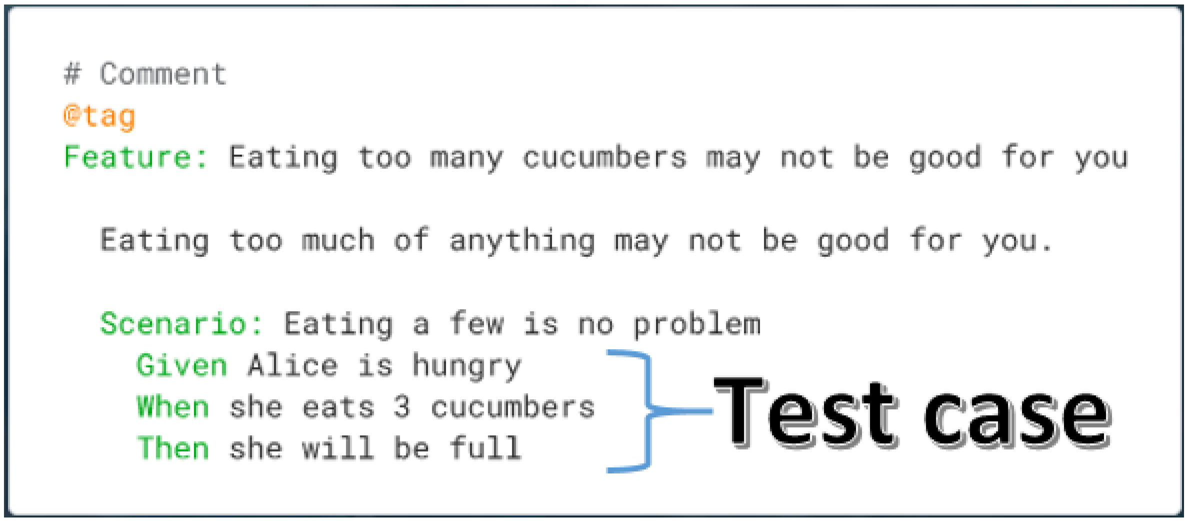
Way2Automation - Tutorial 1 – BDD Introduction

What you will Learn :

- What is BDD (Business driven development) ?
- Gherkin syntax
- E2E shopping cart Gherkin syntax
- Scenario keyword
- Feature keyword
- Linking Gherkin with java
- Cucumber
- Step Definition
- References

What is BDD and Gherkin syntax

Let us first understand what is BDD. Go to the website https://cucumber.io/

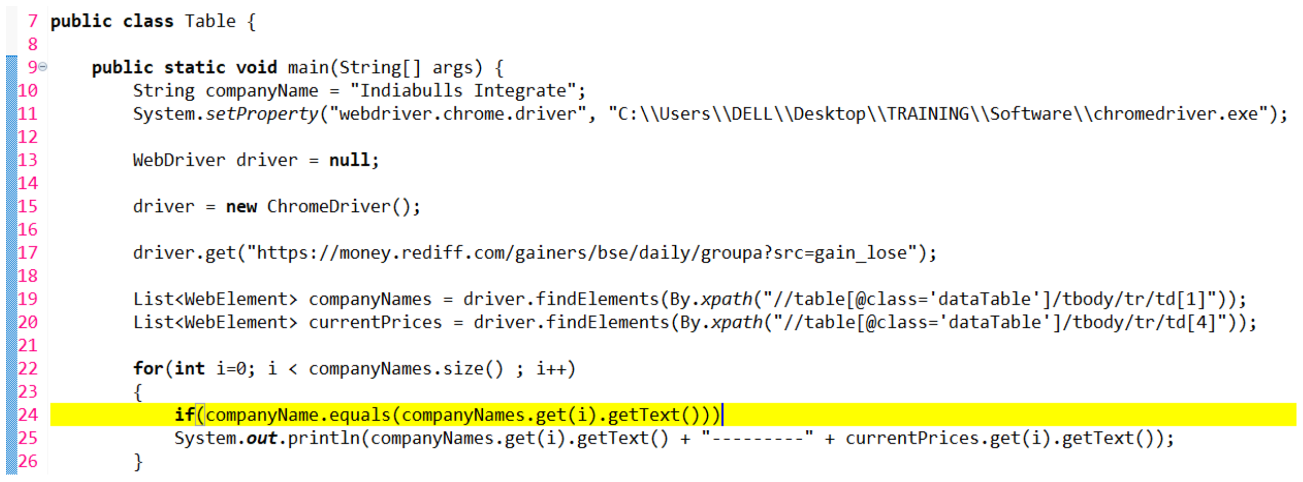


```
# Comment
@tag
Feature: Eating too many cucumbers may not be good for you

  Eating too much of anything may not be good for you.

  Scenario: Eating a few is no problem
    Given Alice is hungry
    When she eats 3 cucumbers
    Then she will be full
```

