Specflow vs Cucumber

cucumber | specflow

Feature File

cucumber

```
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
```

specflow

```
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

cucumber

```
File Name: loginTOGmail.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(){
}
//...

}
```

specflow

```
File Name: loginTOGmail.cs

[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 |
```

```
@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 |
```

```
[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
```

```
@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’);
```

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
```

```
[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”]
}
----------------------------------------
```

Specflow vs Cucumber

cucumber	specflow

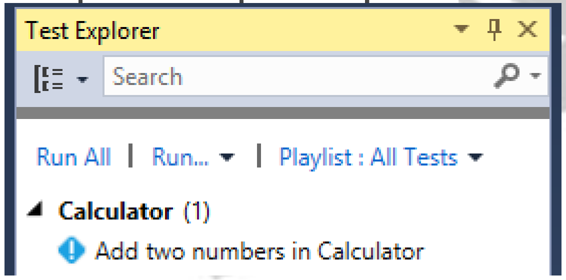
passwd= data.get(1).get('password');
----------------- 	
 	
}
	Table.ContainsColumn("Password")
	-- 	
	SpecFlow Assist Helpers
	(techTalk.specflow.assit)
	var account = table.CreateInstance<Account>();
	var products = table.CreateSet<Product>();
	-- 	
	
	IEnumerable <dynamic> credentials =
	table.CreateDynamicSet();
	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	

@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(){ … }

Specflow vs Cucumber

cucumber	specflow
	[BeforeScenarioBlock] public void BeforeScenario(){ … } [AfterScenarioBlock] public void AfterScenario(){ … } [BeforeStep] public void BeforeStep(){ … } [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	

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 {
	App.config
	
	<specFlow>
	  <unitTestProvider name="MsTest" />
	</specFlow>
	
	Test | Windows | Test Explorer
	
	Test Explorer
	Search
	Run All | Run... | Playlist : All Tests
	Calculator (1)
	Add two numbers in Calculator

Specflow vs Cucumber

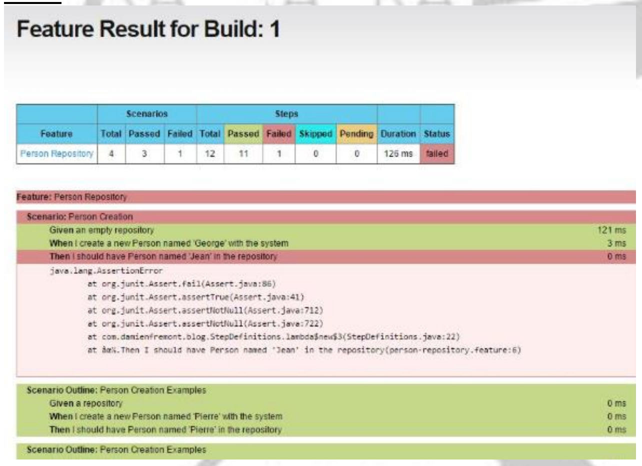
cucumber	specflow
}	
MonoChrome - readable console output **Strict**-skip undefined steps from execution (default-false)	

(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”`
 	
`mvn test -Dcucumber.options='--tags "@smoke and @fast"'`
`mvn -Dtest=RunnerTest test (running Junit class)`
	`C:>> mstest.exe /testcontainer:c:\test\test.UI.dll /resultfile:c:\test\result.trx`
	
	trx >> xml format

Report

cucumber	specflow
HTML	HTML /XML

HTML



Feature Result for Build: 1

	Scenarios			Steps						
Feature	Total	Passed	Failed	Total	Passed	Failed	Skipped	Pending	Duration	Status
Person Repository	4	3	1	12	11	1	0	0	126 ms	failed

```
Feature: Person Repository
  Scenario: Person Creation
      Given an empty repository                                          121 ms
      When I create a new Person named 'George' with the system            3 ms
      Then I should have Person named 'Jean' in the repository             0 ms
      java.lang.AssertionError
              at org.junit.Assert.fail(Assert.java:86)
              at org.junit.Assert.assertTrue(Assert.java:41)
              at org.junit.Assert.assertNotNull(Assert.java:712)
              at org.junit.Assert.assertNotNull(Assert.java:722)
              at com.damienfremont.blog.StepDefinitions.lambda$new$3(StepDefinitions.java:22)
              at âœ½.Then I should have Person named 'Jean' in the repository(person-repository.feature:6)

  Scenario Outline: Person Creation Examples
      Given a repository                                                   0 ms
      When I create a new Person named 'Pierre' with the system            0 ms
      Then I should have Person named 'Pierre' in the repository           0 ms

  Scenario Outline: Person Creation Examples
```

