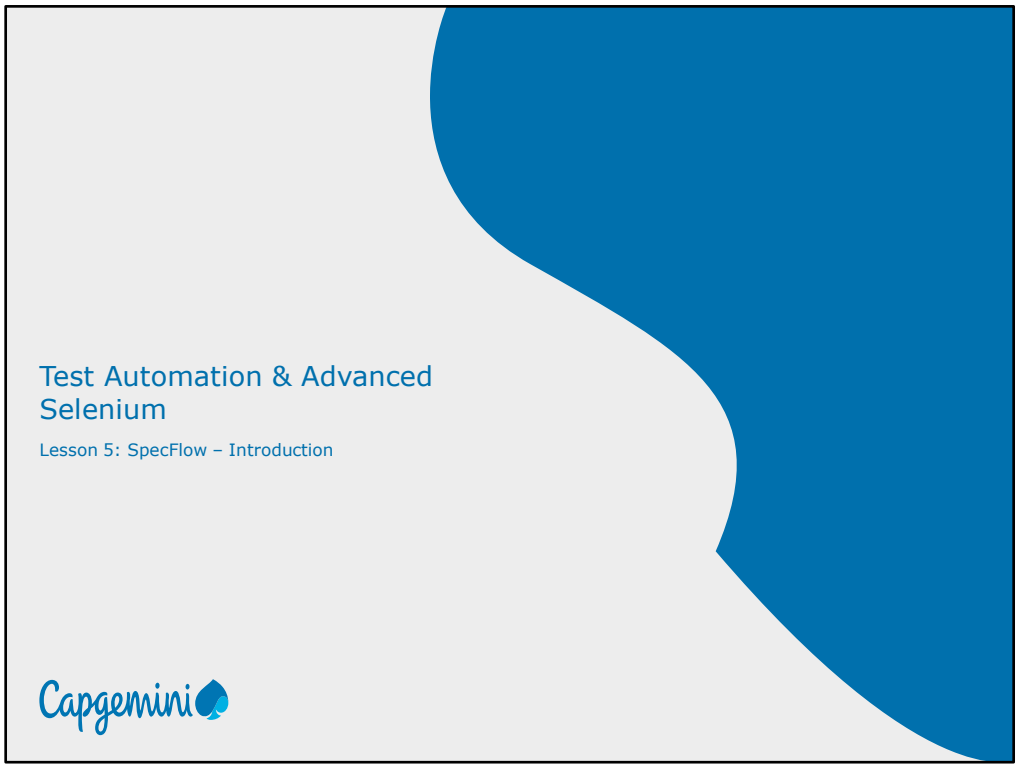

Test Automation & Advanced Selenium
Lesson 5: SpecFlow – Introduction
Capgemini

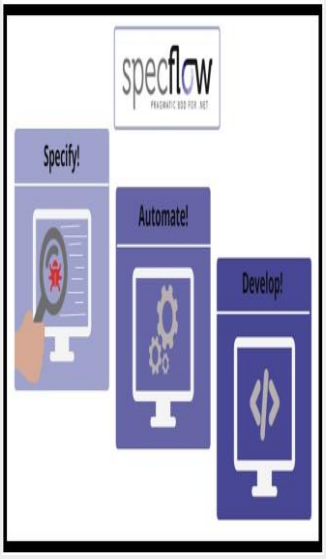
Lesson Objectives

- SpecFlow Framework
- How it Works
- Advantages of SpecFlow
- Feature File
- Steps Definitions
- Scenario Outline
- Data Tables in SpecFlow

5.1: • SpecFlow Framework

- SpecFlow, also known as Cucumber for .NET is an open-source tool used for automating .NET applications. *NUnit* does not create any test scripts by itself.
- It is a Behaviour Driven Development or BDD tool.
- These tools are bridges between the business language and technical language.
- Using these tools, the test-cases for automated software testing with SpecFlow are written in English-like language understandable by all the stakeholders of the project, developers and non-developers alike.

