2.13 Dynamic Fakes



This section will guide you to:

* Create a Windows Class library project for creating dynamic fakes in target classes to test.

**Development Environment**

* Windows 10
* Visual Studio 2019 Community Version

This guide has ten subsections, namely:

2.13.1 Creating a Windows Class Library Project for creating target classes to test

2.13.2 Adding a class with a dynamic method for testing purposes

2.13.3 Creating another Windows Class Library Project for running NUnit tests

2.13.4 Setting up NUnit as part of a Visual Studio project

2.13.5 Setting up NUnit3TestAdapter as part of a Visual Studio project

2.13.6 Setting up Moq as part of a Visual Studio project

2.13.7 Writing a test to test the dynamic method

2.13.8 Building the project

2.13.9 Running all the tests in Test Explorer

2.13.10 Pushing the code to your GitHub repositories

**Step 2.13.1:** Creating a Windows Class library project for creating target classes to test

* Open Visual Studio.
* From the top menu, click **File->New->Project**
* Select **(Class Library (.NET Framework)** from the displayed project types
* Click **Next**
* Input **Project Name** as Phase4Section2.5 and click **Create**
* This will create the Windows Class library project

**Step 2.13.2:** Adding a class with a dynamic method for testing purposes.

* From the **Solution Explorer,** right click **Phase4Section2.5** and click **Add->Class**
* Input name as DynamicCalc.cs and click **Add**
* Add the following code:

**using** System;

**using** System.Collections.Generic;

**using** System.Linq;

**using** System.Text;

**using** System.Threading.Tasks;

**namespace** Phase4Section2.\_5

{

**public** **interface** IDynamicCalc

{

dynamic add(Object x, Object y);

}

**public** **class** DynamicCalc:IDynamicCalc

{

**public** dynamic add(Object x, Object y)

{

**if** (x.GetType().Name == "Int32" && y.GetType().Name == "Int32")

**return** Convert.ToInt32((Int32)x + (Int32)y);

**else**

**return** Convert.ToInt32(0);

}

}

}

**Step 2.13.3:** Creating a Windows Class library project for running NUnit tests

* In **Solution Explorer,** right click the Solution item and click **Add->New Project**
* Select **(Class Library (.NET Framework)** from the displayed project types
* Click **Next**
* Input **Project Name** as Phase4Section2.5.Tests and click **Create**
* This will create the Windows Class library project for using NUnit

**Step 2.13.4:** Setting up NUnit as part of the project

* From the **Solution Explorer,** right click **Phase4Section2.5.Tests** and click **Manage Nuget Packages**
* Click on **Browse** tab and search for NUnit
* Click on the NUnit item and click **Install**
* This will set up NUnit for the project

**Step 2.13.5:** Setting up NUnit3TestAdapter as part of the project

* From the **Solution Explorer,** right click on **Phase4Section2.3** and click **Manage Nuget Packages**
* Click on **Browse** tab and search for NUnit3TestAdapter
* Click on the NUnit3TestAdapter item and click **Install**

**Step 2.13.6:** Setting up Moq as part of the project

* From the **Solution Explorer,** right click **Phase4Section2.5.Tests** and click **Manage Nuget Packages**
* Click on **Browse** tab and search for Moq
* Click on the Moq item and click **Install**
* This will set up Moq for the project

**Step 2.13.7:** Writing a test to test the dynamic method

* From the **Solution Explorer,** expand **Phase4Section2.5.Tests** and double click **Class1.cs**
* Add the following code:

**using** System;

**using** System.Collections.Generic;

**using** System.IO;

**using** System.Linq;

**using** System.Text;

**using** System.Threading.Tasks;

**using** Moq;

**using** NUnit.Framework;

**namespace** Phase4Section2.\_5.Tests

{

[TestFixture]

**public** **class** Class1

{

[Test]

**public** **void** DynamicFake()

{

**int** x = 10, y = 20;

Mock<IDynamicCalc> mockCalc = **new** Mock<IDynamicCalc>();

**var** result = **new**

{

V = Convert.ToInt32(x + y)

};

mockCalc.Setup(c => c.add(It.IsAny<**object**>(), It.IsAny<**object**>())).Returns(result);

}

}

}

**Step 2.13.8:** Building the project

* From the top menu, choose **Build->Build Solution**
* If any compilation errors are shown, fix them as required.

**Step 2.13.9:** Running all the tests in Test Explorer

* From the top menu, choose **Test->Windows->Test Explorer**
* In Test Explorer, click on **Run All**
* This will execute the tests and show the results in Test Explorer

**Step 2.13.10:** Pushing the code to your GitHub repositories

Open your command prompt and navigate to the folder where you have created your files.

cd <folder path>

Initialize your repository using the following command:

git init

Add all the files to your git repository using the following command:

git add .

Commit the changes using the following command:

git commit -m “Changes have been committed.”

Push the files to the folder you created initially using the following command:

git push -u origin master