Test Automation Using Selenium with C#

Vaibhav Mittal
Navneesh Garg

- Microsoft Visual Studio 2017
- Learn Automation on a Web Based Application
- Real Life Experiences
- Step by Step Instructions
- BDD and Continuous Integration

Selenium with C# - Step By Step Guide

Test Automation Using Selenium with C#

Vaibhav Mittal & Navneesh Garg

ISBN - 978-0-9922935-5-0

Publisher: Adactin Group Pty Ltd.

Contents

About the Author

Vaibhav Mittal

Vaibhav Mittal is a seasoned Consultant with experience in the complete life cycle of software development and testing. He has worked across geographies for multiple clients helping them solve their problems in the most efficient manner. He has exposure to multiple technologies including Microsoft .NET, Java, Oracle, ETL, Business Intelligence and Analytics.

As a Chief Information Officer at Adactin Group Pty Ltd, Sydney, he is responsible for multiple projects and works in the capacity of Testing and Automation expert. He is helping his clients grow in the space of software testing while using agile methodologies. As a trainer he has trained multiple professionals on various technologies.

Before joining Adactin Group, he worked with November Research Group, Oracle and Adobe Systems. He has consulted for clients from USA, Europe, Japan and India. He has presented multiple papers in international conferences held by SQS and IQNITE.

Navneesh Garg

Navneesh Garg is a recognized test automation architect and corporate trainer, specializing in test automation, performance testing, security testing and test management. As a tool specialist, he has worked on a variety of functional automation tools including Selenium, HP QTP/UFT, TestComplete, TestPartner, SilkTest, Watir, RFT, and on varied technologies including Web, Java, Dot-net, SAP, Peoplesoft and Seibel.

His previous book “Test Automation using Unified Functional Testing” is among the bestselling books on HP QTP. This book has consistently ranked among the top 100 testing books on Amazon. It was the first book to be released globally on the latest version of HP QTP.

He is an entrepreneur and founder of several successful IT companies which encompass the AdactIn Group, CresTech Software, and Planios Technologies.

As an experienced corporate trainer, he has trained professionals in Selenium and other test tools across a wide range of global clients such as Macquarie Bank, Corporate Express, Max New York Life, Accenture, NSW Road and Maritime Services, Australian Dept of Education, HCL Technologies, Sapient, Fidelity Group, Adobe Systems, and many more. He has training experience in diverse geographies such as Australia, India, Hong Kong and USA.

As a technical test delivery head for his company, he has led and managed functional automation testing and performance testing teams across a wide range of domains, using commercial tools and open source tools. Certified in HP QTP, HP Quality Center, HP LoadRunner, IBM Rational Functional Tester and as a Certified Ethical Hacker, he has designed several high-end automation frameworks including using Selenium and its integrations with tools like TestNG, JUnit, Selenium Grid, Jenkins and ANT.

Preface

The motivation for writing this book stems from my hands-on experience in the IT and testing domains and the experience I have gained as automation consultant working in numerous complex automation projects.

Selenium, being an open source tool, is gaining huge popularity but still is not conceived as an easy to use tool especially by testers due to a variety of reasons, including tool setup, programming background and support issues. A key objective of this book is showcase in a simple guided way to use Selenium WebDriver so that we can attain maximum return on investment from using the tool. Not only will we learn how to use the tool but also how to effectively create maintainable frameworks using Selenium.

Scope of Topics

As part of the scope of this book we will cover **Selenium with C#** as the programming language with Visual Studio 2017 community edition.

We will be using **Visual Studio 2017** as the main IDE for creating selenium tests.

No prior knowledge of C# language is required for this book but having an understanding of object oriented programming language concepts will definitely help. As part of this book we will be covering **Basics of C#** which would be required to use **Selenium** for beginner users.

We will also learn how Selenium integrates with **continuous Integration** tools like **Team Foundation Server.**

Our intent in this book is to discuss the key features of Selenium and cover all crucial aspects of the tool in order to help you **create effective automation frameworks using Selenium with C#**.

Target Audience

The target audience for this book are manual, functional testers who want to **learn Selenium quickl**y and who want to create effective automation frameworks that generate positive ROIs to stakeholders.

Salient Features of this Book

This book has been designed with the objective of **simplicity and ease of understanding**.

