Specflow vs Cucumber

cucumber	specflow

Feature File

File Name: loginToGmail.feature **Feature:** Login to gmail Acccount **Scenario:** Valid Users **Given:** I am on gmail Login Page **When:** Enter username and password **And:** Click on Login button **Then:** User should be redirected to HomePage **And:** welcome message is displayed with first Name	**File Name: loginToGmail.feature** **Feature:** Login to gmail Acccount **Scenario:** Valid Users **Given** I am on gmail Login Page **When** Enter username and password **And** Click on Login button **Then** User should be redirected to Home Page **And** welcome message is displayed with first Name

Step Definition File

File Name: loginTOGmail.java

```java
public class loginToGmail {

@Given("^I am on gmail Login Page$")]
public void launchGmailLogin(){
//...
}

@When("^Enter username and password$")]
public void enterUserNamePassword(){
//...
}

@When("^Click on Login button$")]
public void clickOnLoginButton(){
//...
}

@Then("^User should be redirected to HomePage$")]
public void verifyHomepage(){
//...
}

@Then("^welcome message is displayed with first
Name$")]
public void verifyWelcomeMessage(){
}
//...

}
```

File Name: loginTOGmail.cs

```csharp
[Binding]

public class loginToGmail {

[Given(@"I am on gmail Login Page")]
public void launchGmailLogin(){
//...
}

[When(@"Enter username and password")]
public void enterUserNamePassword(){
//...
}

[When(@"Click on Login button")]
public void clickOnLoginButton(){
//...
}

[Then(@"User should be redirected to HomePage")]
public void verifyHomepage(){
//...
}

[Then(@"welcome message is displayed with first
Name")]
public void verifyWelcomeMessage(){
//...
}

}
```

Specflow vs Cucumber

Parameterization - Feature File

Parameterization - Examples

cucumber

```
Scenario Outline: Various Valid Users
Given: I am on gmail Login Page
When: Enter "<username>" and "<password>"
And: Click on Login button
Examples:
| username | password |
| username1 | password1 |
| username2 | password2 |
```

```java
@When("^Enter \"(.*)\" and \"(.*)\"$")]
public void enterUserNamePassword(String uName,
String passwd){
//...
}
```

specflow

```
Scenario Outline: Various Valid Users
Given I am on gmail Login Page
When Enter <username> and <password>
And Click on Login button
Examples:
| username | password |
| username1 | password1 |
| username2 | password2 |
```

```csharp
[When(@"Enter (.*) and (.*)")]
public void enterUserNamePassword(String uName,
String passwd){
//...
}
```

Parameterization – Table

cucumber

```
Scenario: Various Valid Users
Given: I am on gmail Login Page
When: Enter the user name and password
| Username | Password |
| username1 | password1 |
| username2 | password2 |
And: Click on Login button
```

```java
@When("^Enter the user name and password$")
public void enterUserNamePassword(DataTable
table){
-------------------------------------------
List<List<String>> data= table.row(0);
username = data.get(0).get(0)
passwd= data.get(0).get(1);
-------------------------------------------
List<List<String>> data= table.asList();
username = data.get(0).get(0)
passwd= data.get(0).get(1);
-------------------------------------------
List<Map<String,String>> data=
table.asMap(String.class, String.class);
username = data.get(0).get('userName');
passwd= data.get(1).get('password');
----------------


}
```

specflow

```
Scenario: Various Valid Users
Given I am on gmail Login Page
When Enter the user name and password
| Username | Password |
| username1 | password1 |
| username2 | password2 |
And Click on Login button
```

```csharp
[When(@"Enter the user name and password")]
public void enterUserNamePassword(Table table)
{
-------------------------------------------
foreach (var row in table.Rows){
dictionary.Add(row[0], row[1]);
}
address.Line1 = table.Rows[0]["Username"];
address.Line1 = table.Rows[0]["Password "];
-------------------------------------------
For(int i=0; i<table.RowCount;i++){
String Text = table.Rows[i]["UserName"]
String Text = table.Rows[i]["Password"]
}
-------------------------------------------
Table.ContainsColumn("Password")
-------------------------------------------
```

SpecFlow Assist Helpers
(techTalk.specflow.assit)
```csharp
var account = table.CreateInstance<Account>();
var products = table.CreateSet<Product>();
-------------------------------------------
```

Specflow vs Cucumber

cucumber	specflow

	``IEnumerable <dynamic> credentials =
	table.CreateDynamicSet();
	foreach(var users in credentials){
	}
	