Look at above test case. We have described our test case in the form of a 'behaviour'.

So our automation test case will check whether 'Alice will be full if she eats 3 cucumbers'.

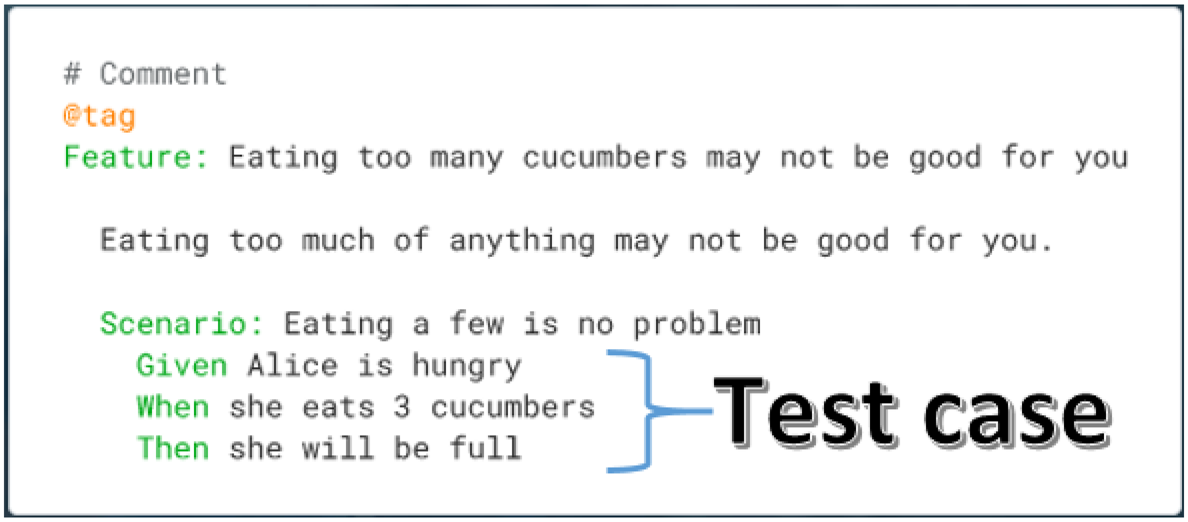
However, look at the below test case that was built using pure java syntax. To a client/end-user, the syntax looks to be slighlty technical and is not so easy to understand



```java
public class Table {

    public static void main(String[] args) {
        String companyName = "Indiabulls Integrate";
        System.setProperty("webdriver.chrome.driver", "C:\\Users\\DELL\\Desktop\\TRAINING\\Software\\chromedriver.exe");

        WebDriver driver = null;

        driver = new ChromeDriver();

        driver.get("https://money.rediff.com/gainers/bse/daily/groupa?src=gain_lose");

        List<WebElement> companyNames = driver.findElements(By.xpath("//table[@class='dataTable']/tbody/tr/td[1]"));
        List<WebElement> currentPrices = driver.findElements(By.xpath("//table[@class='dataTable']/tbody/tr/td[4]"));

        for(int i=0; i < companyNames.size() ; i++)
        {
            if(companyName.equals(companyNames.get(i).getText()))
            System.out.println(companyNames.get(i).getText() + "---------" + currentPrices.get(i).getText());
        }
```

So by looking at above, it is difficult for the client to understand what exactly you have automated, what test case feature has been automated. They can only guess what this test case might be doing. So the business scenarios are missing in above script. So we cannot say for sure what the above test case exactly does.

