Selenium

Contents

Chapter 1. What is IDE

An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of at least a source code editor, build automation tools and a debugger. Some IDEs, such as NetBeans and Eclipse, contain the necessary compiler, interpreter, or both; others, such as SharpDevelop and Lazarus, do not.

The boundary between an IDE and other parts of the broader software development environment is not well-defined; sometimes a version control system or various tools to simplify the construction of a graphical user interface (GUI) are integrated. Many modern IDEs also have a class browser, an object browser, and a class hierarchy diagram for use in object-oriented software development.

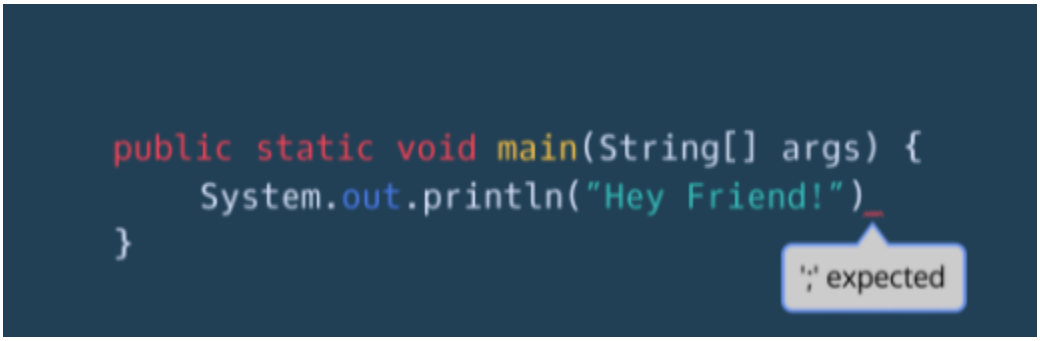
What to use IDE for : Syntax Highlighting

- **Highlight**

```
// with syntax highlighting

public class NiceDay {
  public static void main(String[] args) {
    System.out.println("It's a nice day out!");
  }
}
```

