Testing using NUnit in C#

Overview

In this tutorial, we will create an NUnit Test class with multiple Test Cases, implemented in C#.

Description

The test case validates a Calculator class and exploits some NUnit features such as the ability to validate the same Test against multiple input values, and the possibility of linking Tests with requirements in Jira using NUnit's Test attributes.

Calculator.cs

```
namespace x
{
public class Calculator
{
// Square function
public static int Square(int num)
{
    return num*num;
}
// Add two integers and returns the sum
public static int Add(int num1, int num2 )
{
    return num1 + num2;
}
// Add two integers and returns the sum
public static double Add(double num1, double num2 )
{
    return num1 + num2;
}
// Multiply two integers and retuns the result
public static int Multiply(int num1, int num2 )
{
    return num1 * num2;
}
public static int Divide(int num1, int num2 )
{
    return num1 / num2;
}
// Subtracts small number from big number
public static int Subtract(int num1, int num2 )
{
    if ( num1 > num2 )
    {
    return num1 - num2;
    }
    return num2 - num1;
    }
}
}
```

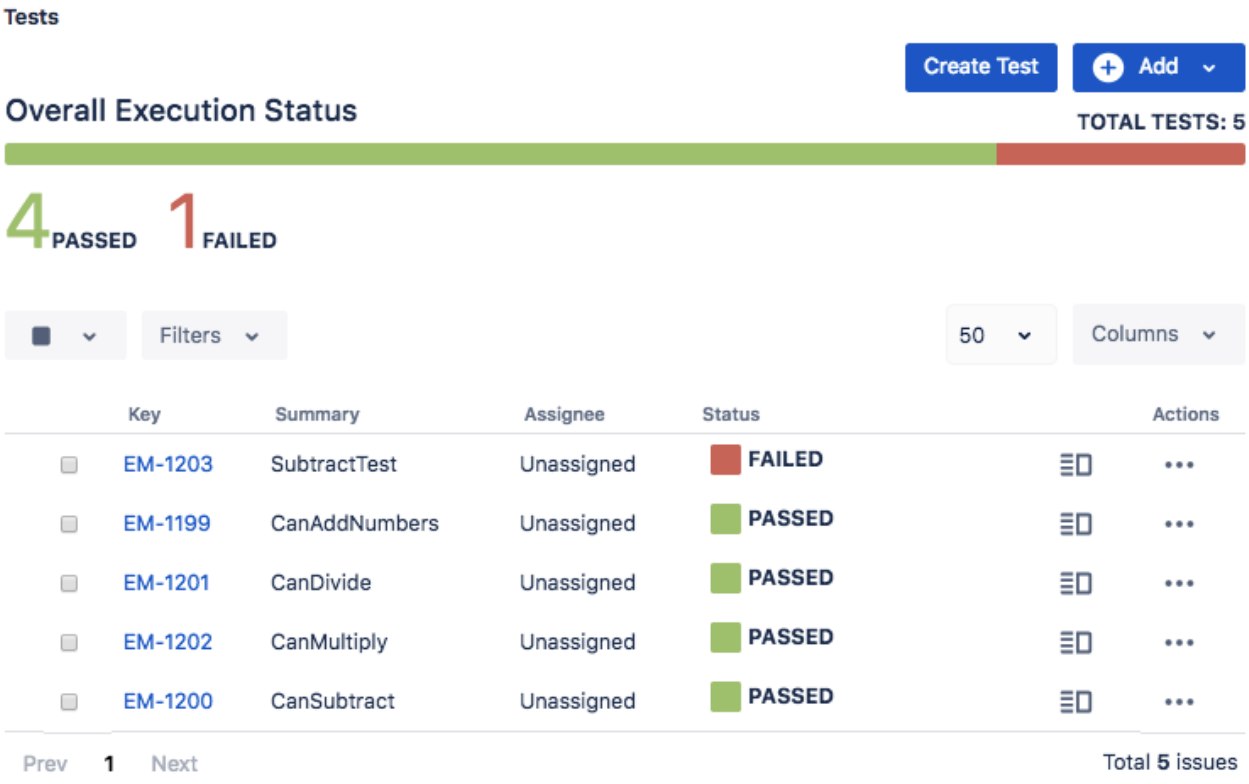
CalculatorTest.cs

```
using NUnit.Framework;
namespace x
{
    [TestFixture]
    public class CalculatorTests
    {
        [Test, Property("Requirement", "CALC-1")]
        [TestCase(1, 1, 2)]
        [TestCase(-1, -1, -2)]
        [TestCase(100, 5, 105)]
        public void CanAddNumbers(int a, int b, int expected)
        {
            Assert.That(Calculator.Add(a, b), Is.EqualTo(expected));
        }

        [TestCase(1, 1, 0)]
        [TestCase(-1, -1, 0)]
        [TestCase(100, 5, 95)]
        public void CanSubtract(int x, int y, int expected)
        {
            Assert.That(Calculator.Subtract(x, y), Is.EqualTo(expected));
        }

        [TestCase(1, 1, 1)]
        [TestCase(-1, -1, 1)]
        [TestCase(100, 5, 500)]
        public void CanMultiply(int x, int y, int expected)
        {
            Assert.That(Calculator.Multiply(x, y), Is.EqualTo(expected));
        }
        [TestCase(1, 1, 1)]
        [TestCase(-1, -1, 1)]
        [TestCase(100, 5, 20)]
        public void CanDivide(int x, int y, int expected)
        {
            Assert.That(Calculator.Divide(x, y), Is.EqualTo(expected));
        }

    }
}
```

