Gherkin Language: Format, Syntax & Gherkin Test in Cucumber

What is Gherkin Language?

Gherkin is a business readable language which helps you to describe business behavior without going into details of implementation. It is a domain specific language for defining tests in Cucumber format for specifications. It uses plain language to describe use cases and allows users to remove logic details from behavior tests.

The text in Gherkin langauge acts as documentation and skeleton of your automated tests. Gherkin format is based on TreeTop Grammar which exists in 37+ languages. Therefore you can write your gherkin in 37+ spoken languages.

This script serves two primary purposes:

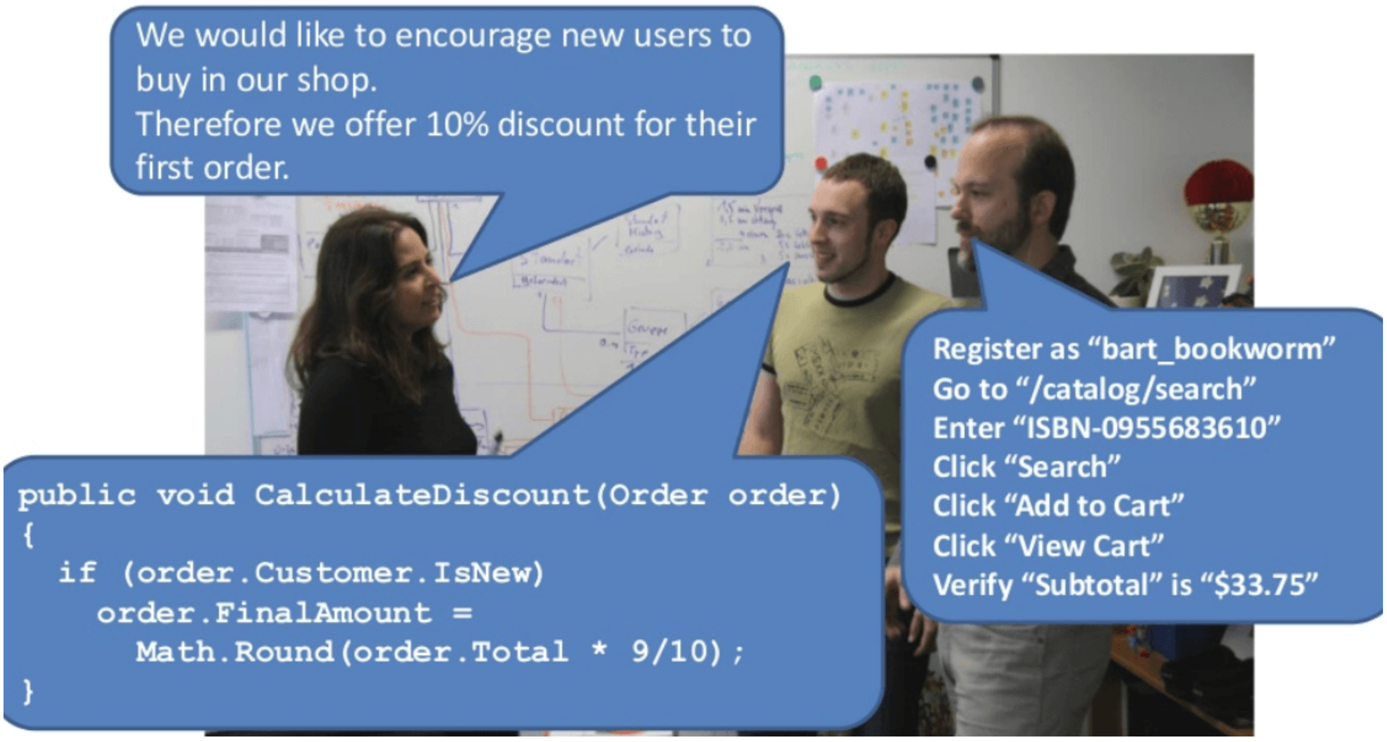
- Documents user scenarios
- Writing an automated test (BDD)

In this Gherkin tutorial, you will learn

Why Gherkin?