A major fear amongst functional testers who want to learn Selenium is the fear of the programming language and coding. We address these fears by covering just enough **basics on C# programming language** that will give you the confidence to use Selenium.

This book follows a **unique training based approach** instead of a regular text book approach. Using a step by step approach, we guide you through the exercises using pictorial snapshots.

We also provide step by step installation and configuration of Visual Studio before using Selenium.

Instead of using custom html pages with few form fields and links, this book utilizes a custom developed, Web based application containing many form fields and links.

Another differentiator is that we have tried to include **many practical examples and issues** which most automation testers encounter in their day-to-day activities. We share our real-life experience with you to give you an insight into what challenges you could face while implementing an automation solution on your project. Our practical examples cover how to use most of the features within Selenium.

We also cover aspects of **Continuous Integration tool; Team Foundation Server** so that Selenium scripts can be integrated with the development environment and run on nightly builds.

Finally, the book includes a special section devoted to answering the most **common interview questions** relating to test automation and Selenium.

Sample Application and Source Used in Book

The sample application used in the book can be accessed at the following URL:

www.adactin.com/HotelApp/

The source code used in the book can be found at the following link

www.adactin.com/store/

Feedback and Queries

For any feedback or queries you can contact the author at www.adactin.com/contact.html or email info@adactin.com

Order this book

For bulk orders, contact us at orders@adactin.com

You can also place your order online at adactin.com/store/

Acknowledgements

My family has always been compassionate in this journey. My wife and two little ones (Vedaang and Sia) have always encouraged and supported me despite it took a lot of my time being away from them. It would not have been possible without their support.

I don't want to miss an opportunity to express my gratitude to all others, who saw me through this book; to all those who provided support, talked things over, read, write, offered comments, allowed me to quote their remarks and assisted in editing, proofreading and overall designing of this book. Thank you all – my co-workers, friends, peers and my commercial team for extending your support.

Vaibhav Mittal

I would like to thank my family (my parents, my wife Sapna, my wonderful kids Shaurya and Adaa) for their continued support. Without them this book would not have been possible.

I would also like to thank my colleagues and clients for the inspiration, knowledge and learning opportunities provided.

Navneesh Garg

1. Introduction to Automation

Introduction

In this chapter, we will talk about automation fundamentals and understand what automation is and the need for automation. An important objective of this chapter is to understand the economics of automation and determine when we should carry out automation in our projects. We will also discuss some popular commercial and open source automation tools available in the market.

Key objectives:

- What is automation?
- Why automate? What are the benefits of automation?
- Economics of automation.
- Commercial and Open Source automation tools.

1.1. What is Functional Automation?

Automation testing is to automate the execution of manually designed test cases without any human intervention.

The purpose of automated testing is to execute manual functional tests quickly in a cost-effective manner. Frequently, we rerun tests that have been previously executed (also called regression testing) to validate the functional correctness of the application. Think of a scenario where you need to validate the username and password for an application which has more than 10,000 users. It can be a tedious and monotonous task for a manual tester and this is where the real benefits of automation can be harnessed. We want to free up the manual functional testers' time so that they can perform other key tasks while automation provides extensive coverage to the overall test effort.

When we use the term "automation", there is usually confusion about whether automation scope includes functional and performance testing. Automation covers both.

- Functional Automation: Used for automation of functional test cases in the regression test bed.
- Performance Automation: Used for automation of non-functional performance test cases. An example of this is measuring the response time of the application under considerable (e.g., 100 users) load.

Functional automation and performance automation are two distinct terms and their automation internals work using different driving concepts. Hence, there are separate tools for functional automation and performance automation.

For the scope of this book, we will be referring only to **Functional Automation**.

1.2. Why do we automate?

Listed below are the key benefits of Functional Automation:

1. Effective Smoke (or Build Verification) Testing

Whenever a new software build or release is received, a test (generally referred to as the "smoke test" or "shakedown test") is run to verify if the build is testable for a bigger testing effort and major application functionalities are working correctly. Many times we spend hours doing this only to discover that a faulty software build resulted in all the testing efforts going to waste. Testing has to now start all over again after release of a new build.

If the smoke test is automated, the smoke test scripts can be run by developers to verify the build quality before being released to the testing team.