Instead, our test case can be described in this behaviour driven style (see below).



```
# Comment
@tag
Feature: Eating too many cucumbers may not be good for you

  Eating too much of anything may not be good for you.

  Scenario: Eating a few is no problem
    Given Alice is hungry
    When she eats 3 cucumbers
    Then she will be full
```

The above syntax is called as 'Gherkin' syntax and our client can easily understand it. The syntax closely matches with the English language (having 'Given', 'When', Then' etc keywords). Most lines in a Gherkin syntax start with one of the keywords (Given, When etc).

So you can understand the test case syntax as below:

Given *explains what you have*
When *you perform some action*

And *you perform additional action*
Then *what is the result or outcome of that action*

Below is another example of a test case in Gherkin syntax.

Scenario: All shopping done

Given I am out shopping

And I have eggs

And I have milk

And I have butter

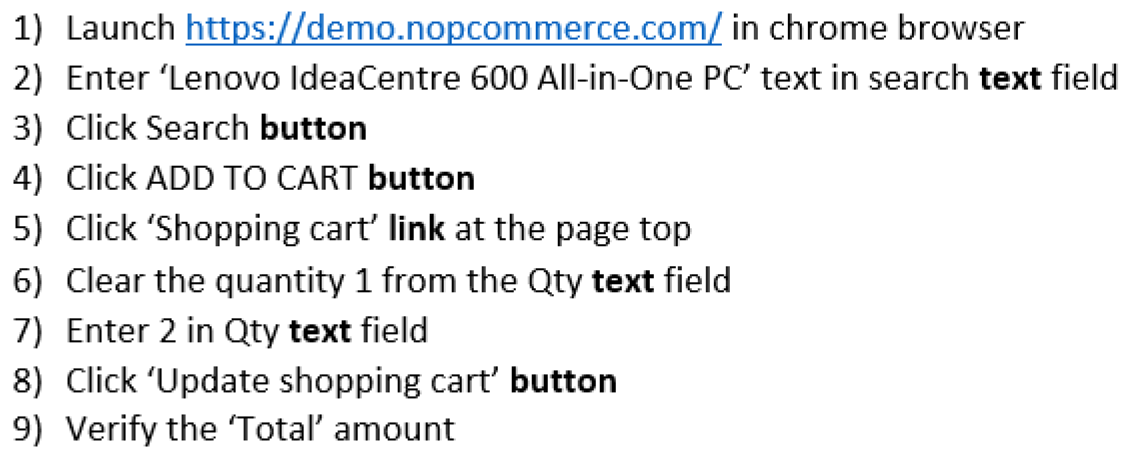
When I check my shopping list

Then I should not have anything left

So you can describe your test scenario using Gherkin syntax and you can link these lines with actual java code.

E2E shopping cart Gherkin syntax

Consider below end to end test case:



1) Launch https://demo.nopcommerce.com/ in chrome browser
2) Enter 'Lenovo IdeaCentre 600 All-in-One PC' text in search **text** field
3) Click Search **button**
4) Click ADD TO CART **button**
5) Click 'Shopping cart' **link** at the page top
6) Clear the quantity 1 from the Qty **text** field
7) Enter 2 in Qty **text** field
8) Click 'Update shopping cart' **button**
9) Verify the 'Total' amount

Let us try to convert above steps into Gherkin syntax. So, below is the business representation in BDD/cucumber world which is self-explanatory.

Recall that 'And' keyword is used to concatenate additional actions.

Scenario: E2E automation

Given I launch an e-commerce website

When I enter ‘Lenovo IdeaCentre 600 All-in-One PC’ in search field

And I search the item

And I add item to shopping cart

And I go to shopping cart

And I enter 2 in quantity field

And I update my shopping cart

Then 'Total' amount should be $1000

Scenario keyword

As you must have noticed in above test cases, in addition to other keywords, we also used ‘Scenario’ keyword. A ‘Scenario’ describes intended behaviour of the test case. In other words, a ‘Scenario’ describes *what*, not *how*.

Consider below test case.