The need for Gherkin can be easily explained by following images

Before Gherkin



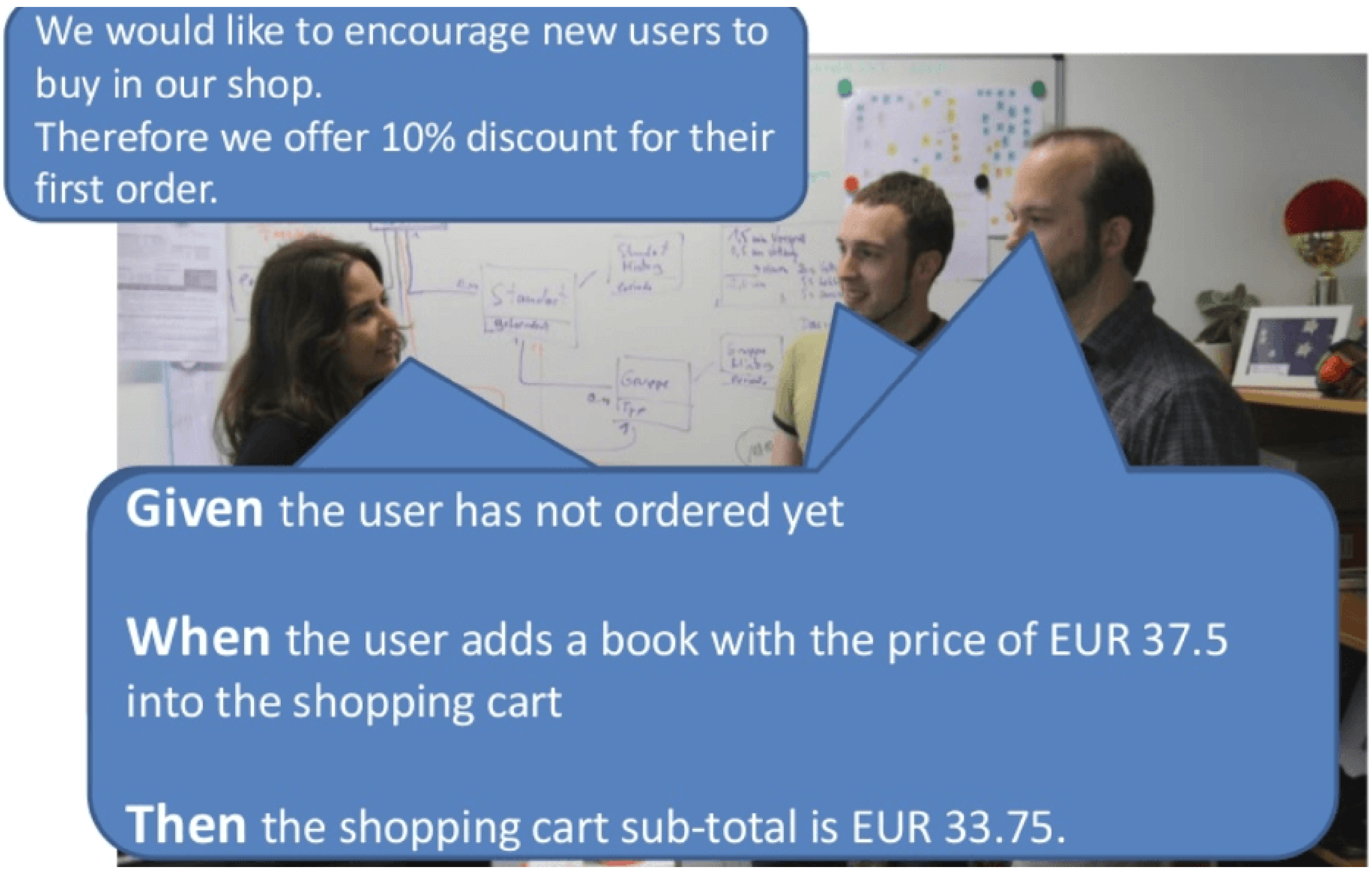
We would like to encourage new users to buy in our shop.
Therefore we offer 10% discount for their first order.

```
public void CalculateDiscount(Order order)
{
  if (order.Customer.IsNew)
    order.FinalAmount =
      Math.Round(order.Total * 9/10);
}
```

