

# Testing using Specflow and NUnit in C#

## Overview

In this tutorial, we will create a simple Specflow test in C#, using NUnit as the test runner.



### Notes

Although this tutorial explores a way of managing Specflow tests in Jira, it does not take advantage of Xray's Cucumber features.

Therefore, in this case, Jira isn't used to make the BDD specification; only to abstract the Tests. The tests in Jira will be created as Generic Tests, not Cucumber Tests. Since the semantics of Cucumber Tests is lost, so do the Scenario Outline examples-related results.

We suggest you to have a look at this tutorial instead: [Testing using SpecFlow and Gherkin scenarios in C#](#).

## Description

Specflow is a tool used for BDD in C#.

In this example, 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 also the possibility of linking Tests with requirements in Jira by using Test attributes.

### Calculator.feature

```
Feature: Calculator
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the arithmetic operation of two numbers
@mytag
Scenario: Add two numbers
  Given I have entered 50 into the calculator
  And I have also entered 70 into the calculator
  When I press add
  Then the result should be 120 on the screen

Scenario: Multiply two numbers
  Given I have entered 2 into the calculator
  And I have also entered 3 into the calculator
  When I press multiply
  Then the result should be 6 on the screen

Scenario Outline: Amazing addition of two numbers
  Given I have entered <input_1> into the calculator
  And I have also entered <input_2> into the calculator
  When I press add
  Then the result should be <output> on the screen
  Examples:
    | input_1 | input_2 | output |
    | 20      | 30      | 50     |
    | 30      | 50      | 80     |
```

### CalculatorSteps.cs

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TechTalk.SpecFlow;

[Binding]
public sealed class CalculatorSteps
{
    private int result { get; set; }
    private Calculator calculator = new Calculator();
    [Given(@"I have entered (.*) into the calculator")]
    public void GivenIHaveEnteredIntoTheCalculator(int number)
    {
        calculator.FirstNumber = number;
    }
    [Given(@"I have also entered (.*) into the calculator")]
    public void GivenIHaveAlsoEnteredIntoTheCalculator(int number)
    {
        calculator.SecondNumber = number;
    }
    [When(@"I press add")]
    public void WhenIPressAdd()
    {
        result = calculator.Add();
    }
    [When(@"I press multiply")]
    public void WhenIPressMultiply()
    {
        result = calculator.Multiply();
    }
    [Then(@"the result should be (.*) on the screen")]
    public void ThenTheResultShouldBeOnTheScreen(int expectedResult)
    {
        Assert.AreEqual(expectedResult, result);
    }
}
```

### packages.config

```
<?xml version="1.0" encoding="utf-8"?>
<packages>
  <package id="NUnit" version="3.6.0" targetFramework="net452" />
  <package id="NUnit.Console" version="3.6.0" targetFramework="net452" />
  <package id="NUnit.ConsoleRunner" version="3.6.0" targetFramework="net452" />
  <package id="NUnit.Extension.NUnitProjectLoader" version="3.5.0" targetFramework="net452" />
  <package id="NUnit.Extension.NUnitV2Driver" version="3.6.0" targetFramework="net452" />
  <package id="NUnit.Extension.NUnitV2ResultWriter" version="3.5.0" targetFramework="net452" />
  <package id="NUnit.Extension.TeamCityEventListener" version="1.0.2" targetFramework="net452" />
  <package id="NUnit.Extension.VSProjectLoader" version="3.5.0" targetFramework="net452" />
  <package id="NUnit3TestAdapter" version="3.6.0" targetFramework="net452" />
  <package id="Shouldly" version="2.8.2" targetFramework="net452" />
  <package id="SpecFlow" version="2.1.0" targetFramework="net452" />
  <package id="SpecFlow.NUnit" version="2.1.0" targetFramework="net452" />
</packages>
```

After successfully running the Scenarios and generating the NUnit XML report (e.g., [TestResult.xml](#)), it can be imported to Xray via the REST API or the **Import Execution Results** action within the Test Execution.

```
nunit3-console bin\Debug\UnitTestProject2.dll
```

```
curl -H "Content-Type: multipart/form-data" -u admin:admin -F "file=@TestResult.xml" "http://localhost:8080/rest/raven/1.0/import/execution/nunit?projectKey=CALC"
```

### Overall Execution Status

Do you want to discuss?

[Connect](#) [Dismiss](#)

3 PASS

TOTAL TESTS: 3

#### FILTERS

Test Set	Assignee	Status	Component	Search
All	All			Contains text <input type="button" value="Clear"/>



Show 10 entries

Columns

Key	Summary	Test Type	#Req	#Def	Test Sets	Assignee	Status
1	CALC-146 AddTwoNumbers	Generic	0	0		lime@rootfest.net	PASS
2	CALC-148 AmazingAdditionOfTwoNumbers	Generic	0	0		lime@rootfest.net	PASS
3	CALC-147 MultiplyTwoNumbers	Generic	0	0		lime@rootfest.net	PASS

NUnit's Test Case is mapped to a Generic Test in Jira, and the **Generic Test Definition** field contains 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 Test Suite, which in this case corresponds to the Test Case class.

## Execution Details

#### Test Description

Test Details	
Test Type:	Generic
Definition:	UnitTestProject2.CalculatorFeature.AmazingAdditionOfTwoNumbers

Results			
Context	Error Message	Duration	Status
TestCase 0-1003 - AmazingAdditionOfTwoNumbers("20","30","50",System.String[])		0 millisc	PASS
TestCase 0-1004 - AmazingAdditionOfTwoNumbers("30","50","80",System.String[])		0 millisc	PASS

## References

- <http://specflow.org/docs/>

- <https://github.com/techtalk/SpecFlow/wiki/Reporting>