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 wil 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 (by using either the REST API or the **Import Execution Results** action within the Test Execution).

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

Once you have the report file available you can upload it to Xray through a request to the REST API endpoint for NUnit, and for that the first step is to follow the instructions in v1 or v2 (depending on your usage) to obtain the token we will be using in the subsequent requests.

Authentication

The request made will look like:

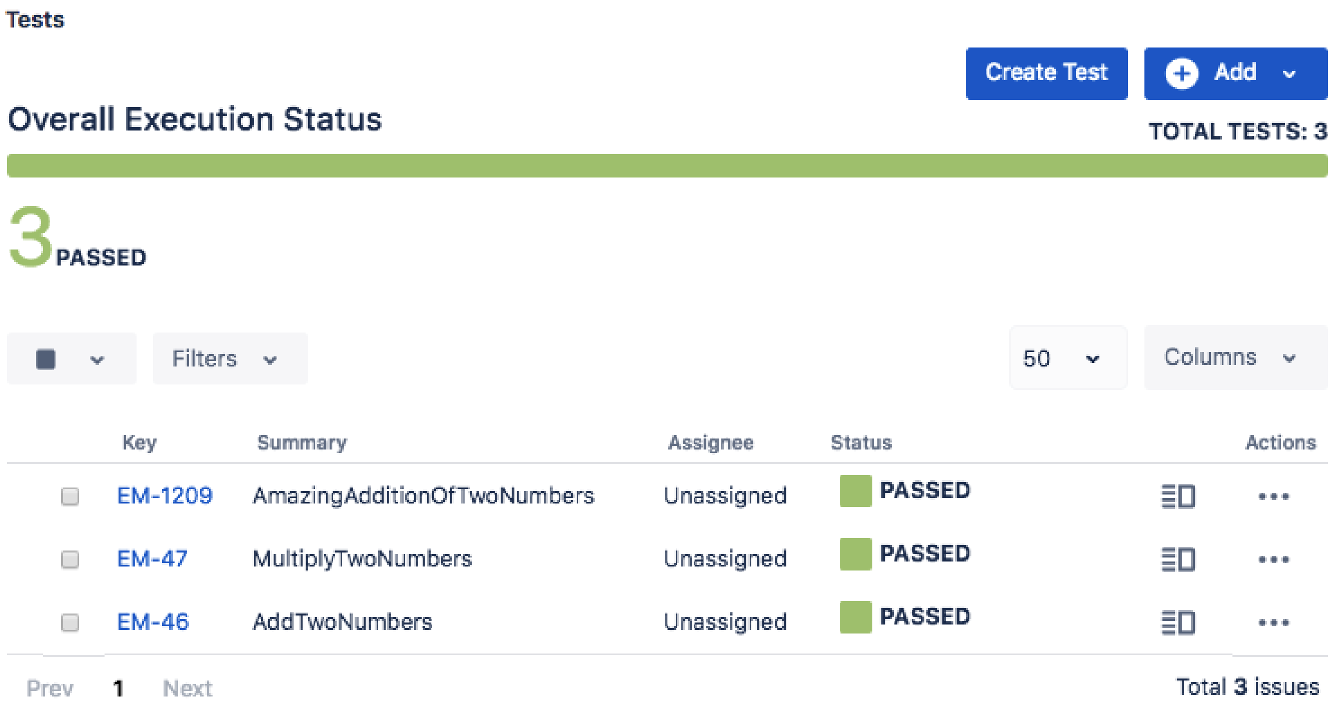
```
curl -H "Content-Type: application/json" -X POST --data '{ "client_id": "CLIENTID","client_secret":
"CLIENTSECRET" }'  https://xray.cloud.getxray.app/api/v1/authenticate
```

The response of this request will return the token to be used in the subsequent requests for authentication purposes.

NUnit XML results

Once you have the token we will use it in the API request with the definition of some common fields on the Test Execution, such as the target project, project version, etc.

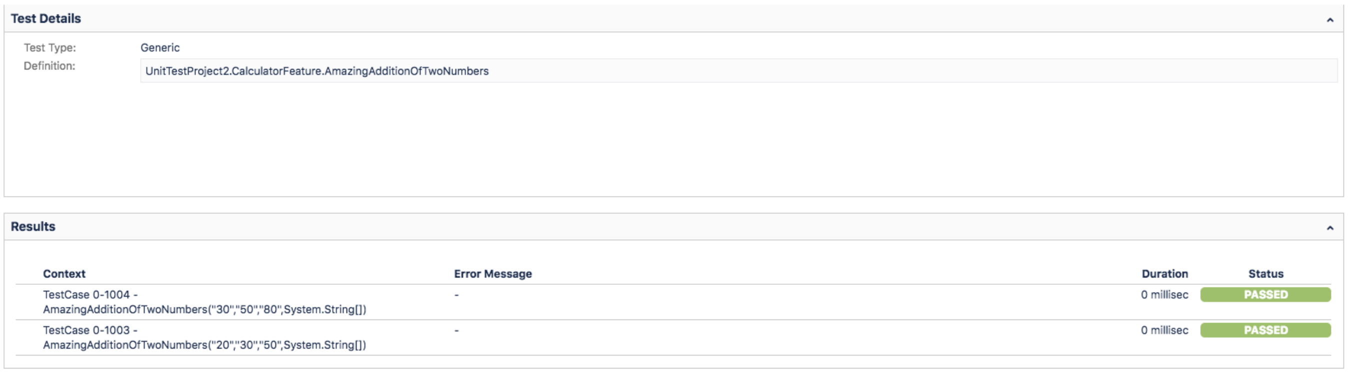
```
curl -H "Content-Type: text/xml" -X POST -H "Authorization: Bearer $token"  --data @"TestResult.
xml" https://xray.cloud.getxray.app/api/v2/import/execution/nunit?projectKey=CALC
```



Key	Summary	Assignee	Status	Actions
EM-1209	AmazingAdditionOfTwoNumbers	Unassigned	PASSED	
EM-47	MultiplyTwoNumbers	Unassigned	PASSED	
EM-46	AddTwoNumbers	Unassigned	PASSED	

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.


Test Details

Test Type: Generic

Definition: UnitTestProject2.CalculatorFeature.AmazingAdditionOfTwoNumbers

Results

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

References

- http://specflow.org/docs/
- https://github.com/techtalk/SpecFlow/wiki/Reporting