Register as "bart_bookworm"
Go to "/catalog/search"
Enter "ISBN-0955683610"
Click "Search"
Click "Add to Cart"
Click "View Cart"
Verify "Subtotal" is "$33.75"

(/images/tensorflow/082918_1505_WhatisGherk1.png)

After Gherkin


We would like to encourage new users to buy in our shop.
Therefore we offer 10% discount for their first order.
Given the user has not ordered yet
When the user adds a book with the price of EUR 37.5 into the shopping cart
Then the shopping cart sub-total is EUR 33.75.

(/images/tensorflow/082918_1505_WhatisGherk2.png)

Gherkin Syntax

Gherkin is line-oriented language just like YAML and Python. Each line called step and starts with keyword and end of the terminals with a stop. Tab or space are used for the indentation.

In this script, a comment can be added anywhere you want, but it should start with a # sign. It read each line after removing Ghrekin's keywords as given, when, then, etc.

Typical Gherkin steps look like:

Gherkin Scripts: connects the human concept of cause and effect to the software concept of input/process/output.

Gherkin Syntax:

```
Feature: Title of the Scenario
Given [Preconditions or Initial Context]
When [Event or Trigger]
Then [Expected output]
```

A Gherkin document has an extension .feature and simply just a test file with a fancy extension. Cucumber reads Gherkin document and executes a test to validate that the software behaves as per the Gherkin syntax.

Important Terms used in Gherkin

- Feature
- Background
- Scenario
- Given
- When
- Then
- And
- But
- Scenario Outline Examples

The naming convention is used for feature name. However, there is no set rules in Cucumber about names.

Feature:

The file should have extension .feature and each feature file should have only one feature. The feature keyword being with the Feature: and after that add, a space and name of the feature will be written.

Scenario:

Each feature file may have multiple scenarios, and each scenario starts with Scenario: followed by scenario name.

Background:

Background keyword helps you to add some context to the scenario. It can contain some steps of the scenario, but the only difference is that it should be run before each scenario.

Given:

The use of Given keyword is to put the system in a familiar state before the user starts interacting with the system. However, you can omit writing user interactions in Given steps if Given in the "Precondition" step.

Syntax:

```
Given
```

```
Given - a test step that defines the 'context
Given I am on "/."
```

When:

When the step is to define action performed by the user.

Syntax:

```
When
```

```
A When - a test step that defines the 'action' performed
When I perform "Sign In."
```

Then:

The use of 'then' keyword is to see the **outcome** after the action in when step. However, you can only verify noticeable changes.

Syntax:

```
 Then
```

```
Then - test step that defines the 'outcome.'
Then I should see "Welcome Tom."
```

And & But

You may have multiple given when or Then.

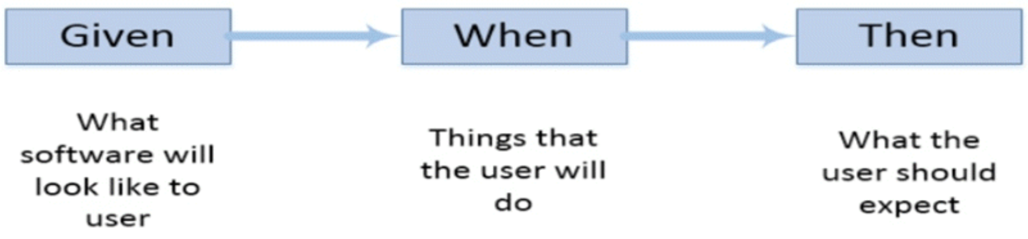
Syntax:

```
But
```

```
A But - additional test step which defines the 'action' 'outcome.'
But I should see "Welcome Tom."
```

```
And - additional test step that defines the 'action' performed
And I write  "EmailAddress" with "Tomjohn@gmail.com (mailto:Tomjohn@gmail.
com)."
```