2. Standalone - Lights Out Testing

Automated testing tools can be programmed to kick off a script at a specific time.

If needed, automated tests can be automatically kicked off overnight, and the testers can analyseanalyze the results of the automated test the next morning. This will save valuable test execution time for the testers.

3. Increased Repeatability

At times it becomes impossible to reproduce a defect which was found during manual testing. The key reason for this could be that the tester forgot which combinations of test steps led to the error message; hence, he is unable to reproduce the defect. Automated testing scripts take the guesswork out of test repeatability.

4. Testers can Focus on Advanced Issues

As tests are automated, automated scripts can be baselined and rerun for regression testing. Regression tests generally yield fewer new defects as opposed to testing newly developed features. So, functional testers can focus on analysinganalyzing and testing newer or more complex areas that have the potential for most of the defects while automated test scripts can be used for regression test execution.

5. Higher Functional Test Coverage

With automated testing, a large number of data combinations can be tested which might not be practically feasible with manual testing. We use the term "Data driven testing" which means validating numerous test data combinations using one automated script.

6. Other Benefits

- **Reliable**: Tests perform precisely the same operations each time they are run, thereby eliminating human error.
- **Repeatable**: You can test how the software reacts under repeated execution of the same operations.
- **Programmable**: You can program sophisticated tests that bring out hidden information from the application.
- **Comprehensive**: You can build a suite of tests that cover every feature in your application.

7. Managing User Interface Controls

Introduction

In the last chapter, we recorded a sample script and replayed it. You would be very curious to understand how Katalon actually replayed the whole script. How could it identify the location field and enter the value that we had earlier recorded in the previous chapter? Was it like a video recording that got replayed? In this chapter, we will understand test automation fundamentals of how a recorded script is replayed and how automation tools recognize objects on the application.

Key objectives:

- Understanding Object Recognition fundamentals and how Selenium replays.
- Understanding various locators to identify objects.

7.1. How Does Selenium Replay Scripts?

Let us take a simple example:

Assume that you parked your car on some level of a big shopping mall before going to a party. For the sake of this example, say you had too many drinks at the party and so took a cab to get home. Next morning, you come back to the mall to pick up your car but you do not remember the location of the car apart from knowing the level on which your car was parked. How will you find your car? Assume that you do not have a remote control for the car!

Figure 7.1 – Sample Car

If I were to find my car, I will go looking for my car in the first row and look for the Make and Color of my car. If I find a car with the same make and color, I will go closer to the car and try to identify my car based on the registration number. If I can match all these three properties, I am sure I will find my car. So the three properties I will look for will be:

- Make of the car.
- Color of the car.
- Registration No. of the car.

I do not really need to care about height, width or any other details about my car.

This is what Selenium as a tool does and as a matter of fact, this principle is followed by all other automation tools available in the market. They use some key properties of the objects to identify the user interface controls and then use those properties to identify the objects. For instance, if the user clicks on a button, Selenium uses the label of the button to identify the object.

So this is the process of how Katalon Recorder replays a script:

- While recording, Selenium stores object property information somewhere in the script.
- When we replay the script, Selenium will pick up the object properties and try to find the object in the application by matching the properties.
- Once it finds the object, it will perform the operation (click, select, etc.) on that object.

This is the basic automation fundamental required to understand how functional automation tools work. The key point to remember is that the Selenium script is not a video recording of functionality but a step-by-step execution of actions recorded in the script.

7.2. Locating the Elements on a Web page

Every Web page is nothing but a set of different UI web elements or objects. Before we work with an element on a page, we need Selenium to locate that element. The element on the Web page can be located by various locator types.

To understand the various locators one needs to have a basic understanding of HTML. *Id, name, input, type*, etc., are the HTML tags/attributes. Using these HTML tags, attributes like "xpath" can be constructed. We can use these tags or attributes to identify elements.

Let us follow these steps to understand tags and locators.