The ‘Scenario’ is describing the *what* viz *what the free subscribers will see*. It does not describe *how the free subscribers will see only free articles*.

```
Scenario: Free subscribers see only the free articles
  Given users with a free subscription can access "FreeArticle1" but not "PaidArticle1"
  When I type "freeFrieda@example.com" in the email field
  And I type "validPassword123" in the password field
  And I press the "Submit" button
  Then I see "FreeArticle1" on the home page
  And I do not see "PaidArticle1" on the home page
```

Feature keyword

‘Feature’ represents a Test Suite. A test suite can have multiple test cases. Each scenario represents one test case. So if you consider below example, we have a ‘Feature’ that contains 2 test cases or 2 scenarios.

```
Feature: Subscribers see different sets of stock images based on their subscription level

Scenario: Free subscribers see only the free articles
  Given users with a free subscription can access "FreeArticle1" but not "PaidArticle1"
  When I type "freeFrieda@example.com" in the email field
  And I type "validPassword123" in the password field
  And I press the "Submit" button
  Then I see "FreeArticle1" on the home page
  And I do not see "PaidArticle1" on the home page

Scenario: Subscriber with a paid subscription can access "FreeArticle1" and "PaidArticle1"
  Given I am on the login page
  When I type "paidPattya@example.com" in the email field
  And I type "validPassword123" in the password field
  And I press the "Submit" button
  Then I see "FreeArticle1" and "PaidArticle1" on the home page
```

Linking Gherkin with java

Let us look at below test case again. We will ultimately be linking all our Gherkin syntax with java code, as shown below.

Scenario: E2E automation

Given I launch an e-commerce website linked to
<some java codeto launch website> .get('https://demo.nopcommerce.com/')

When I enter ‘Lenovo IdeaCentre 600 All-in-One PC’ in search field linked to
<some java code to type text in the search box> .sendkeys('Lenovo IdeaCentre 600 All-in-One PC')

And I search the item linked to
<some java code to click the search button> .click()

And I add item to shopping cart linked to
<some java code to add item to cart> ('.product-box-add-to-cart-button').click()

And I go to shopping cart linked to
<some java code to click the shopping cart> ('.cart-label').click()

And I enter 2 in quantity field linked to
<some java code to type numeric 2 in qty field> .sendkeys('2')

And I update my shopping cart linked to
<some java code to click and update the cart> ("[value='Update shopping cart']").click()

Then 'Total' amount should be $1000 linked to
<some java code to validate the amount> ('.product-subtotal').assertEquals('$1,000.00')

However, we will not share the java code with end user, we will only share the BDD (behaviour driven development) Gherkin syntax with client.

Cucumber

As we have seen above, we describe the desired behaviour in a test case and based upon this behaviour, we drive our development. By development, we mean writing the actual java code and linking it to behaviour.

That’s the reason we call it as ‘Behaviour Driven Development’

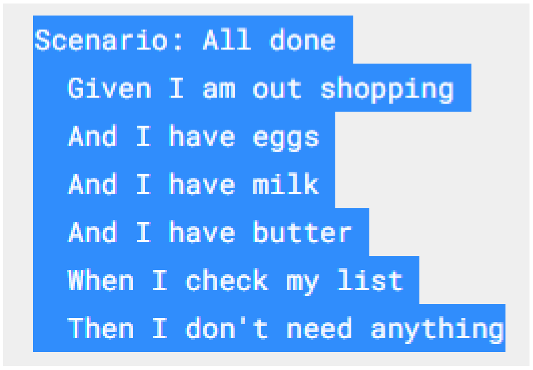
```
Scenario: E2E automation
  Given I launch an e-commerce website
  When I enter ‘Lenovo IdeaCentre 600 All-in-One PC’ in search field
  And I search the item
  And I add item to shopping cart
  And I go to shopping cart
  And I enter 2 in quantity field
  And I update my shopping cart
  Then 'Total' amount should be $1000
```

Cucumber is one of the tool which supports this BDD framework.

Step Definition

Now, what is a step? Each line of a scenario is a step.

So basically, all the below are steps. To define these steps would mean to write the actual java code.