5.2: **SpecFlow – How it Works**



- SpecFlow can be integrated with a number of IDEs including Visual Studio, SharpDevelop and MonoDevelop to roll out automated testing.
- It uses gherkin language for writing the test scenarios using keywords Given, When and Then. The test scripts can be written in any popular language like C#,Visual Basic .NET, Boo, F#, IronPython and IronRuby.

- *A feature file created for writing the test scenarios. This file contains the name of the feature to be tested, test scenarios and when to execute the test scripts.*
- *The test scripts are prepared in a class based on the test scenarios created. Later these test scripts can be executed as unit tests using NUnit or MTest.*
- *The result of the tests run is available. The tests can be passed, failed or inconclusive.*

5.3 Advantages of SpecFlow

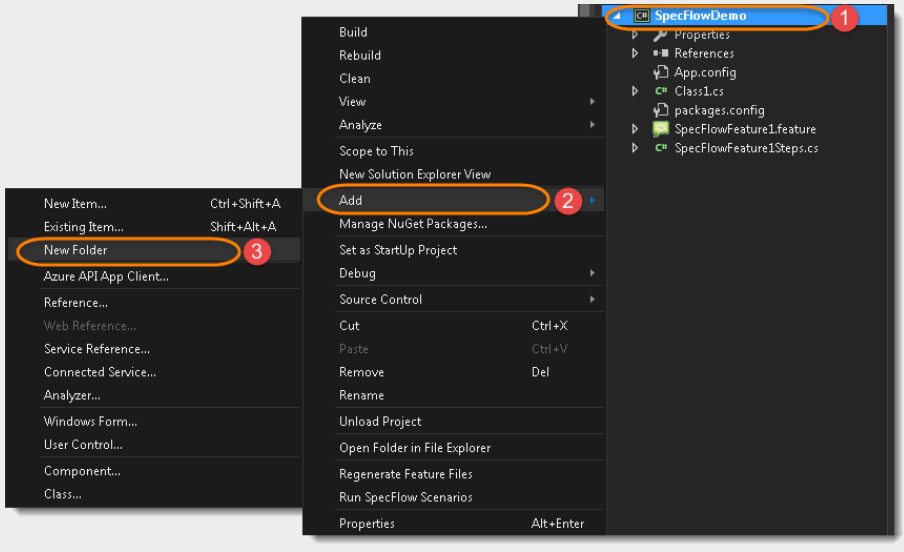
- Tests can be prioritized using SpecFlow while Cucumber does not provide the functionality of prioritizing tests. This can be beneficial if only some of the scenarios are to be tested in one run.
- In SpecFlow, parameters passed in the tests can be mapped to the text instructions avoiding the need to the sequence of parameters in text. It can be mapped irrespective of the sequence. This feature is not supported in Cucumber.
- Scoping functionality is available in Specflow which is not present in Cucumber. By defining the Scope, the same test definition can be used for executing multiple test scripts for automated software testing with SpecFlow. For example, if UI and functional test scripts are to be executed for the same test scenario, then in Specflow there is no need to create two different scenarios. Tests can be executed by using scope. While in Cucumber, for every test script there must be a different scenario.
- When test scripts are being written for a very large project, there are hundreds of test scenarios. Creating test scripts for these scenarios and remembering them while writing becomes cumbersome. SpecFlow provides a feature named Auto-Complete, with Visual Studio integration, using which the phrases can be automatically completed in test scripts. Only by typing a single word, a phrase can be selected.
- Cucumber does not provide any such feature and this is one major drawback for this tool for automated software testing.
- SpecFlow very naturally integrates with Visual Studio, and new feature files are created very easily by File/New templates.
- SpecFlow gives complete Visual Studio debugger support, allowing setting of breakpoints on Given/When/Then lines in the feature files and step through their execution.
- All step definitions are implemented in any .NET language.
- When a project containing SpecFlow feature files is compiled, the output is a NUnit test assembly. Any NUnit-compatible test runner or existing CI infrastructure can be then used to execute the test specifications with no additional configuration to conduct the automated software testing.