1. Launch Sample Application URL: www.adactin.com/HotelApp
2. Right-click on the home page and select **View Page Source.**

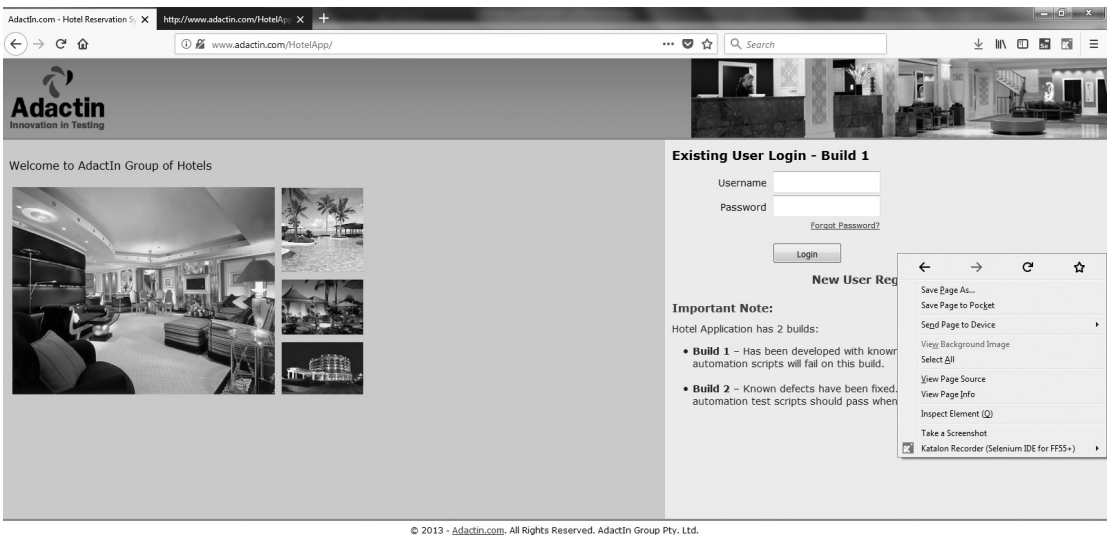

Figure 7.2 – View Page Source Option

15. Synchronizing WebDriver Scripts

In any Web automation project, the automation success depends upon the robustness of your scripts; whether that's adaptation of your code to project or software changes or synchronization of the script with the site's performance.

Many a time, your application performance will vary which will require you to manipulate your WebDriver script's execution speed.

In one of the applications that we tested, it took more than 60 seconds for an application form to save and confirm that save was successful. How does WebDriver support these situations?

Synchronization is a critical issue for any test automation script. You may think that synchronization of test script actions is a built-in ability of today's functional testing tools. Reality shows that many unexpected test script failures are related to synchronization issues generating false negative results. These false negatives make it hard to detect real application problems as each test script failure may be related to a test script synchronization issue. Synchronization errors are timing issues, therefore, they are non-deterministic, heavily dependent on the HW/SW, the network, and their utilization. The biggest challenge in automating a Web application is the loading of a Web page which is always at the mercy of certain conditions, such as:

- Load on the server.
- Network speed.
- Performance of AUT.
- Ajax call to load an element.

Key objectives:

- What is Synchronization?
- Approaches to script synchronization.
- Synchronizing a script.

15.1. What is Synchronization?

What is script synchronization?

Test scripts need to be synchronized in a way that the script driving the application **waits** until the AUT is ready to accept the next user input.

The following are some situations where synchronization is required:

2. Select the **project** and click on paste.

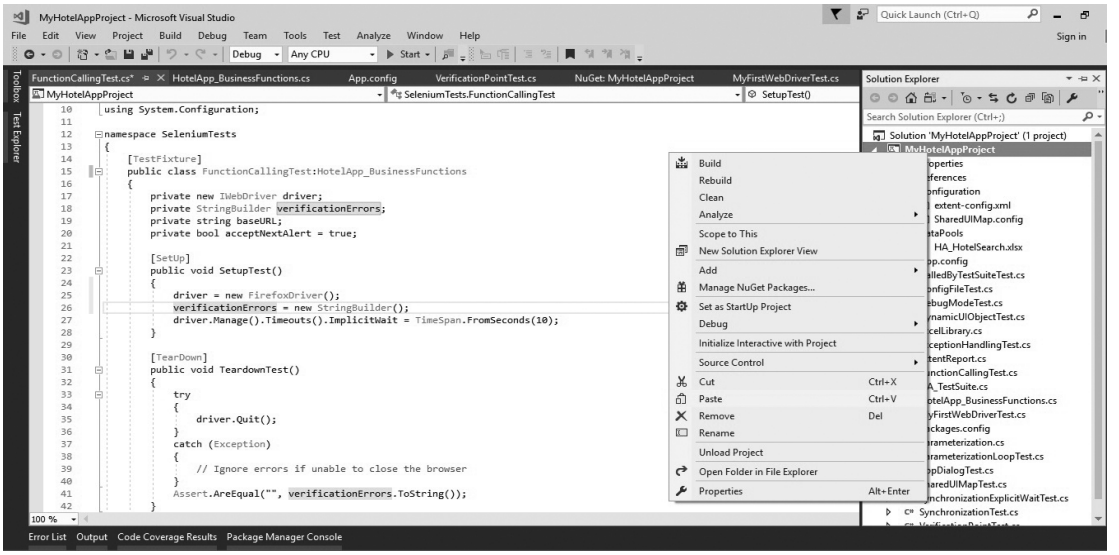

Figure 15.5 – Paste Script

A copy of the class file will be created. Right-click on the new class file to rename. Rename to **"SynchronizationTest.cs".** Click **OK**.

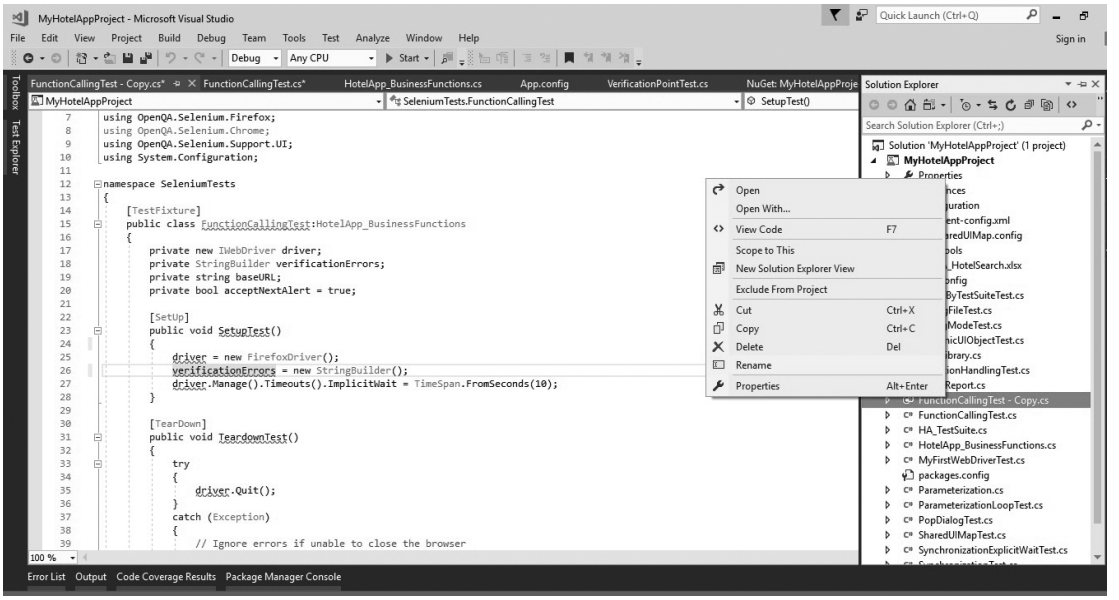

Figure 15.6 – Rename

24. Behavior Driven Development

Introduction

Behavior Driven Development or BDD is an approach to development that improves communication and bridges the gap between business stakeholders and technical teams to create software with business value. BDD uses a common language for communication which is easily understandable.

Let us understand in detail about BDD and how it works.

Test Driven Development (TDD)

BDD is an extension of TDD so we have to understand TDD before we go further.

TDD is an iterative development process. Each iteration starts with a set of tests written for a new piece of functionality. These tests are supposed to fail during the start of iteration as there will be no application code corresponding to the tests. In the next phase of the iteration Application code is written with an intention to pass all the tests written earlier in the iteration. Once the application code is ready tests are run.

Any failures in the test run are marked and more Application code is written/re-factored to make these tests pass. Once application code is added/re-factored the tests are run again. This cycle keeps on happening till all the tests pass. Once all the tests pass, we can be sure that all the features for which tests were written have been developed.