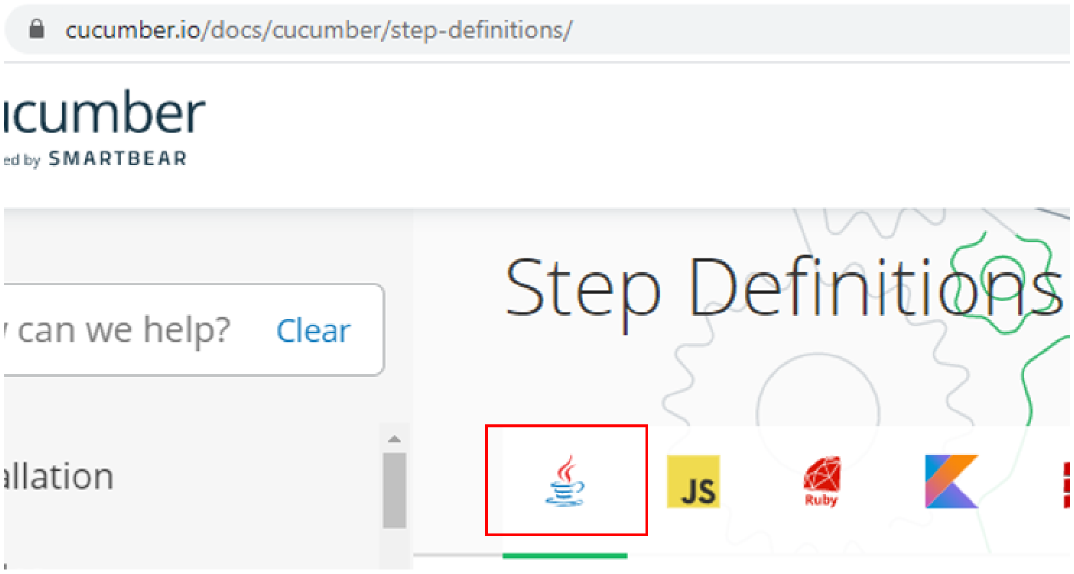


```
Scenario: All done
  Given I am out shopping
  And I have eggs
  And I have milk
  And I have butter
  When I check my list
  Then I don't need anything
```

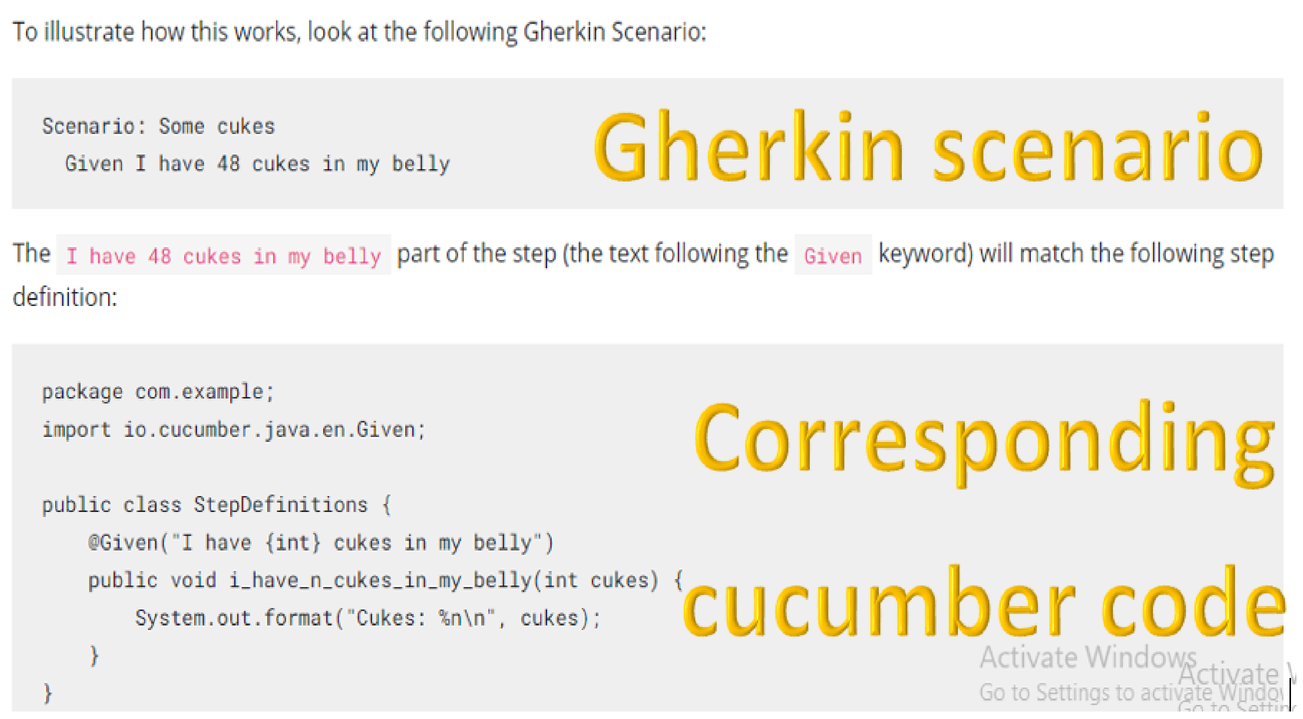
In 'Behaviour Driven Development', we write our java code inside 'Step Definition' file. The link-up depends upon the programming language you are using. Different programming languages have different syntaxes to link.