	IEnumerable <dynamic> credentials =
	table.CreateDynamicInstance();
	foreach(var users in credentials){
	}
	}```

Set
```
ScenarioContext.Current["UserName"] = UserName1
ScenarioContext.Current.Add("UserName",UserName1);
```
Get
```
Var uname= ScenarioContext.Current["UserName1"];
Var uname=
ScenarioContext.Current.Get<int>("UserName");
```

Data can be share between Steps by using Context Injection

Hooks

cucumber	specflow

```@Before(Order = 2)```
```public void Before2() {```
```Run before executing each scenario```
```}```
 	
```@Before(Order = 1)```
```public void Before1() {```
```Run before executing each scenario```
```}```
 	
```@After```
```public void After() {```
```Run after executing each scenario```
```}```
 	
```***Order can be used in the cook hook```
	```[BeforeTestRun(Order = 1)]```
	```public static void BeforeTestRun1(){ … }```
	
	```[BeforeTestRun(Order = 2)]```
	```public static void BeforeTestRun2(){ … }```
	
	```[AfterTestRun]```
	```public static void AfterTestRun(){ … }```
	
	```[BeforeFeature]```
	```public static void BeforeFeature(){ … }```
	
	```[AfterFeature]```
	```public static void AfterFeature(){ … }```
	
	``[BeforeScenario]
		[Before]```
		```public void BeforeScenario(){ … }```
		
		``[AfterScenario]
			[Before]```
			```public void AfterScenario(){ … }```
			
			```[BeforeScenarioBlock]```
			```public void BeforeScenario(){ … }```
			
			```[AfterScenarioBlock]```
			```public void AfterScenario(){ … }```
			
			```[BeforeStep]```
			```public void BeforeStep(){ … }```

Specflow vs Cucumber

cucumber	specflow

	`[AfterStep]`
	`public void AfterStep(){ … }`
	
	`***Order can be used in all the hook`

Tag

`@SmokeTest @RegressionTest @DNA`
`Scenario: Successful Login`
`Given: This is a blank test`
 	
`@RegressionTest @DNA`
`Scenario: UnSuccessful Login`
`Given: This is a blank test`
 	
`@SmokeTest  @DNA`
`Scenario: Add a product to bag`
`Given: This is a blank test`
	`@SmokeTest @RegressionTest @DNA`
	`Scenario: Successful Login`
	`Given This is a blank test`
	
	`@RegressionTest @DNA`
	`Scenario: UnSuccessful Login`
	`Given This is a blank test`
	
	`@SmokeTest @DNA`
	`Scenario: Add a product to bag`
	`Given This is a blank test`

Run

<table>
<tr><td>

```
package cucumberJava;
import org.junit.runner.RunWith;
import cucumber.junit.Cucumber;
@RunWith(Cucumber.class)
@Cucumber.Options(
plugin = {"pretty",
"html:Folder_Name" ,
"json:Folder_Name/cucumber.json" ,
"junit:Folder_Name/cucumber.xml"
},
features = {"src/test/features"},
glue = "src/test/stepDeinition"
tags = {"@SmokeTest"}
monochrome=true
strict=false
)


public class runTest {
}
```

MonoChrome - readable console output

Strict-skip undefined steps from execution (default-false)

</td><td>

App.config

```
<specFlow>
  <unitTestProvider name="MsTest" />
</specFlow>
```

Test | Windows | Test Explorer

</td></tr>
<tr><td>

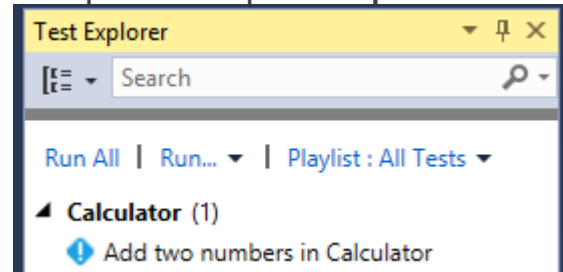`(Include plugin in POM.xml)`
```
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-compiler-plugin</artifactId>...
</plugin>
```

```
mvn clean install
mvn test
mvn test -Dcucumber.options=
"src/test/resources/functionalTests/End2End_Tests.
feature"
```

</td><td>

```
C:>> mstest.exe
/testcontainer:c:\test\test.UI.dll
/resultfile:c:\test\result.trx
```

trx >> xml format

</td></tr>
</table>

Specflow vs Cucumber

cucumber	specflow