JSON

HTML /XML

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

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

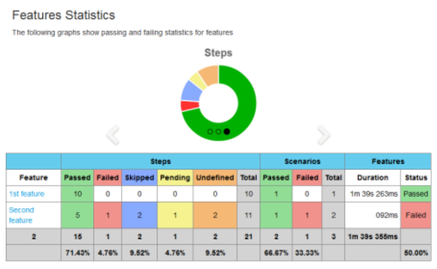
Specflow vs Cucumber

cucumber	specflow

```
[
  {
    "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

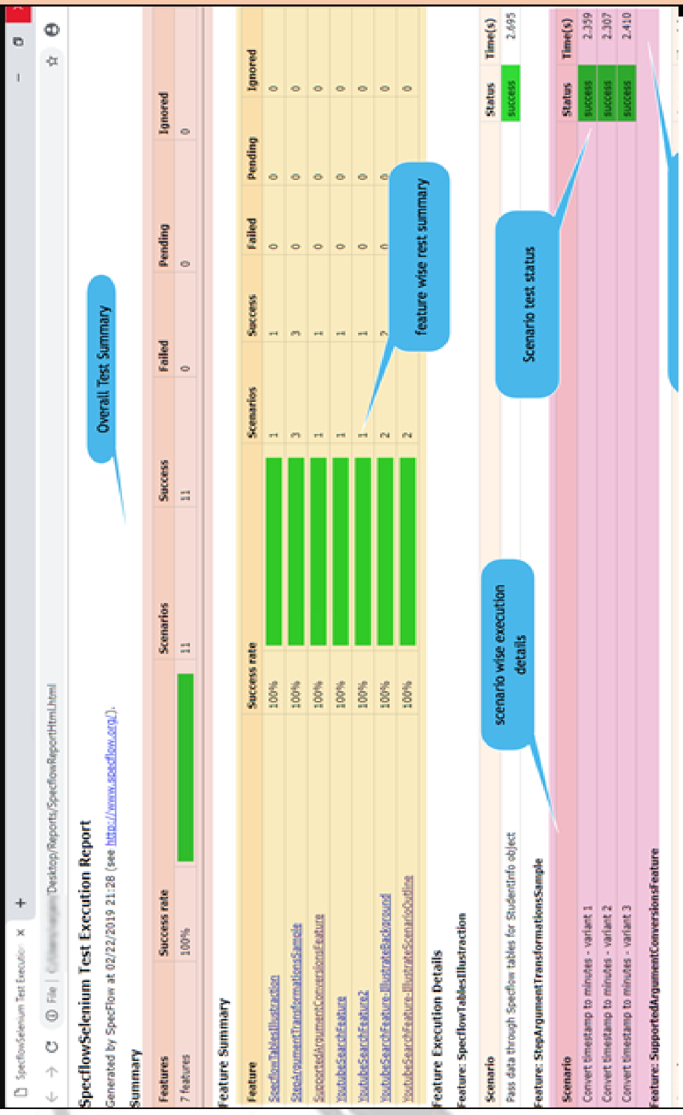
```
<groupId>net.masterthought</groupId>
<artifactId>cucumber-reporting</artifactId>
```



Features Statistics

The following graphs show passing and failing statistics for features

	Steps						Scenarios			Features	
Feature	Passed	Failed	Skipped	Pending	Undefined	Total	Passed	Failed	Total	Duration	Status
1st feature	10	0	0	0	0	10	1	0	1	1m 39s 263ms	Passed
Second feature	5	1	2	1	2	11	1	1	2	092ms	Failed
2	15	1	2	1	2	21	2	1	3	1m 39s 355ms	
	71.43%	4.76%	9.52%	4.76%	9.52%		66.67%	33.33%			50.00%



SpecflowSelenium Test Execution
File | Desktop/Reports/SpecflowReportHtml.html
Overall Test Summary
feature wise rest summary
scenario wise execution details
Scenario test status

SpecflowSelenium Test Execution Report

Generated by SpecFlow at 02/22/2019 21:28 (see http://www.specflow.org/).

Summary

Features	Success rate		Scenarios	Success	Failed	Pending	Ignored
7 features	100%	🟩	11	11	0	0	0

Feature Summary

Feature	Success rate		Scenarios	Success	Failed	Pending	Ignored
SpecflowTablesIllustraction	100%	🟩	1	1	0	0	0
StepArgumentTransformationsSample	100%	🟩	3	3	0	0	0
SupportedArgumentConversionsFeature	100%	🟩	1	1	0	0	0
YoutubeSearchFeature	100%	🟩	1	1	0	0	0
YoutubeSearchFeature2	100%	🟩	1	1	0	0	0
YoutubeSearchFeature-IllustrateBackground	100%	🟩	2	2	0	0	0
YoutubeSearchFeature-IllustrateScenarioOutline	100%	🟩	2				0

Feature Execution Details

Feature: SpecflowTablesIllustraction

Scenario	Status	Time(s)
Pass data through Specflow tables for StudentInfo object	success	2.695

Feature: StepArgumentTransformationsSample

Scenario	Status	Time(s)
Convert timestamp to minutes - variant 1	success	2.359
Convert timestamp to minutes - variant 2	success	2.307
Convert timestamp to minutes - variant 3	success	2.410

Feature: SupportedArgumentConversionsFeature

Specflow vs Cucumber

cucumber	specflow
Scope Binding	

	`[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");`
	`}`