Given, When, Then, and, but are test steps. You can use them interchangeably. The interpreter doesn't display any error. However, they will surely not make any 'sense' when read.


Given
When
Then
What software will look like to user
Things that the user will do
What the user should expect

(/images/tensorflow/082918_1505_WhatisGherk3.png)
Important Terms used in Gherkin

```
Given The login page is opening
When I input username, password and click the Login button
Then I am on the Homepage
```

Gherkin Examples

Example 1:

```
Feature:  Login functionality of social networking site Facebook.
Given:  I am a facebook user.
When: I enter username as username.
And I enter the password as the password
Then I should be redirected to the home page of facebook
```

The scenario mentioned above is of a feature called user login.

All the words written in bold are Gherkin keywords.

Gherkin will analyze each step written in the step definition file. Therefore, the steps are given in the feature file and the step definition file should match.

Example 2:

```
Feature: User Authentication Background:
Given the user is already registered to the website Scenario:
Given the user is on the login page
When the user inputs the correct email address
And the user inputs the correct password
And the user clicks the Login button
Then the user should be authenticated
And the user should be redirected to their dashboard
And the user should be presented with a success message
```

Best practices of using Gherkin

- Each scenario should execute separately
- Every feature should able to be executed along
- Steps information should be shown independently
- Connect your Scenario's with your requirements
- Keep a complete track of what scenarios should be included in a requirement document
- Create modular and easy to understand steps
- Try to combine all your common scenarios

Advantages of Gherkin

- Gherkin is simple enough for non-programmers to understand
- Programmers can use it as a very solid base to start their tests
- It makes User Stories easier to digest
- Gherkin script can easily understand by business executives and developers
- Gherkin Testing targets the business requirements
- A significant proportion of the functional specifications is written as user stories
- You don't need to be expert to understand the small Gherkin command set
- Gherkin Test cases link acceptance tests directly to automated tests
- Style of writing tests cases are easier to reuse code in other tests

Disadvantages of Gherkin

- It requires a high level of business engagement and collaborations
- May not work well in all scenarios
- Poorly written tests can easily increase test-maintenance cost

Summary:

- Gherkin is the format for cucumber specifications
- Gherkin is line-oriented language just like YAML and Python (/python-tutorials.html)
- Gherkin Scripts connects the human concept of cause and effect to the software concept of input/process and output
- Feature, Background, Scenario, Given, When, Then, And But are importantly used in Gherkin
- In Gherkin, each scenario should execute separately
- The biggest advantage of Gherkin is simple enough for non-programmers to understand
- Gherkin Test may not work well in all type of scenarios

Prev (/cucumber-basics.html) **Report a Bug** Next (/your-first-cucumber-script.html)

Cucumber Tutorials

About

About Us (/about-us.html)
Advertise with Us (/advertise-us.html)
Write For Us (/become-an-instructor.html)
Contact Us (/contact-us.html)

Career Suggestion

SAP Career Suggestion Tool (/best-sap-module.html)
Software Testing as a Career (/software-testing-career-complete-guide.html)

Interesting

eBook (/ebook-pdf.html)
Blog (/blog/)
Quiz (/tests.html)
SAP eBook (/sap-ebook-pdf.html)

Execute online

Execute Java Online (/try-java-editor.html)
Execute Javascript (/execute-javascript-online.html)
Execute HTML (/execute-html-online.html)
Execute Python (/execute-python-online.html)