To see a typical step definition file for cucumber/java combination, go to the link https://cucumber.io/docs/cucumber/step-definitions/

Select Java image



You would see an example of Gherkin and the corresponding java syntax, see below


Gherkin scenario
Corresponding cucumber code
Activate Windows
Go to Settings to activate Windows.

To illustrate how this works, look at the following Gherkin Scenario:

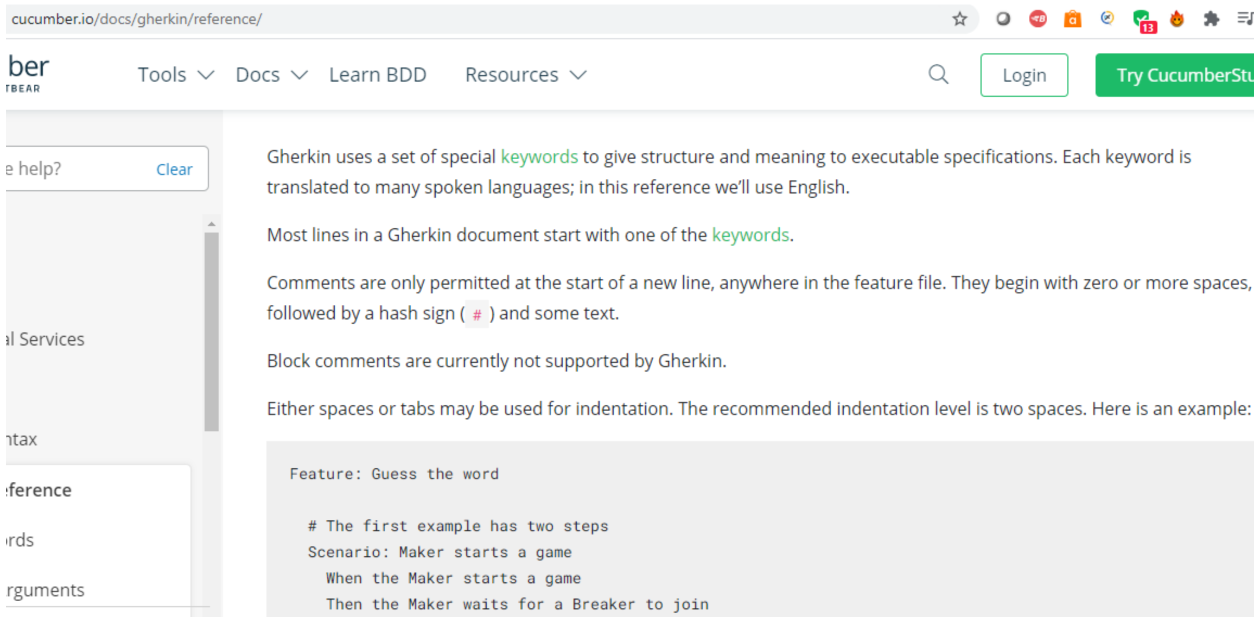
```
Scenario: Some cukes
  Given I have 48 cukes in my belly
```

