BDD

Lesson – 8 : Selenium – Business readable UI automation with cucumber

Lesson Objectives

In this lesson, you will learn:
Business readable UI automation with cucumber

Reference : https://www.cs.colorado.edu/~kena/classes/5828/s12/lectures/09-bddcucumber.pdf

Introduction

- Cucumber does not communicate directly with the applications, it needs to be used in conjunction with tools such as Selenium WebDriver
- Cucumber acts more like an execution framework.
- Cucumber's use of English-like language (Gherkin) for its programming function, test case execution is supported with features and scenarios executed based on their definitions.

Reference : https://www.cigniti.com/blog/successful-test-automation-with-selenium-and-cucumber/

Business readable UI automation with cucumber

There are 3 steps for writing automated tests that deliver high value while requiring only low maintenance using Cucumber:

Step 1 : *Define scenarios*:

- The acceptance tests are written in English-like language called Gherkin.
- The scenarios defined are based on the user stories and features defined by the BA team.
- Sometimes, the BA also creates the Feature files for use by the testing team.

Step 2: Create step definitions:

- Once the scenarios are defined, the steps have to be implemented for execution.
- This can be done in a variety of different languages supported by Cucumber.
- For instance, if a language like Java is selected for implementation, the necessary classes and methods are defined by creating a project structure.
- The project can have references added to the Selenium jars, so that the packages can be imported and used to implement the steps to drive browsers using Selenium API.

Step 3: Define UI Element descriptions:

- One of the best ways to define UI element descriptions is using the PageObject design pattern.
- PageObject pattern makes automated test maintenance easier.
- This is because any changes made to the page elements are abstracted into the PageObjects itself, without the need to update feature files and step definitions.

Reference : https://www.cigniti.com/blog/successful-test-automation-with-selenium-and-cucumber/

Business readable UI automation with cucumber

Write a test in a Feature File

Test to Automate

Ritika visits ToolsQA Demo Website for the first time and SEARCH for Dress. She selects the first product and goes to product page. She successfully adds it to the bag. She continues to Cart Page from mini cart icon at the top right corner. Then she moves forward to Checkout page. She choose to be an ANONYMOUS USER (Not Registering) completes email and address details. She selects FREE delivery, and places the order using CHECK payment method with Billing & Delivery address as same.

Cucumber BDD Style Test
Given user is on Home Page
When he search for "dress"
And choose to buy the first item
And moves to checkout from mini cart
And enter personal details on checkout page
And select same delivery address
And select payment method as "check" payment
And place the order

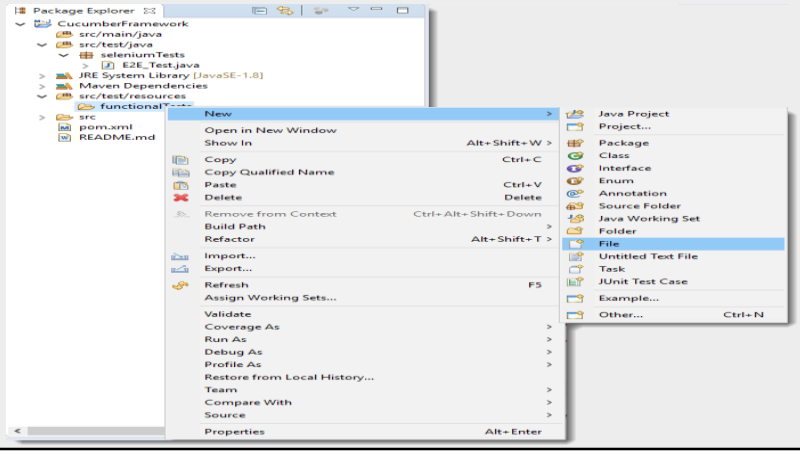
Reference : https://www.cigniti.com/blog/successful-test-automation-with-selenium-and-cucumber/

Business readable UI automation with cucumber

Create Feature File

1. Create a *New Package* and name it as ***functionalTests***, by *right click* on the ***src/test/resources*** and select ***New >> Package***. ***Note****: It is always recommended to put all the feature files in the* ***resources*** *folder.*

2. Create a *Feature* file and name it as ***End2End_Test.feature*** by right click on the above created package and select ***New >> File***.

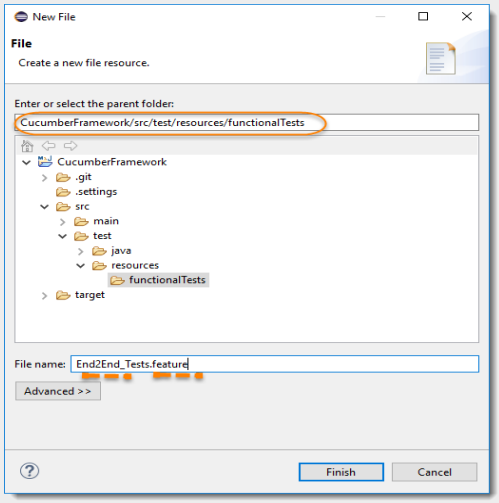


Reference : http://toolsqa.com/selenium-cucumber-framework/convert-selenium-test-into-cucumber-bdd-style-test/

Business readable UI automation with cucumber

Create Feature File

Make sure to add ***.feature*** extension to the file.



Reference : http://toolsqa.com/selenium-cucumber-framework/convert-selenium-test-into-cucumber-bdd-style-test/

Business readable UI automation with cucumber

Create Feature File

Add the test steps to the feature file.