project.json

```
{
  "version": "1.0.0-*",
  "buildOptions": {
    "debugType": "portable",
    "emitEntryPoint": true
  },
  "dependencies": {
    "NUnit": "3.5.0",
    "dotnet-test-nunit": "3.4.0-beta-2"
  },
  "testRunner": "nunit",
  "frameworks": {
    "netcoreapp1.1": {
      "dependencies": {
        "Microsoft.NETCore.App": {
          "type": "platform",
          "version": "1.1.0"
        }
      },
      "imports": "dnxcore50"
    }
  }
}
```

After successfully running the Test Case and generating the NUnit XML report (e.g., TestResult.xml), it can be imported to Xray (by using either the REST API or the **Import Execution Results** action within the Test Execution).



NUnit's Test Case is mapped to a Generic Test in Jira, and the **Generic Test Definition** field contains the namespace, the name of the class, and the method name that implements the Test case.

The Execution Details of the Generic Test contains information about the context, which in this case corresponds to the Test case method, along with the different input values that were validated.

Test Details

Test Type:	Generic
Definition:	x.CalculatorTests.CanAddNumbers

Results

Context	Error Message	Duration	Status
TestCase 1005 - CanAddNumbers(1,1,2)	-	1 millisec	PASSED
TestCase 1007 - CanAddNumbers(100,5,105)	-	0 millisec	PASSED
TestCase 1006 - CanAddNumbers(-1,-1,-2)	-	0 millisec	PASSED

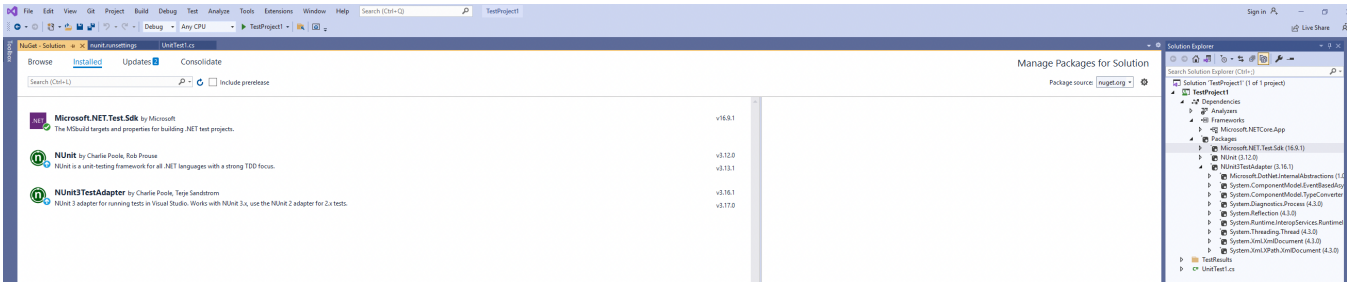
The Test "CanAddNumbers" was automatically linked to the sum requirement (i.e., the user story "CALC-1").