5.3 **Feature File**

- A feature file is an entry point to the SpecFlow test. This is a file where you will describe your tests in Descriptive language (Like English).
- It is an essential part of SpecFlow, as it serves as an automation test script as well as live documents.
- A feature file can contain a scenario or can contain many scenarios in a single feature file but it usually contains a list of scenarios.

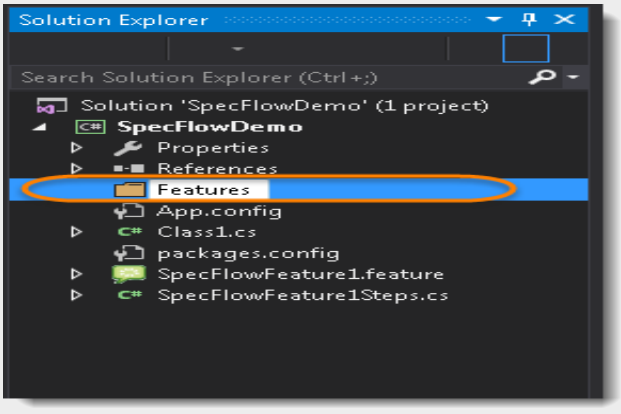
Create a Feature File Folder

- Create a new Folder by right click on the 'Project' and navigating to Add -> New Folder.



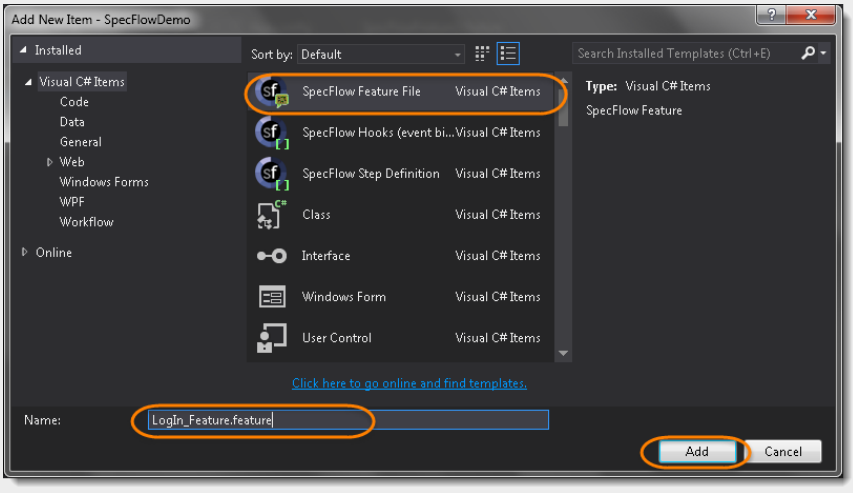
Create a Feature File Folder(Cont…)

Name the folder as '*Features*' and hit enter. Once the folder for feature file is created, we are good to go to create a feature file.



Create a Feature File Folder(Cont...)

- On the Feature folder Right-click and navigate to Add -> New Item...
- Select SpecFlow Feature File in the middle and give it a logical name, for the sake of this tutorial, please use the same name 'LogIn_Feature' referred in the below screenshot.