End2End_Tests.feature

```
1  Feature: Automated End2End Tests
2  Description: The purpose of this feature is to test End 2 End integration.
3
4  Scenario: Customer place an order by purchasing an item from search
5      Given user is on Home Page
6      When he search for "dress"
7      And choose to buy the first item
8      And moves to checkout from mini cart
9      And enter personal details on checkout page
10     And select same delivery address
11     And select payment method as "check" payment
12     And place the order
```

***Note**: Notices that all the keywords used in the test steps are in different colors. These are Gherkin keywords. By default Eclipse does not understand these but if you install the cucumber Eclipse plugin-in, this will be recognized.*

Business readable UI automation with cucumber

Create a JUnit Test Runner

1. Create a *New Package* and name it as ***runners*** by *right click*on the *src/test/java* and select ***New >> Package***.
2. Create a *New Java Class* file and name it as ***TestRunner*** by *right click* on the above created package and select ***New >> Class***.

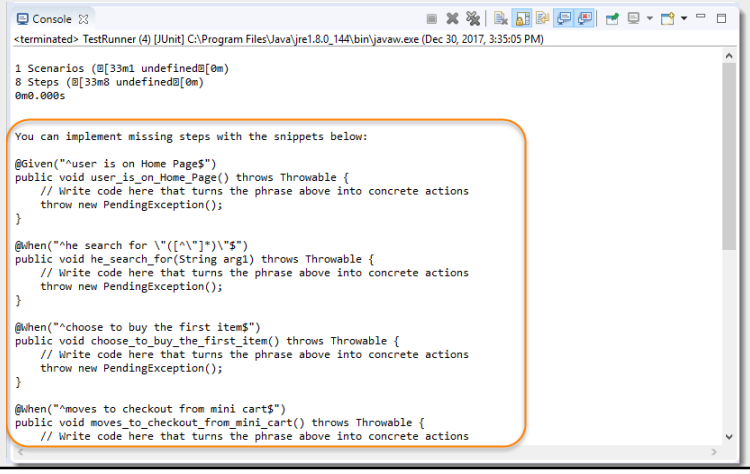
```
package runners;
import org.junit.runner.RunWith;
import cucumber.api.CucumberOptions;
import cucumber.api.junit.Cucumber;

@RunWith(Cucumber.class)
@CucumberOptions(
features = "src/test/resources/functionalTests"
)
public class TestRunner {
}
```

Note**: Make sure not to click on the **public static void main.

Business readable UI automation with cucumber

Write test code to Step file

1) To get the steps automatically generated, we need to execute ***TestRunner*** class. *Right click* on the *TestRunner* file and select ***Run As >> JUnit Test***. You would get the below result in the Eclipse Console.



Note**: Make sure not to click on the **public static void main.

Business readable UI automation with cucumber

Write test code to Step file

2) Create a ***New Package*** and name it as ***stepDefinitions*** by *right click* on the *src/test/java* and select ***New >> Package***.

3) Create a ***New Java Class*** and name it is as ***Steps*** by *right click* on the above created package and select ***New >> Class***.

4) Now copy all the steps created by Eclipse to this *Steps* file and start filling up these steps with Selenium Code. Take all the code from our Selenium Test file created in the ***End2End Test.*** *Steps* test file will look like this:

Note**: Make sure not to click on the **public static void main.

Business readable UI automation with cucumber

Write test code to Step file

5) We also need to make sure that the ***TestRunner*** file would also able to find the steps files.To achieve that we need to mention the path of the package, where we have kept all of our step definitions in *CucumberOptions.*

```
package runners;

import org.junit.runner.RunWith;

import cucumber.api.CucumberOptions;
import cucumber.api.junit.Cucumber;

@RunWith(Cucumber.class)
@CucumberOptions(
      features = "src/test/resources/functionalTests",
      glue= {"stepDefinitions"}
)

public class TestRunner {

}
```

Note*: By default Junit/Cucumber finds the test code in the* ***src/test/java*** *folder, this is why we just need to specify the package name for the cucumber glue.*

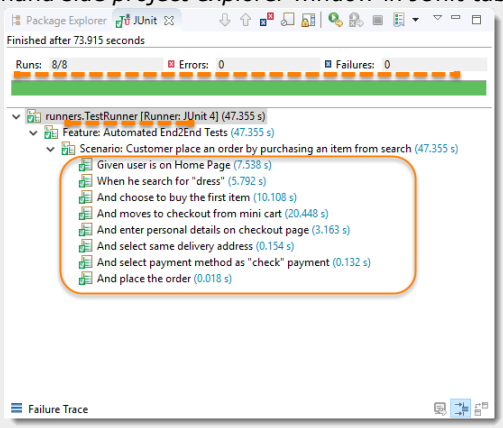
Business readable UI automation with cucumber

Run the Cucumber Test

Run as JUnit

Now we are all set to run the first Cucumber test. *Right Click* on ***TestRunner*** class and Click ***Run As >> JUnit Test.***

Cucumber will run the script the same way it runs in *Selenium WebDriver* and the result will be shown in the left hand side *project explorer window* in *JUnit* tab.



***Note**: By default Junit/Cucumber finds the test code in the **src/test/java** folder, this is why we just need to specify the package name for the cucumber glue.*

Summary

In this lesson, you have learnt :

- Business readable UI automation with cucumber