- **Building Executables**
- **Debugging**

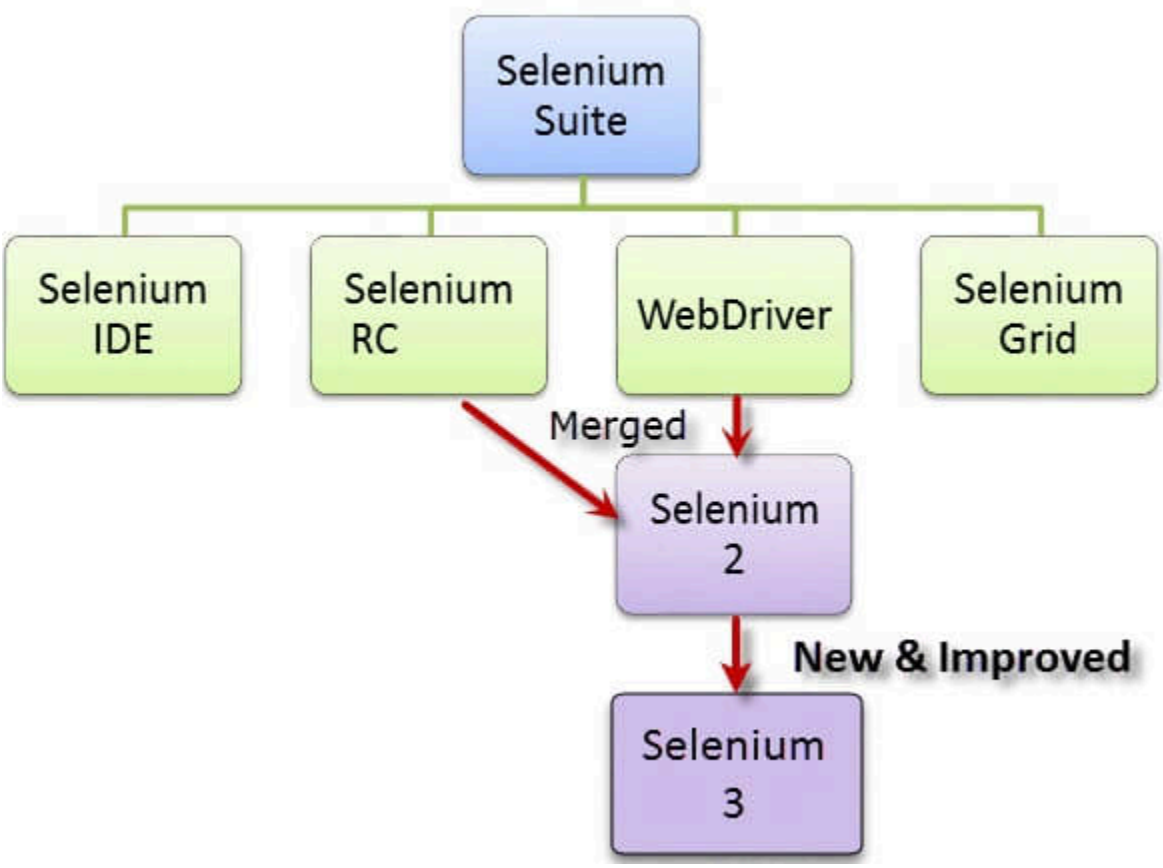


- **Coding On Your Computer**

Chapter 2. What is Selenium

Selenium is a free (open-source) automated testing framework used to validate web applications across different browsers and platforms. You can use multiple programming languages like Java, C#, Python etc to create Selenium Test Scripts. Testing done using the Selenium testing tool is usually referred to as Selenium Testing.

Selenium Software is not just a single tool but a suite of software, each piece catering to different Selenium QA testing needs of an organization.Here is the list of tools



Chapter 3. Pros and cons of automation testing

Just like everything - automation testing has its pros and cons. Here we will look at the pros and cons of the automation process.

Pros	Cons
Reduce the time required to regression	Unstable outcomes and false test alarms due to their poor design and implementation
After fixing all bugs or when new functionality is added, report on product status as quickly as possible;	Mistakes in automated tests lead to errors and omissions
Get an infinite number of test runs when using continuous software development mechanisms	Recruitment of employees with knowledge of programming languages and testing technologies;
Reduce the number of manual tests	High cost of implementation
Automate complex test cases	Due to strict algorithms, automated tests can find only part of mistakes
Detect errors overlooked during manual testing	Much time is needed to support automation

In addition there are couple of things to take care when you automate such as

1. Not all test cases need to be automated
2. Time needed for automation testing is comparable to the time spent on software development
3. Before writing automated tests you need to work on test system architecture. Otherwise apart from manual testing you will need to support automated tests
4. When implementing automation, it is impossible to foresee all the risks and challenges you might face in the future

Chapter 4. SpecFlow BDD

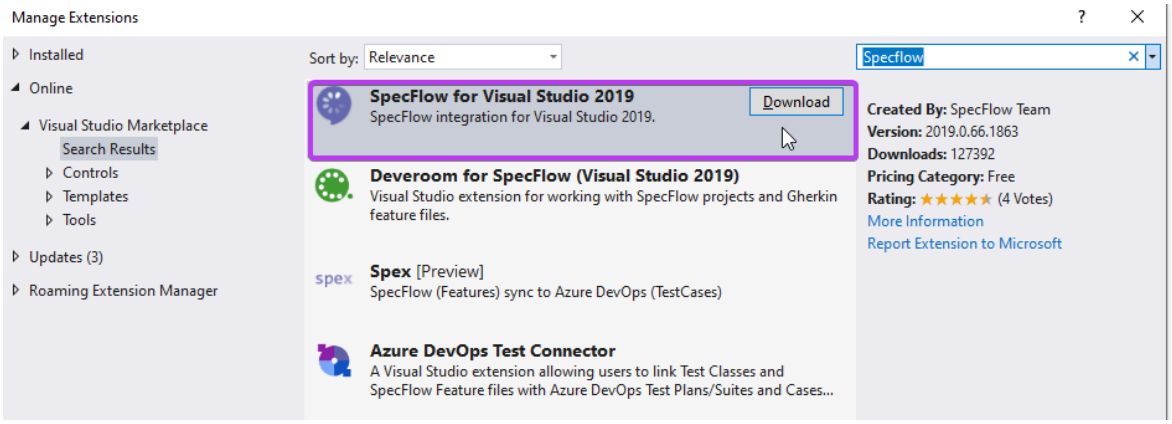
How to add SpecFlow into your project

Your IDE needs a plugin called SpecFlow in order to functionate properly and red your tests.