Create a Feature File Folder(Cont…)

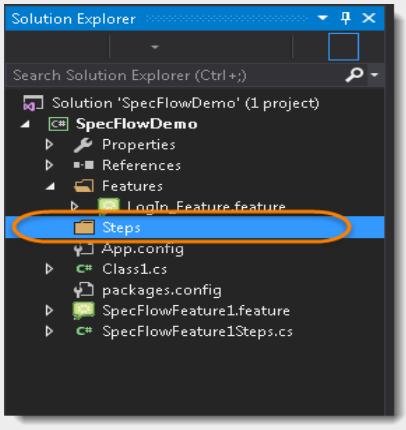
```
Feature: LogIn_Feature
        In order to access my account
    As a user of the website
    I want to log into the website

@mytag
Scenario: Successful Login with Valid Credentials
        Given User is at the Home Page
        And Navigate to LogIn Page
        When User enter UserName and Password
        And Click on the LogIn button
        Then Successful LogIN message should display

Scenario: Successful LogOut
        When User LogOut from the Application
        Then Successful LogOut message should display
```

5.5 Steps Definitions

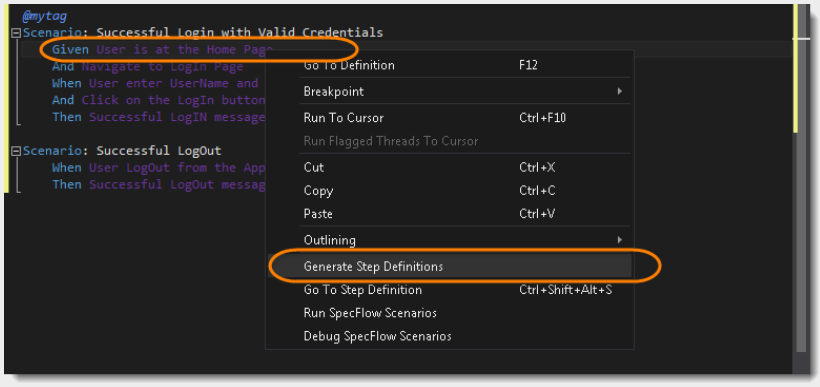
- A Step Definition file is a small piece of code with a pattern attached to it or in other words a Step Definition is a C# method in a class with an annotation above it.
- An annotation followed by the pattern is used to link the Step Definition to all the matching Steps, and the code is what SpecFlow will execute when it sees a Gherkin Keywords.
- Create a new Folder by right click on the 'Project' and navigating to Add -> New Folder.
- Name the folder as 'Steps' and hit enter.



5.5 Steps Definitions(Cont…)

Create a Step Definition File

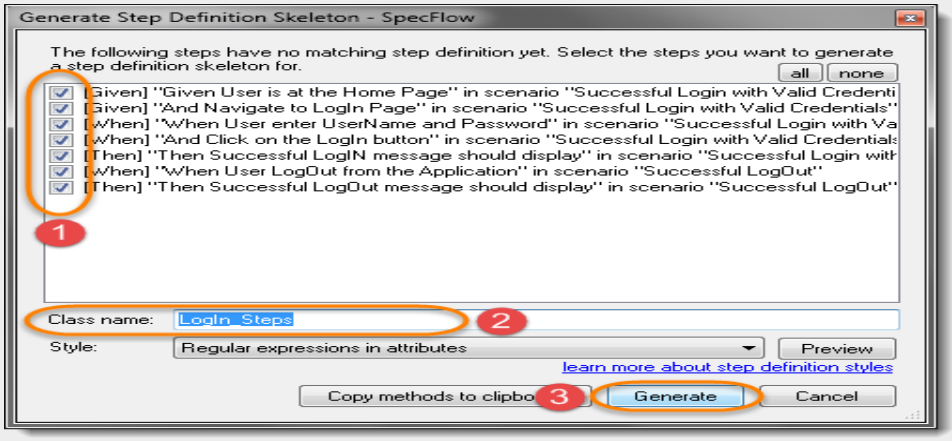
Right-click on your *feature file* in the code editor and select ***Generate Step Definitions*** from the popup menu.



