has a single method to call. Anonymous on Nov 21 2007, 17:27

Hi Lars,

It's absolutely brilliant to meet someone else who believes in filtering before deisgners get their mits on anything



. I do something very

similar for a few projects using the Zend Framework since I really hate using it's View escape() method (it's boring to have to write it all the time too).

On the filtering bit, I'd identify a need for global filters (effects all return values) and Expectation filters (effect only a single method/argument pair forming a unique Expectation on a Mock Object). It wouldn't be difficult to add so I'll keep it on my todo list for when we start working in trunk on this. Anonymous on Nov 21 2007, 17:58

The majority of my work is done with Ruby on Rails where I've become a serious convert to rspec and mocking. One thing I find invaluable in rspec's mocking framework is the ability to be able to stub or mock class methods, eg.

```
Post.stub!(:find).and_return(@parent)
```


Now back working on a PHP project I'd really like to be able to do something similar, particularly when specing/mocking web-service components. So I'd be able to transparently replace the HTTP class that my Service class would have called with a mock.

Is that sort of functionality in the works for PHPMock? I guess it would be harder to implement in PHP than in ruby, but it'd definitely be very useful for specing complex object hierarchies. Anonymous on Nov 22 2007, 22:29


Yes, that's definitely on the cards. At the moment all methods are initially tampered with - PHPMock is present on an experimental branch until everyone involved has a good debate about what to put into it so it only has an all-in approach for now.

The API will have to depart rspec's though. PHP can't replicate the object calls from Ruby. I'll likely do something where you generate a partial explicitly (say PHPMock::partial('Class')) which preserves all methods as-is until you decide to make a "shouldReceive" call to signal it should be mocked individually. Even that might too complex to implement - in which case it would mean declaring which methods you want to Mock/Stub when generating the partial - not ideal, but I'm stuck with PHP's limitations.

Sound like a plan?

P.S. It's a newborn library, but PHPMock is being developed using PHPSpec  to apply some BDD. Not quite up to the polished level of rspec but we're getting there with it and hopefully a first alpha turns up soon. Anonymous on Nov 23 2007, 01:16

Is there a mailing list for PHPMock yet? I viewed the google project page and couldn't find one yet.

I would like to discuss the possibility of introducing function mocking (as an addon) to PHPMock, but blog comments aren't quite the best venue  . Anonymous on Dec 3 2007, 13:46

We haven't added a specific public mailing list for it just yet. I would suggest posting to the public phpspec-dev mailing list for the moment - once a dedicated public PHPMock mailing list is up and running we'll move all public discussion there (and maintain PHPMock's independence from PHPSpec which is important).

At the moment most of the PHPMock discussion is in private forums (email/IM/etc) so we haven't need a public list just yet. Anonymous on Dec 3 2007, 17:54