Get familiar with what IDE is

Here you will learn how to add SpecFlow into your project :

1. Open Visual Studio.
2. Go to Extensions.
3. Then go to Manage Extensions.
4. Choose online.
5. In the search box type SpecFlow.
6. Hit the download button to begin the installation procces



Your IDE will be able to read your tests

```
1   Feature: SpecFlow Selenium Sample
2
3   @Home
4   Scenario: Sign In Link Exists
5       Given I have navigated to my test site
6       When the Home page loads
7       Then the Sign In link appears on the Home page
8
9   @Home @SignIn
10  Scenario: Sign In Link Takes User To The Sign In Page
11      Given I have navigated to my test site
12      When I click the Sign In link
13      Then I am taken to the Sign In page
14
```

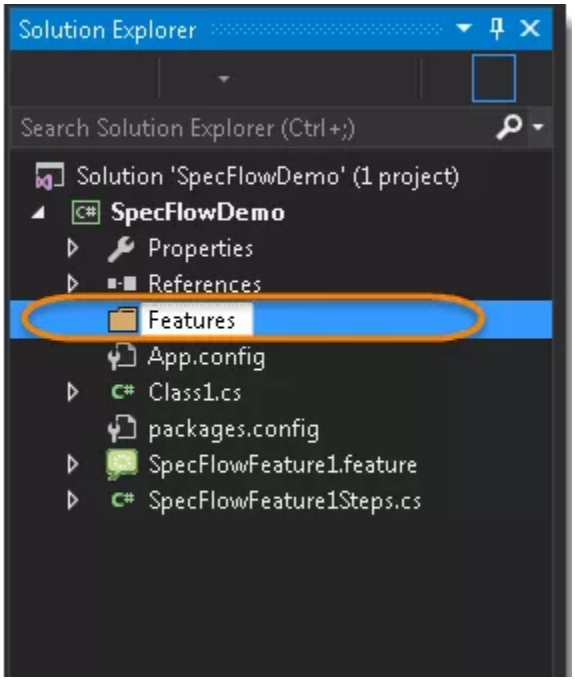
The Blue color of the Given, When, Then means the IDE reads the tests.

How to Create feature files

To run your test you need steps to be followed. In the future files you will write down your steps. Every web page in the given web site needs a separate feature file.

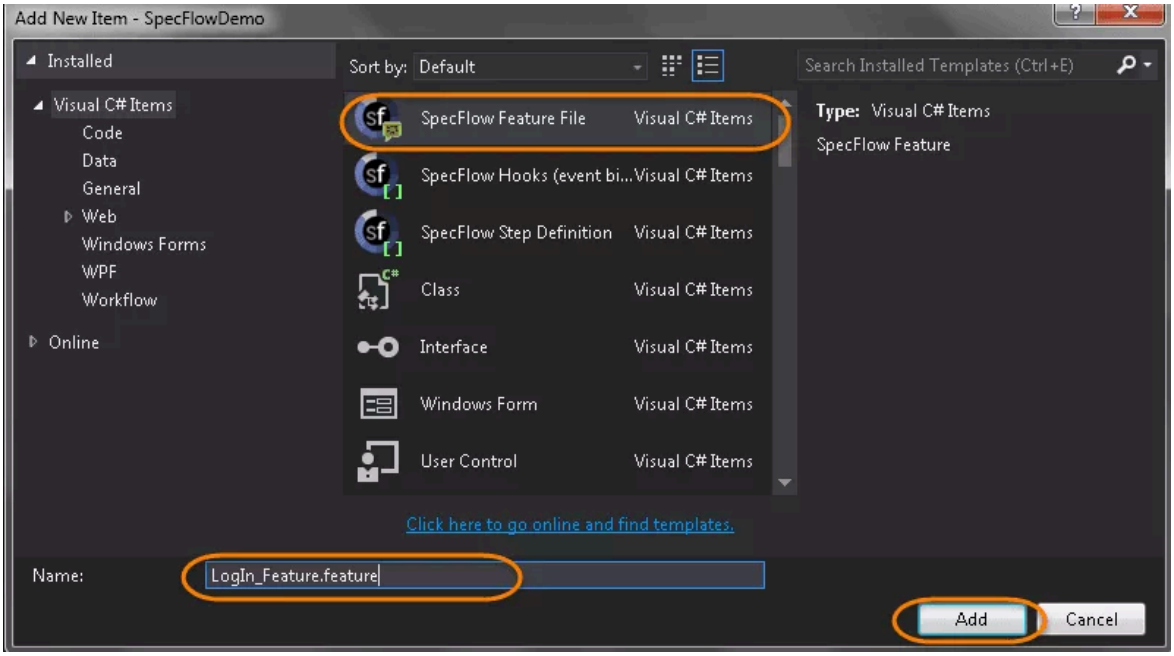
You will learn how to create feature files :

