Stub - Just a replacement

```
// Create a stub
var mock = new Mock<I MyD epe nde ncy >();
// Get the Stub
var myDep = mock.O bject;
Assert.Th at( myDep .GetVa lue(), Is.Equ alT -
o(0));
```

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

Spy - What happened?

```
// Create a spy
var mock = new Mock<I MyD epe nde ncy >();
// Spy on executions
mock.V eri fy( myDep => myDep.G et Val ue());
```

Optional:

```
Verify can be called with optional argument that as
serts how many times the method as called: AtLeas t
Once, AtLeast, AtMost, AtMost Once, Betwee nEx clu -
sive, Betwee nIn clu sive, 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<I MyD epe nde ncy >();
string usedArg;
mock
       // Setup to capture any call to done to
DoSome thing method.
       .Se tup (myDep => myDep.D oS ome thi ng( -
It.I sA ny< Str ing >())
       // Callback function is called when DoSome -
thing was called.
       .Ca llb ack <St rin g>(arg => usedArg =
arg);

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

Assert.Th at( use dArg, Is.Equ alT o("t est "));
```

Mock - Do what I say!

Mock - Setup - Matching Arguments

Expected Argument	Setup
1	`Setup(e => e.Get(1))`
Any	`Setup(e => e.Get( It.I sA ny< int >())`
> 10	`Setup(e => e.Get( It.I s< int >(i => i > 10`
Range [0..10]	`Setup(e => e.Get( It.I sI nRa nge <in t>(0, lus ive))`

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	`Setup(...).R etu rns (value)`
access argument	`Setup(...).R etu rns(arg => arg + 10)`
Throw exception	`Setup(...).T hro ws< Exc ept ion >()`
Throw exception with message	`Setup(...).T hro ws(new Except ion ("My Mes e"))`
Lazy access	`Setup(...).R etu rns(() => value)`

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

mock
       // What behavior is going to be mocked
       .Se tup (myDep => myDep.G et Val ue())
       // What's going to be returned by
GetValue()
    .Returns(1);

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

Assert.Th at( myDep .GetVa lue(), Is.Equ alT -
o(1));
```



By **Sérgio Ferreira**
(AlienEngineer)

cheatography.com/alienengineer/

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

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