5.5 Steps Definitions(Cont…)

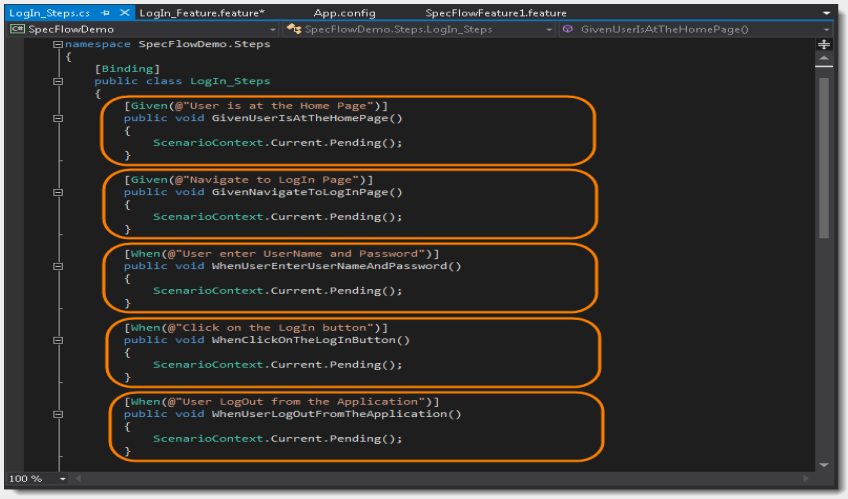
Create a Step Definition File

This display a Pop Up window, which will ask to select the statements for which *Step Definition* file is to be created. Select all and click on ***Generate*** *button.* Do not forget to give it a logical name. In this case I named it as '*Login_Steps*', as all the steps in this will file will be related to user Login.



5.5 Steps Definitions(Cont…)
Create a Step Definition File

To have a look at the attached *definitions*, click on the any statement and press ***F12*** button. This will open up the linked *definition file* and the cursor will be pointing to the linked definition.



5.6 Scenario Outline

- *SpecFlow* inherently supports *Data Driven testing* by the use of the ***Scenario Outline*** and ***Examples*** section.
- It is with these keywords that *SpecFlow* allows for easy *Data Driven testing* to be completed where no changes need to be made to the Java file.
- ***Scenario Outline*** - *This is used to run the same scenario for 2 or more different sets of test data. E.g. In our scenario, if you want to register another user you can data drive the same scenario twice.*
- ***Examples*** – *All scenario outlines have to be followed with the Examples section. This contains the data that has to be passed on to the scenario.*

```
Scenario Outline: Successful Login with Valid Credentials
        Given User is at the Home Page
        And Navigate to LogIn Page
        When User enter <username> and <password>
        And Click on the LogIn button
        Then Successful LogIN message should display
Examples:
| username   | password |
| testuser_1 | Test@123 |
| testuser_2 | Test@153 |
```

5.6 Scenario Outline (Continue….)

There is a little change in the syntax of the *Definition* file for the corresponding method.
Change the below line:
[When(@"User enter '(.)' and '(.)'")]
To
[When(@"User enter (.) and (.)")]
***Note**: it looks similar, but if you notice carefully there is a little change in the regular expression*.

```
[When(@"User enter (.*) and (.*)")]
public void WhenUserEnterAnd(string username, string password)
{
    driver.FindElement(By.Id("log")).SendKeys(username);
    driver.FindElement(By.Id("pwd")).SendKeys(password);
}
```

5.7 Data Table

- *No keyword is used to define the test data*
- *This works only for the single step, below which it is defined*
- *A separate code needs to understand the test data and then it can be run single or multiple times but again just for the single step, not for the complete test*

- Feature: LogIn_Feature
  In order to access my account
  As a user of the website
- I want to log into the website

5.7 Data Table Continue…

```
Scenario: Successful Login with Valid Credentials
        Given User is at the Home Page
        And Navigate to LogIn Page
        When User enter credentials
        | Key      | Value      |
        | Username | testuser_1 |
        | Password | Test@123   |
        And Click on the LogIn button
        Then Successful LogIN message should display

Scenario: Successful LogOut
        When User LogOut from the Application
        Then Successful LogOut message should display
```