Benefits of TDD

- Unit test proves that the code actually works.
- Can drive the design of the program.
- Refactoring allows improving the design of the code.
- Low Level regression test suite.
- Test first reduces the cost of the bugs.

Drawbacks of TDD

- The developer can consider it as a waste of time.
- The test can be targeted on verification of classes and methods and not on what the code really should do.
- Test becomes part of the maintenance overhead of a project.
- Rewrite the test when requirements change.

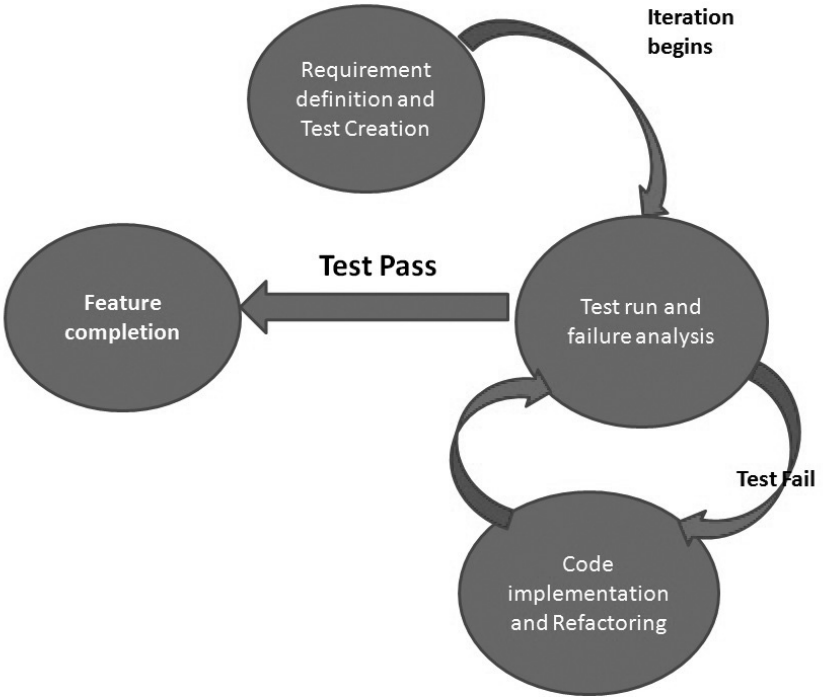

Figure 24.1 – Test Driven Development

With this understanding of TDD we will move to BDD which will form the basis of understanding **Gherkin** and eventually **SpecFlow**.

Behavior Driven Development

In the last section, we discussed what TDD is. We discussed how TDD is test-centered development process in which we start writing tests firsts. Initially, these tests fail but as we add more application code these tests pass. This helps us in many ways:

- We write application code based on the tests. This gives a test first environment for development and the generated application code turns out to be bug free.
- With each iteration, we write tests and as a result, with each iteration, we get an automated regression pack. This turns out to be very helpful because, with every iteration, we can be sure that earlier features are working.
- These tests serve as documentation of application behavior and reference for future iterations.

Behavior Driven testing is an extension of TDD. Like in TDD, in BDD also we write tests first and then add application code. The major differences that we get to see here are:

- Tests are written in plain descriptive English type grammar.
- Tests are explained as behavior of the application and are more user-focused.
- Using examples to clarify requirements.
- This difference brings in the need to have a language which can define, in an understandable format.

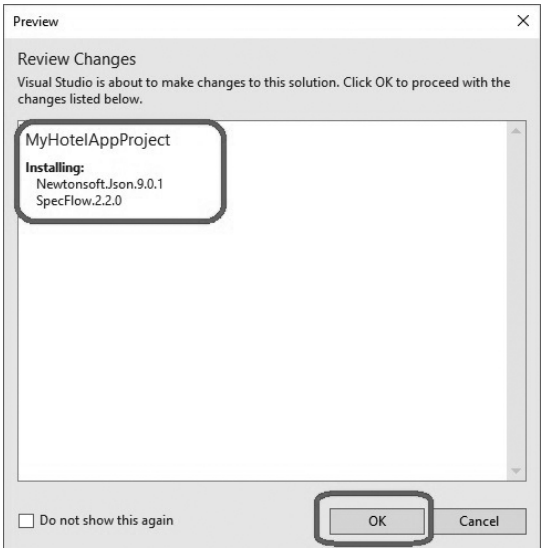

Figure 24.6 – SpecFlow Installation Pop-up

24.4. Creating your First SpecFlow Test

Create a Feature File

1. Now that you have the environment setup, you can actually use *SpecFlow*. Before adding any new files, let us create two new folders called **Features** and **Steps** in the project. To do this, right-click on the **project** and select **Add → New Folder**. The purpose of this step is to add all the feature files into the Features folder and all the Step Definitions into the Steps folder. This helps in code readability. We will understand it in detail as we proceed further.

