

Arrange-Act-Assert C# API

Changing Behavior	
Changing returned values	<pre>Isolate.WhenCalled(() => SomeClass.StaticMethod()) .WillReturn(10); SomeClass myInstance = new SomeClass(); Isolate.WhenCalled(() => myInstance.ReturnFive()) .WillReturn(10);</pre>
Throwing exceptions	<pre>Isolate.WhenCalled(() => myInstance.ReturnFive()) .WillThrow(new Exception());</pre>
Ignoring methods	<pre>Isolate.WhenCalled(() => myInstance.VoidMethod()) .IgnoreCall();</pre>
Changing Linq queries return values	<pre>Isolate.WhenCalled(() => from c in customerList select c).WillReturn(new List<Customer> { new Customer{ Id = 1, Name="Dave" }, new Customer{ Id = 2, Name="John" }, new Customer{ Id = 3, Name="Abe" } });</pre>
Custom return values	<pre>Isolate.WhenCalled(()=> myInstance.GetAgeOf("AnyName")).DoInstead((callContext)=> { string name = callContext.Parameters[0] as string; if(name == "John Smith") return 30; return -1; });</pre>
Conditional behavior	<pre>Isolate.WhenCalled(()=>myInstance.GetAgeOf ("John Smith")) .WithExactArguments().WillReturn(30);</pre>
Collections	<pre>Isolate.WhenCalled(() => SomeClass.Products) .WillReturnCollectionValuesOf(new[] { new ProductItem {Product = prodA, Quantity = 20}, new ProductItem {Product = prodB, Quantity = 40}, new ProductItem {Product = prodA, Quantity = 30} });</pre>
Replacing Ref and Out values	<pre>int refParamToReplace = 6; string outParamToReplace = "FakeName"; Isolate.WhenCalled(() => myInstance.RefOutMethod(ref refParamToReplace, out outParamToReplace)).WillReturn(10);</pre>
Non-public Members	<pre>Isolate.NonPublic.WhenCalled(myInstance, "PrivateMethod").WillReturn("Name"); Isolate.NonPublic.Property.WhenGetCalled(myInstance, "PrivateProp").WillReturn(5); Isolate.NonPublic.Property.WhenSetCalled(myInstance, "PrivateProp").IgnoreCall();</pre>

Creating Objects	
Fake object creation	<pre>var myFake = Isolate.Fake.Instance<SomeClass>(Members.ReturnRecursiveFakes); Members.CallOriginal // Default, returns fake objects. Members.ReturnNulls // Methods not faked, constructor called. Members.MustSpecifyReturnValues // Methods fail if called without setting behavior</pre>
Interfaces	<pre>var myFake = Isolate.Fake.Instance<ISomeInterface>(); // Also for abstract classes</pre>
Faking static methods	<pre>Isolate.Fake.StaticMethods <SomeClass>(); // Same arguments as Isolate.Fake.Instance.</pre>
Future objects	<pre>var myFake = Isolate.Swap.NextInstance<SomeClass>().WithRecursiveFake();</pre>
Faking dependencies	<pre>var myInstance = Isolate.Fake.Dependencies<SomeClass>(); var fakeDependency = Isolate.GetFake<ISomeInterface>(myInstance);</pre>

Call Verification	
Public methods	<pre>Isolate.Verify.WasCalledWithAnyArguments (()=>myInstance.ReturnFive()); Isolate.Verify.WasCalledWithExactArguments (()=>myInstance.GetCustomerByName ("John")); Isolate.Verify.WasNotCalled (()=>myInstance.ReturnFive());</pre>
Non public Methods	<pre>Isolate.Verify.NonPublic.WasCalled(myInstance, "PrivateMethod"); Isolate.Verify.NonPublic.Property .WasCalledGet(myInstance, "PrivateProp"); Isolate.Verify.NonPublic.Property .WasCalledSet(myInstance, "PrivateProp"); Isolate.Verify.NonPublic .WasCalled(myInstance, "GetCustomerByName").WithArguments("John");</pre>