2.7 Data-Driven Testing



This section will guide you to:

* Create a Windows Class library project for adding test fixture to show data-driven testing

**Development Environment**

* Windows 10
* Visual Studio 2019 Community Version

This guide has ten subsections, namely:

2.7.1 Creating a Windows class library project for creating target classes to test

2.7.2 Creating a Windows class library project for running NUnit tests

2.7.3 Setting up NUnit as part of a Visual Studio project

2.7.4 Setting up NUnit3TestAdapter as part of a Visual Studio project

2.7.5 Setting up Moq as part of a Visual Studio project

2.7.6 Creating a data file for generating test data

2.7.7 Adding Test Fixture for showing Data-Driven Testing

2.7.8 Building the project

2.7.9 Running all the tests in Test Explorer

2.7.10 Pushing the code to your GitHub repositories

**Step 2.7.1:** 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**
* Name the **Project Name** as Phase4Section2.5 and click **Create**

**Step 2.7.2:** 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**
* Name the **Project Name** as Phase4Section2.5.Tests and click **Create**

**Step 2.7.3:** 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**

**Step 2.7.4:** 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 2.7.5:** 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**

**Step 2.7.6:** Creating a data file for generating test data

* From the **Solution Explorer,** right click **Phase4Section2.5.Tests** and click **Add->New Item**
* Choose **Text File** from list of file types. Add **Name** as testdata.txt and click **Add**
* Enter the following data to the file:

10,20,300

100,200,300

1000,2000,3000

* Copy this file to the root of C drive **c:\testdata.txt**

**Step 2.7.7:** Adding Test Fixture for showing Data-Driven Testing

* 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]

[TestCase(10, 20, ExpectedResult = 30)]

[TestCase(100, 200, ExpectedResult = 300)]

[TestCase(1000, 2000, ExpectedResult = 3000)]

**public** **int** DataDriven1(**int** x, **int** y)

{

**var** calc = **new** Calculator();

**return** calc.add(x, y);

}

**public** **static** List<TestCaseData> TestCases

{

**get**

{

**var** testCases = **new** List<TestCaseData>();

**using** (**var** fs = File.OpenRead("c:\\testdata.txt"))

**using** (**var** sr = **new** StreamReader(fs))

{

**string** line = **string**.Empty;

**while** (line != **null**)

{

line = sr.ReadLine();

**if** (line != **null**)

{

**string**[] split = line.Split(**new** **char**[] {','},

StringSplitOptions.None);

**int** a = Convert.ToInt32(split[0]);

**int** b = Convert.ToInt32(split[1]);

**int** answer = Convert.ToInt32(split[2]);

**var** testCase = **new** TestCaseData(a, b).Returns(answer);

testCases.Add(testCase);

}

}

}

**return** testCases;

}

}

[Test]

[TestCaseSource("TestCases")]

**public** **int** DataDriven2(**int** x, **int** y)

{

**var** calc = **new** Calculator();

**return** calc.add(x, y);

}

}

}

* From the **Solution Explorer,** expand **Phase4Section2.5.Tests** and right click **Add->New Class** name it as **Calculator.cs**
* Add the following code in **Calculator.cs**:

**using** System;

**using** System.Collections.Generic;

**using** System.Linq;

**using** System.Text;

**using** System.Threading.Tasks;

**namespace** Phase4Section2.\_5

{

**public** **interface** ICalculator

{

**int** add(**int** x, **int** y);

**int** addStrings(**string** x, **string** y);

}

**public** **class** Calculator:ICalculator

{

**public** **int** add(**int** x, **int** y)

{

**return** x + y;

}

**public** **int** addStrings(**string** x, **string** y)

{

**int** a = 0, b = 0;

Int32.TryParse(x, **out** a);

Int32.TryParse(y, **out** b);

**if** (a == 0 || b == 0)

**throw** **new** InvalidOperationException("String values are not valid integers");

**return** a + b;

}

}

**Step 2.7.8:** Building the project

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

**Step 2.7.9:** 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 2.7.10:** 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