2.6 Exceptions



This section will guide you to:

* Create a Windows Class library project for adding a test fixture to show exceptions

**Development Environment**

* Windows 10
* Visual Studio 2019 Community Version

This guide has nine subsections, namely:

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

2.6.2 Creating a Windows class library project for running NUnit tests

2.6.3 Setting up NUnit as part of a Visual Studio project

2.6.4 Setting up NUnit3TestAdapter as part of a Visual Studio project

2.6.5 Setting up Moq as part of a Visual Studio project

2.6.6 Adding Test Fixture to show Exceptions

2.6.7 Building the project

2.6.8 Running all the tests in Test Explorer

2.6.9 Pushing the code to your GitHub repositories

**Step 2.6.1:** 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 2.6.2:** Creating another 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 2.6.3:** 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 2.6.4:** 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 2.6.5:** 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 2.6.6:** Adding Test Fixture to show Exceptions

* 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** Exceptions()

{

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

Assert.Throws<InvalidOperationException>(() => calc.addStrings("aaa", "Bbb"));

}

}

}

* 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 2.6.7:** Building the project

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

**Step 2.6.8:** 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 2.6.9:** 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