```
mvn test -Dcucumber.options='--tags "@smoke and
@fast"'
mvn -Dtest=RunnerTest test (running Junit class)
```

Report

HTML

Feature Result for Build: 1

JSON

```
[
  {
    "id": "a-docstring-feature",
    "uri": "features/doc_string.feature",
    "keyword": "Feature",
    "name": "A DocString feature",
    "line": 1,
    "description": "",
    "elements": [
      {
        "id": "a-docstring-feature;",
        "keyword": "Scenario",
        "name": "",
        "line": 3,
        "description": "",
        "type": "scenario",
        "steps": [
          {
            "keyword": "Then ",
            "name": "I should fail with",
            "line": 4,
            "doc_string": {
              "content_type": "",
              "value": "a string",
              "line": 5
            },
          }
```

cucumber-reporting

```
<groupId>net.masterthought</groupId>
<artifactId>cucumber-reporting</artifactId>
```


HTML /XML

```
specflow.exe stepdefinitionreport SpecFlowTalk.csproj
/BinFolder:bin/debug /out:myTestResult.html

XML>>> /BinFolder:bin\debug /out:TestResult.xml
```
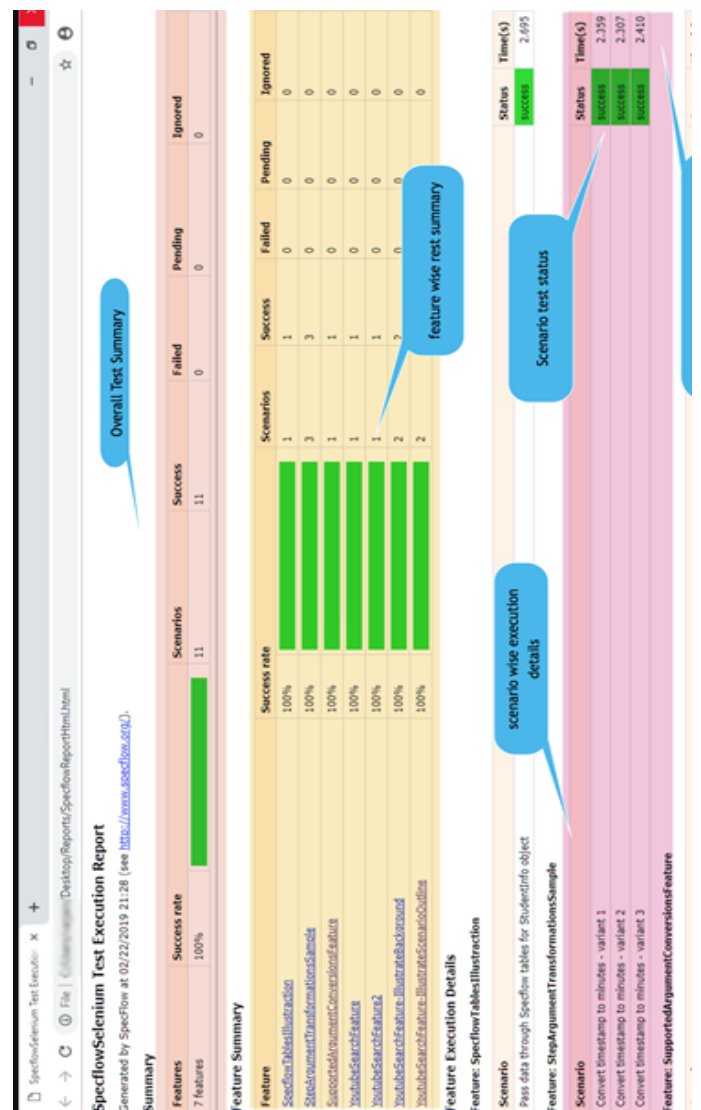
Specflow vs Cucumber

cucumber	specflow
Scope Binding	
	```[Scope(Tag = "mytag", Feature = "feature title", Scenario = "scenario title")]```

```
[Scope(Tag = "mytag", Feature = "feature title",
Scenario = "scenario title")]
[When(@"I perform a simple search on '(.*)'",
Scope(Tag = "controller"))]
public void WhenIPerformASimpleSearchOn(string
searchTerm)
{
var controller = new CatalogController();
actionResult = controller.Search(searchTerm);
}

[When(@"I perform a simple search on '(.*)'"),
Scope(Tag = "web")]
public void PerformSimpleSearch(string title)
{
selenium.GoToThePage("Home");
selenium.Type("searchTerm", title);
selenium.Click("searchButton");
}
```