Cheatography

C# Unit-Tests - Advanced Cheat Sheet

by Sérgio Ferreira (AlienEngineer) via cheatography.com/89520/cs/20391/

Implementation testing

```
public void Test() {
   // Arrange
   var mock = new Mock<I Act ion Int erf ace >();
   var subject = new MySubj ect (mo ck.O bj ect);

   // Act
    sub jec t.D oSo met hing();

   // Assert
    moc k.V erify(e => e.Stor e(I t.I sAn y<i nt>
()), Times.O nce);
}
```

- Locks tests with the dependency:
  - The test method is aware of IActionInterface.
  - Creates duplicated code.
- Changes to MySubject constructor forces changes throughout all the tests.

Verify vs Callbacks

```
// If one of these equality comparisons fail we
just know that MyAction wasn't called with those
arguments.
mock.V erify(e => e.MyAc tion(
    It.I s< Par ame ter Typ e>( param =>
        par am.F ield1 == "some value" &&
        par am.F ield2 == 3),
    Tim es.O nce);
// Instead:
Parame terType calledArgs = null;
mock
   .Se tup(e => e.MyAc tio n(I t.I sAn y<P ara -
met erT ype >()))
   .Ca llb ack <Pa ram ete rTy pe> (param =>
calledArgs = param);
Assert.Th at( cal led Args, Is.Not.Null);
Assert.Th at( cal led Arg s.F ield1, Is.Equ alT -
o("some value"));
Assert.Th at( cal led Arg s.F ield2, Is.Equ alT -
o(3));
```

Tip: Avoid stating in your tests how things were done, instead strive to describe what happened. **Observable Behavior**

Behavior testing

```
private int _someVariable;

private MySubject MakeSu bject()
{
   var mock = new Mock<I Act ion Int erf ace >();

    moc k.S etup(e => e.Acti on( It.I sA ny< int >
()))
           .Ca llb ack <In t>(s => _someV ari able=
s);

    return new MySubj ect (mo ck.O bj ect);
}

private bool WasSto red(int value) =>
    _so meV ariable == value;

[Test]
public void SomeTe stM ethod()
{
   // Arrange
   var subject = MakeSu bje ct();

   // Act
    sub jec t.D oSo met hing();

   // Assert
    Ass ert.Tr ue( Was Sto red (5));
}
```

- Changes to dependencies are done in one place only.
- Focuses on Behavior not implementation.

Dependencies - Common missed tests

What happens when the dependency..

- throws exception?
- returns null?
- returns wrong format? (e.g. expected json)

Tip: Testing is about making sure we have answers to questions. Testing is done when we don't have doubts, therefore no more questions.



By **Sérgio Ferreira** (AlienEngineer)

cheatography.com/alienengineer/

Published 28th August, 2019.
Last updated 3rd September, 2019.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Web Apis

```
internal abstract class WebApiTests {
    private WebApp lic ati onF act ory <St art up>
_factory;

    pro tected WebApi Tests() =>
        _fa ctory = new
WebApplicationFactory<Startup>();

    pro tected HttpClient GetCli ent() =>
        _fa cto ry.C re ate Cli ent();
}
class MyApiC ont rol ler Tests : WebApi Tests {
    public async Task reques tTest() {
       // Arrange
       var client = GetClient();

       // Act
       var response = await
client.GetAsync("api/someuri");

       // Assert
        res pon se.E ns ure Suc ces sSt atu sCo -
de();
       var data = await
response.Content.ReadAsAsync<Model>();

       // Assert data ...
   }
}
```

using **Microsoft.AspNetCore.Mvc.Testing** library for auto mocking out the http layer.

Files

```
// Hide file access through Stream Factory
public interface IStrea mFa ctory
{
    Stream OpenSt ream();
   void CloseS tre am( Stream stream);
}
// On test file
private IStrea mFa ctory MakeSt rea mSt orage()
{
   // Create one isolated Memory Stream per test
run!
    _stream = new Memory Str eam();
   var mock = new Mock<I Str eam Fac tor y>();
    moc k.S etup(s => s.Open Str eam ()).Re tur -
ns( _st ream);
    return mock.O bject;
}
```

Test arrange with files

```
void Test() {
   // slow & can fail
   var data = File.R ead All Tex t(@ " Dat a.j -
son ");
   // slow & can fail
   var x = JsonCo nve rt.D es eri ali zeO bje ct< -
MyM ode l>( data);
   // None of the above lines have anything to do
with the test. It's all about Arrange section.
}
```

Tip: If you feel the need to do this, probably your subject under test is doing to much. See **Don't ignore the signs!**



By **Sérgio Ferreira**
(AlienEngineer)

cheatography.com/alienengineer/

Published 28th August, 2019.
Last updated 3rd September, 2019.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com