For NUnit Testing , we need two packages

**NUnit, NUnit3 Test Adapater**

<Project Sdk="Microsoft.NET.Sdk">

<PropertyGroup>

<TargetFramework>netcoreapp3.1</TargetFramework>

</PropertyGroup>

<ItemGroup>

<PackageReference Include="Microsoft.NET.Test.Sdk" Version="15.9.2" />

<PackageReference Include="NUnit" Version="3.13.2" />

<PackageReference Include="NUnit3TestAdapter" Version="4.1.0" />

</ItemGroup>

<ItemGroup>

<ProjectReference Include="..\DemoProject\DemoProject.csproj" />

</ItemGroup>

</Project>

**NUnit** is a unit-**testing framework**

The **NUnit Test Adapter** allows you to run **NUnit tests** inside Visual Studio. If you do not add this component in your project you will not be able to find your **tests** in **test** Explorer. You need to install both the Libraries in the project where you are writing the **test** methods only.

Code Coverage : That our test cases are enough or not

using NUnit.Framework;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace NUnitTestingDemo.Tests

{

[TestFixture]

public class NumericFunctionsTestClass

{

[Test]

public void addTest()

{

// Arrange

NUnitTestingDemo.NumericFunctions obj = new NUnitTestingDemo.NumericFunctions();

// Act

int res = obj.add(10, 30);

// Assert

Assert.AreEqual(40, res);

}

}

}

using NUnit.Framework;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace NUnitTestingDemo.Tests

{

[TestFixture]

public class NumericFunctionsTestClass

{

[Test]

public void addTestFor40()

{

// Arrange

NUnitTestingDemo.NumericFunctions obj = new NUnitTestingDemo.NumericFunctions();

// Act

int res = obj.add(10, 30);

// Assert

Assert.AreEqual(40, res);

}

[Test]

public void addTestFor60()

{

// Arrange

NUnitTestingDemo.NumericFunctions obj = new NUnitTestingDemo.NumericFunctions();

// Act

int res = obj.add(30, 30);

// Assert

Assert.AreEqual(60, res);

}

}

}

But anyone can write in a Test case

Assert(true,true) > It will always give you success , how to check that we have written enough test cases

using NUnit.Framework;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace NUnitTestingDemo.Tests

{

[TestFixture]

public class NumericFunctionsTestClass

{

[Test]

public void addTestFor40()

{

// Arrange

NUnitTestingDemo.NumericFunctions obj = new NUnitTestingDemo.NumericFunctions();

// Act

int res = obj.add(10, 30);

// Assert

Assert.AreEqual(40, res);

}

[Test]

public void addTestFor60()

{

// Arrange

NUnitTestingDemo.NumericFunctions obj = new NUnitTestingDemo.NumericFunctions();

// Act

int res = obj.add(30, 30);

// Assert

Assert.AreEqual(60, res);

}

[Test]

public void CheckForHR()

{

NUnitTestingDemo.NumericFunctions obj = new NumericFunctions();

Assert.AreEqual(20,obj.GetLeaves("HR"));

}

}

}

How to provide multiple values & expected output together to the test case

[TestCase(10,20, 30)]

[TestCase(40, 20, 60)]

[TestCase(60, 20, 80)]

[TestCase(20, 20, 30)]

[TestCase(20, 20, 40)]

public void addTest(int x, int y, int expectedOutput)

{

// Arrange

NUnitTestingDemo.NumericFunctions obj = new NUnitTestingDemo.NumericFunctions();

// Act

int res= expectedOutput = obj.add(x, y);

// Assert

Assert.AreEqual(res, expectedOutput);

}

Code Coverage is an Enterprise Edition feature, not available in Professional.

