

### ⚡ Implementation testing

```
public void Test() {
    // Arrange
    var mock = new Mock<I Action Interface >();
    var subject = new MySubject (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 Action Interface >();

    moc k.S etup(e => e.Acti on( It.I sAn y< 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)

Published 28th August, 2019.  
Last updated 3rd September, 2019.  
Page 1 of 2.

Sponsored by **CrosswordCheats.com**  
Learn to solve cryptic crosswords!  
<http://crosswordcheats.com>

### Web Apis

```
internal abstract class WebApiTests {
    private WebApplicationFactory<Startup>
        _factory;

    protected WebApiTests() =>
        _factory = new
        WebApplicationFactory<Startup>();

    protected HttpClient GetClient() =>
        _factory.CreateClient();
}

class MyApiControllerTests : WebApiTests {
    public async Task requestTest() {
        // Arrange
        var client = GetClient();

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

        // Assert
        response.EnsureSuccessStatusCode();

        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 IStreamFactory
{
    Stream OpenStream();
    void CloseStream(Stream stream);
}

// On test file
private IStreamFactory MakeStreamStorage()
{
    // Create one isolated Memory Stream per test
    run!
    _stream = new MemoryStream();
    var mock = new Mock<IStreamFactory>();
    mock.Setup(s => s.OpenStream()).Returns(
        _stream);
    return mock.Object;
}
```

### Test arrange with files

```
void Test() {
    // slow & can fail
    var data = File.ReadAllText(@"Data.json");
    // slow & can fail
    var x = JsonConvert.DeserializeObject<MyModel>(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/](http://cheatography.com/alienengineer/)

Published 28th August, 2019.

Last updated 3rd September, 2019.

Page 2 of 2.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>