

Forum:[Hack Forums](#)

Thread: [Art of Unit Testing](#)

Snoweyes - December 23, 2014, 1:01 am

Hello guys, this week i've got some fresh food from my work on my table and i think it is pretty interesting!

So i thought about sharing it with you guys.

I've been watching these presentations of Roy Osherove and learned a few new things about Unit Testing.

Unit Testing is not new for me, i already knew how to write a valid test but these presentations gives you a push in a direction to code better software while you write tests.

- What is Unit Testing?

A Unit Test is made to test your application automatically, this means you write a test for your application based on a condition.

You know that this condition is valid and your application has to adapt to it and so you write a test for it that verifies this.

When the test is written correctly and it fails you know that there is a bug in your application.

More info about Unit Testing in general goes here:

[http://msdn.microsoft.com/en-us/library/aa292197\(v=vs.71\).aspx](http://msdn.microsoft.com/en-us/library/aa292197(v=vs.71).aspx)

http://en.wikipedia.org/wiki/Unit_testing

- Useful presentations

I've been watching these presentations at <http://artofunittesting.com/>

But those i find really useful are these three:

https://www.youtube.com/watch?v=fAb_OnooCsQ

<https://www.youtube.com/watch?v=q5Xd1tmlgec>

https://www.youtube.com/watch?v=dJUVNFxrK_4

- For the lazy guys!

If you dont like watching those presentations that are in total 3 hours long then here are some useful tips :).

- Tests may never ever fail, so do not tell ur fellow worker that the test is supposed to fail.

- Only test logical things, so testing methods that has only an if or a loop in it with no logic conditions then it has no use to test it.

- Do not change bad code before you proved it with a test.

- Create fake classes when you are using a third party framework, do not test the framework. Because this framework has already been tested by someone else and you do not want your tests to be depending on a third party framework.

- Test method names may be long but keep it readable. You can see your test methods as documentation and an API.
- Try to have test reviews with your fellow workers, this will help each other a lot.
- As mentioned above you can make fake classes that implements the same interface as the framework or class you do not want to invoke. These fakes are called stubs, when you use the stub to assert in your test then they are called a mock.
- Do not remove or change tests, if you build the tests correctly then they exist for a reason.

I hope you guys find this useful and i hope i explained it correct to you.
If you have any feedback or tips, please feel free to tell me :).

Truth - December 23, 2014, 1:04 am

I am too lazy to unit test my shit lol if it works once it works :D good tut

Snoweyes - December 23, 2014, 1:07 am

(Dec 23, 2014 02:04 AM)TryAgain Wrote: [\[CITE\]](#) I am too lazy to unit test my shit lol if it works once it works :D good tut

This is indeed the biggest mistake most people make ^^.

But at my work we have over 10k lines of code (maybe more) and this needs to be tested!

Because if we dont and we change code inside the application we dont know if something broke down in another place.

+Bourbon - December 23, 2014, 2:03 am

Although having a regression suite is certainly a compelling benefit of testing, testing, in my opinion, pertains to more than detecting regressions.

Writing tests:

gives you immediate feedback about the quality of your design ideas;

encourages you towards code that is both loosely coupled and highly cohesive;

adds an executable specification of the what the code should do; and,

lets you know when you've done enough, discouraging [gold plating](#) and from adding functionality until deemed necessary (because [YAGNI!](#)).