

### Stub - Just a replacement

```
// Create a stub
var mock = new Mock<IMyDependency>();
// Get the Stub
var myDep = mock.Object;
Assert.That(myDep.GetValue(), Is.EqualTo(0));
```

**By default:** Methods with returning types return default values. (e.g. `int GetValue()` returns 0;

### Spy - What happened?

```
// Create a spy
var mock = new Mock<IMyDependency>();
// Spy on executions
mock.Verify(myDep => myDep.GetValue());
```

#### Optional:

Verify can be called with optional argument that asserts how many times the method as called: `AtLeastOnce`, `AtLeast`, `AtMost`, `AtMostOnce`, `BetweenExclusive`, `BetweenInclusive`, `Exactly`, `Once`, `Never`

**Tip:** Always strive to use concrete expectations. Avoid `AtMost`, `AtLeast` and `Between`.

### Spy via Callbacks

```
// Create a spy
var mock = new Mock<IMyDependency>();
string usedArg;
mock
    // Setup to capture any call to done to
    // DoSomething method.
    .Setup(myDep => myDep.DoSomething(It.IsAny<String>()))
    // Callback function is called when DoSomething
    // was called.
    .Callback<String>(arg => usedArg = arg);

// Executing with "test" argument
mock.Object.DoSomething("test");

Assert.That(usedArg, Is.EqualTo("test"));
```

### Mock - Do what I say!

```
// Create a mock
var mock = new Mock<IMyDependency>();

mock
    // What behavior is going to be mocked
    .Setup(myDep => myDep.GetValue())
    // What's going to be returned by GetValue()
    .Returns(1);

// Using the object
var myDep = mock.Object;

Assert.That(myDep.GetValue(), Is.EqualTo(1));
```

### Mock - Setup - Matching Arguments

Expected Argument	Setup
1	<code>Setup(e =&gt; e.Get(1))</code>
Any	<code>Setup(e =&gt; e.Get(It.IsAny&lt;int&gt;()))</code>
> 10	<code>Setup(e =&gt; e.Get(It.Is&lt;int&gt;(i =&gt; i &gt; 10))</code>
Range [0..10]	<code>Setup(e =&gt; e.Get(It.IsInRange&lt;int&gt;(0, 10, Range.Inclusive))</code>

An instance of a mock can have more than 1 setup for a given method provided we setup the method with different arguments.

**Tip:** Avoid using non specific values on capture, except when it's possible to validate what was provided.

### Mock - Setup - Returns

value	<code>Setup(...).Returns(value)</code>
access argument	<code>Setup(...).Returns(arg =&gt; arg + 10)</code>
Throw exception	<code>Setup(...).Throws&lt;Exception&gt;()</code>
Throw exception with message	<code>Setup(...).Throws(new Exception("My Message"))</code>
Lazy access	<code>Setup(...).Returns(() =&gt; value)</code>



By **Sérgio Ferreira**  
(AlienEngineer)

[cheatography.com/alienengineer/](https://cheatography.com/alienengineer/)

Published 26th August, 2019.  
Last updated 3rd September, 2019.  
Page 1 of 1.

Sponsored by **ApolloPad.com**  
Everyone has a novel in them. Finish Yours!  
<https://apollopad.com>