2. To create your first feature file, you can right-click on the ***Feature* folder** and select **Add –> New Item…** from the context menu. Select **SpecFlow Feature File**. Give it a logical name. When you name your *SpecFlow* feature files, try to name them similar to the feature it will be testing. Click **Add.** The Feature file will be added to your project **Features** folder.

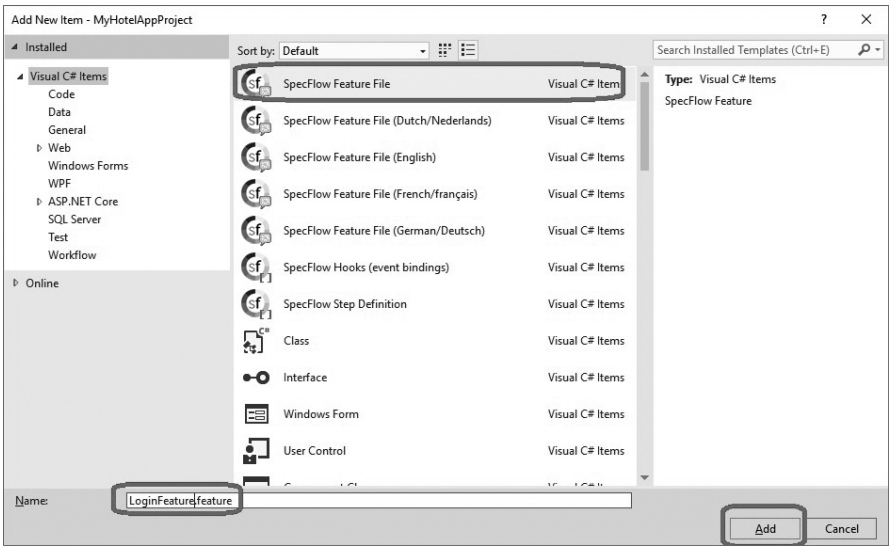

Figure 24.7 – Add Feature File

25. Automation Frameworks

Being a part of the software testing domain, we would have heard the term “Automation Frameworks” many times. Again, it is a very common question one encounters at interviews too. In this chapter we will try to understand the answers to these basic questions:

- Why do we need a framework? What are the advantages of frameworks?
- What exactly is an automation framework? What are the components of the framework?
- How do we implement frameworks? What are the different types of frameworks?

25.1. Why do we need Automation Frameworks?

1. Maintainability

One of the key reasons behind creating an automation framework is to reduce the cost of script maintenance. If there is any change in the functionality of the application, then we need to get our scripts fixed and working utilizing the least amount of time and effort.

Ideally, there should not be too much need to update the scripts, in case the application changes. Most of the fixes should be handled at the framework level.

2. Productivity

If we ask how many manual test cases we can automate in a day that might be a difficult question to answer. But the important thing to ask is whether we can increase our productivity by automating more test cases per day?

Yes, we can. If we have an effective framework, we can increase the productivity manifold. In one of our previous projects, we increased the productivity from 3-4 test cases a day to 10-12 test cases a day, mainly through effective framework implementation.

3. Learning curve

If you have a new person joining your team, you would like to reduce the effort in training the person, and have him/her up and running on the framework as soon as possible.

Creating an effective framework helps reduce the learning curve.

As a best practice, we always advise our clients to keep the framework as simple as possible.

4. Make result analysis easier

Once the test cases are automated, a lot of time is spent by the testing team on analyzing the results. Sometimes they are not detailed enough, which might make it hard to pinpoint the error. Most often, it is not script failure but environment or data issues that turn out to be the source of problems. A better reporting format in the framework will cut down on result analysis time considerably.