1. Create a separate folder for the feature files in the project



2. Click with the right button of the mouse on the folder
3. Choose new item
4. Navigate to SpecFlow
5. Choose the top option

6. Name the file in the name field
7. Click on Add



You will be able to write down your tests.

For Example:

You want to create a couple of tests for a Login.

Chapter 5. Locators

How to create locator variables

In order to perform an automation test we need locators for the Selenium to perform actions on.

In this topic we gonna found out how to create locators :

1. Create a class named after the web page you are on (every page of the web site must have separte class).
2. Create a variable as a private of type IWebElement.
3. Name the variable. `WebElement textbox=`

You will learn how to find locators.

For Example:

You want to click on a login button. To do that you have to find the locator and perform action on it.

How to find locators by Id

Locators can be found with many possible ways. One of them is by Id which is the most reliable one.

Read how to create locator variables.

You will find out how to find locators by id :

1. Create a locator
2. Go to the dom in the web site and search for the id of the given web element

```
<input type="text" name="j_username" id="usernameId" class="loginUserId" autocomplete="off" size="15" maxlength="35" tabindex="1"> == $0
```

3. Use lambda => after the locator's name or =
4. Use the command driver.FindBy(By.id()) after the lambda
5. Put the id into the By.id brackets

```
WebElement textbox=driver.findElement(By.id("usernameId"));
```

You will be able to perform action on the locator

How to find locators by Class Name

Locators can be found with many possible ways. One of them is by class name which is the most reliable one after id.

Read how to create local variables

You will find out how to find locators by class name :

1. Create a locator
2. Go to the dom in the web site and search for the class name of the given web element

3. Use lambda => after the locator's name or =
4. Use the command driver.FindBy(By.className()) after the lambda
5. Put the class into the by.className brackets

```
driver.findElement(By.className("<< Class Name of the particular Web Element >>"));
```

You will be able to perform action on the locator

Chapter 6. Hooks

How to create your webdriver

In order to run your test on a certain browser you need to do some architecture work such as initiating the webdriver of the given browser you want

1. Every browser has a different webdriver. If you want cross browser testing you need to initialize a webdriver for every browser
2. Get familiar with annotations

Here you will find out how to initilize a webdriver :

1. Create a separate class named Hooks and use [Binding] annotation
2. Initiate the webdriver in a method and use [BeforeScenario] annotation over the method

```
[Binding]
public class Hooks
{
    public IWebDriver _driver;

    [BeforeScenario]
    public void BeforeScenario()
    {
        if (_driver == null)
        {
            _driver = new ChromeDriver();
        }
        else { throw new Exception("Couldn't initialize the driver"); }
    }
```

You will be able to run your test in the given browser

How to close your webdriver

To finish your test successfully you have to close your driver

Read the document of how to create your webdriver :

Here you will learn how to make the hook to be able to finish your test

1. Go to the Hooks class you have created
2. Close the webdriver in a method and use [AfterScenario] annotation over the method

```
[AfterScenario]
public void AfterScenario()
{
    if (_driver != null)
    {
        _driver.Quit();
    }
    else throw new Exception("There was an error while trying to close the driver");
}
```

3. Inherit the page hook into your class of the given page

```
public class GoogleSearchPageObjects : Hooks
```

You will be able to finish your test successfully

If you want to run your next test you will have to create a new webdriver. In order to create a new one you need to close the old one. That's why its so important to close your driver