The `I have 48 cukes in my belly` part of the step (the text following the `Given` keyword) will match the following step definition:

```
package com.example;
import io.cucumber.java.en.Given;

public class StepDefinitions {
    @Given("I have {int} cukes in my belly")
    public void i_have_n_cukes_in_my_belly(int cukes) {
        System.out.format("Cukes: %n\n", cukes);
    }
}
```

References

You can refer the link https://cucumber.io/docs/gherkin/reference/ to read more about Gherkin syntax



Gherkin uses a set of special keywords to give structure and meaning to executable specifications. Each keyword is translated to many spoken languages; in this reference we'll use English.

Most lines in a Gherkin document start with one of the keywords.

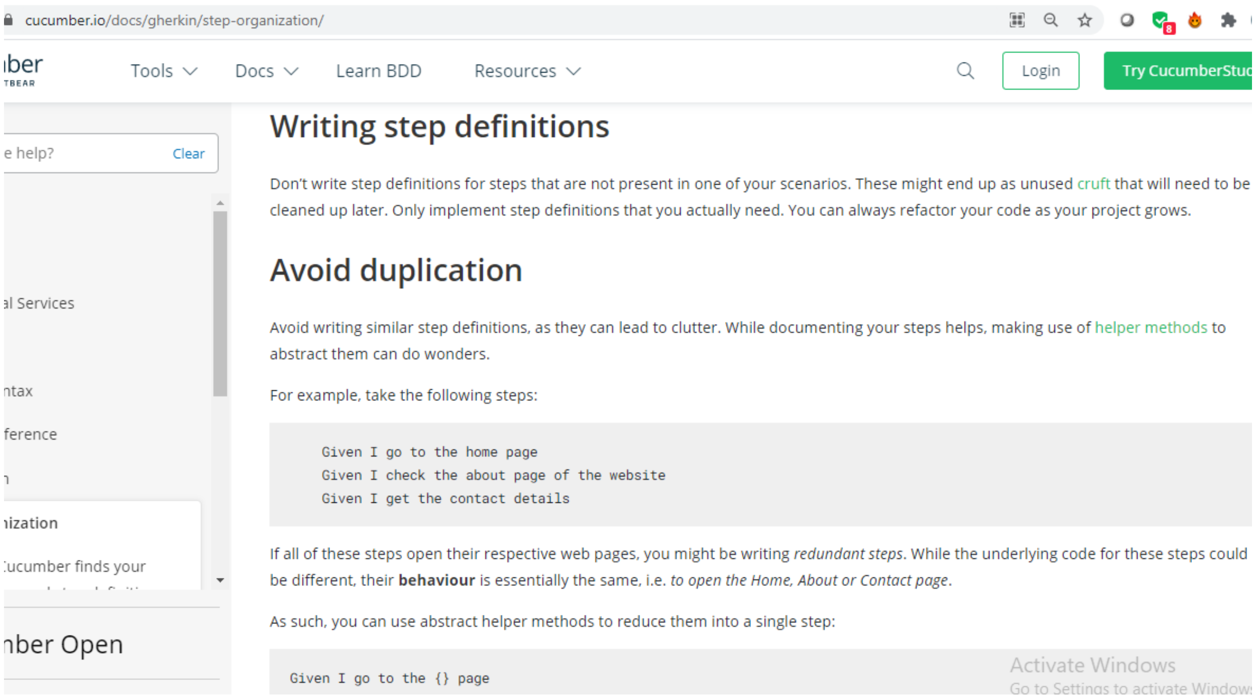
Comments are only permitted at the start of a new line, anywhere in the feature file. They begin with zero or more spaces, followed by a hash sign (`#`) and some text.

Block comments are currently not supported by Gherkin.