Tips

If you're using Visual Studio as your IDE, you need to have some dependencies/packages installed.

- NUnit
- NUnit3TestAdapter

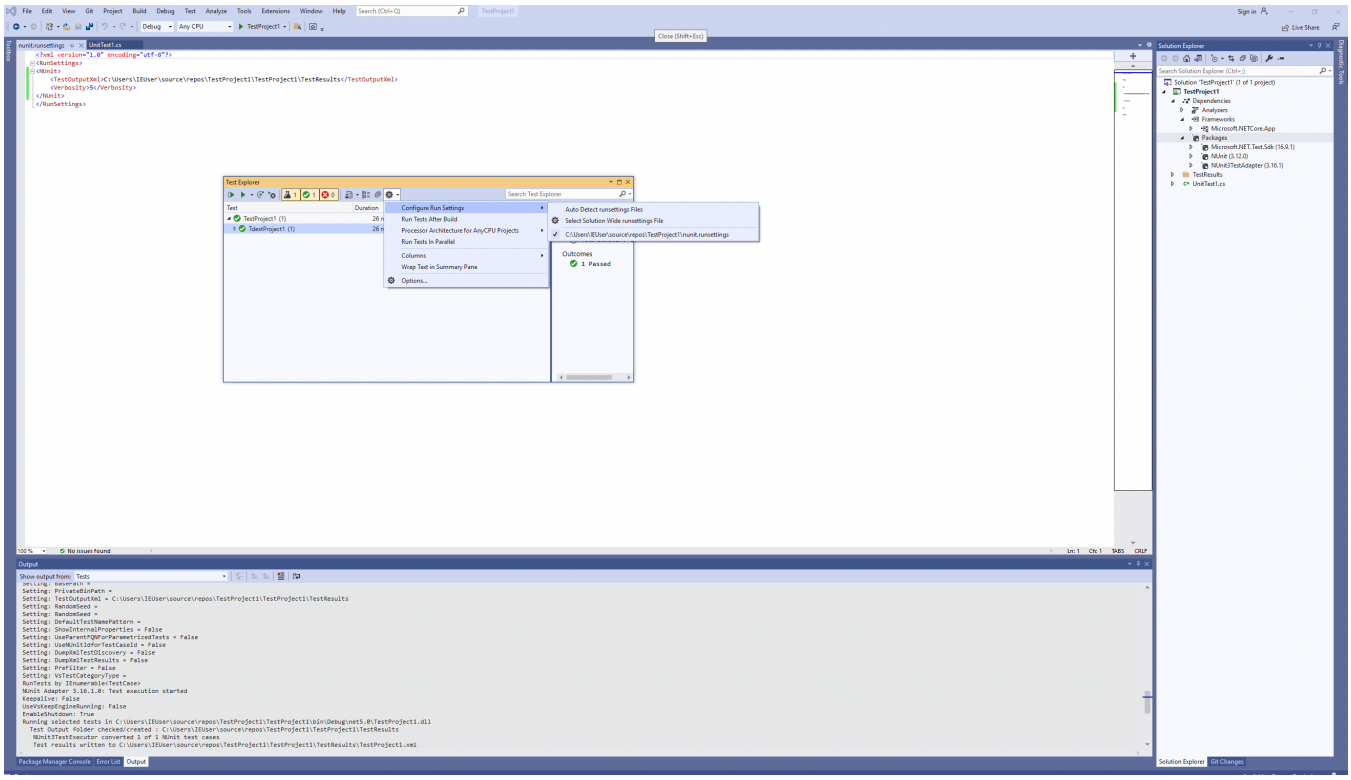
These can be installed from **Tools>NuGet Package Manager** (using the console or the manager's UI).



Then you can configure the Test Explorer to run the NUnit tests while at the same time producing a NUnit XML report.

nunit.runsettings

```
<?xml version="1.0" encoding="utf-8"?>
<RunSettings>
   <NUnit>
     <TestOutputXml>C:\TestResults</TestOutputXml>
 </NUnit>
</RunSettings>
```



References

- https://github.com/nunit/docs/wiki
- Configure unit tests by using a .runsettings file