Either spaces or tabs may be used for indentation. The recommended indentation level is two spaces. Here is an example:

```
Feature: Guess the word

  # The first example has two steps
  Scenario: Maker starts a game
    When the Maker starts a game
    Then the Maker waits for a Breaker to join
```

You can refer the link https://cucumber.io/docs/gherkin/step-organization/ to read more about how to group and write your step definition file


cucumber.io/docs/gherkin/step-organization/

ıber

Tools Docs Learn BDD Resources Login Try CucumberStuc

e help? Clear

al Services

ntax

ference

nization

Cucumber finds your

nber Open

Writing step definitions

Don't write step definitions for steps that are not present in one of your scenarios. These might end up as unused cruft that will need to be cleaned up later. Only implement step definitions that you actually need. You can always refactor your code as your project grows.

Avoid duplication

Avoid writing similar step definitions, as they can lead to clutter. While documenting your steps helps, making use of helper methods to abstract them can do wonders.

For example, take the following steps:

```
Given I go to the home page
Given I check the about page of the website
Given I get the contact details
```

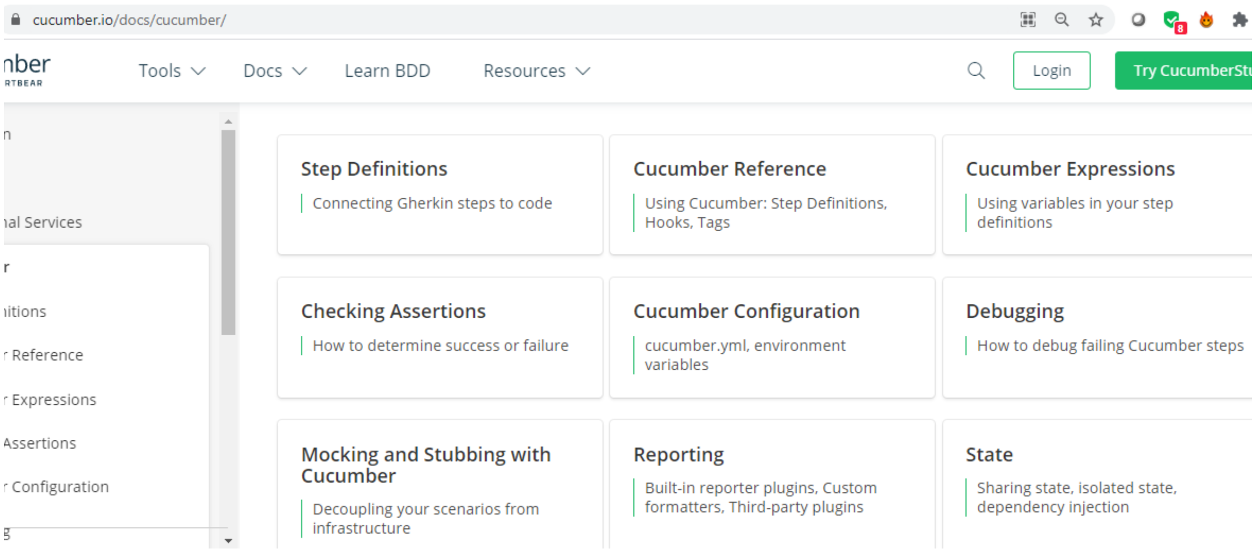
If all of these steps open their respective web pages, you might be writing *redundant steps*. While the underlying code for these steps could be different, their **behaviour** is essentially the same, i.e. *to open the Home, About or Contact page*.

As such, you can use abstract helper methods to reduce them into a single step:

```
Given I go to the {} page
```

Activate Windows
Go to Settings to activate Windows

You can also refer https://cucumber.io/docs/cucumber/ to further dig into various other cucumber related documents


cucumber.io/docs/cucumber/

nber

Tools Docs Learn BDD Resources Login Try CucumberSt

n

al Services

r

itions

r Reference

r Expressions

Assertions

r Configuration

g

Step Definitions Connecting Gherkin steps to code	**Cucumber Reference** Using Cucumber: Step Definitions, Hooks, Tags	**Cucumber Expressions** Using variables in your step definitions
Checking Assertions How to determine success or failure	**Cucumber Configuration** cucumber.yml, environment variables	**Debugging** How to debug failing Cucumber steps
Mocking and Stubbing with Cucumber Decoupling your scenarios from infrastructure	**Reporting** Built-in reporter plugins, Custom formatters, Third-party plugins	**State** Sharing state, isolated state, dependency injection

Thank you for reading!