Our framework is Cucumber that is a testing framework that supports Behavior Driven Development (BDD). It is written in plain English text called Gherkin. It is defined as a scenario of inputs, actions and outcomes. There are business and implementation layers on framework.

In order to achieve that we ***Converted Selenium Test into Cucumber BDD Style*** test and Cucumber understands [***Gherkin***](https://toolsqa.com/cucumber/gherkin-keywords/) language, read [***Feature***](https://toolsqa.com/cucumber/cucumber-jvm-feature-file/) files and execute code written in [***Step Definition***](https://toolsqa.com/cucumber/step-definition/) files with the help of [***Cucumber Options***](https://toolsqa.com/cucumber/data-tables-in-cucumber/) specified in [***TestRunner***](https://toolsqa.com/cucumber/junit-test-runner-class/).

Framework based on POM.

*Page Object Model Framework* which is also known as *Page Object Design Pattern* or *Page Objects*. The main advantage of *Page Object Model* is that if the UI or any HTML object changes for any page, the test does not need any fix. Only the code within the page objects will be impacted but it does not have any impact to the test.

**FRAMEWORK** consist of:

* Configuration.properties (to store sensitive/reusable data-URL, browser, credentials etc)
* pom.xml file (we are storing all our dependencies and plugins)
* feature file (The test is written in plain English which is common to all the domains of your project team.This test is structured that makes it capable of being read in an automated way. There by creating automation tests at the same time while describing the scenario. Inside feature file we have: Background, Scenario, Scenario Outline with Examples keyboard that support DDT)

**Source folder is splitted by source->main->:**

**Packages:**

**Cucumber:**

***[Sharing Test Context with PicoContainer](https://toolsqa.com/selenium-cucumber-framework/sharing-test-context-between-cucumber-step-definitions/)***

Scenario in Cucumber is a series of steps which gets executed one after one. Each step in scenario may have some state which can be required by other step in the scenario. Cucumber supports several *Dependency Injection* (*DI*) Containers – it simply tells a DI container to instantiate your step definition classes and wire them up correctly. One of the supported DI containers is *PicoContainer*, which helps in sharing context between steps.

[***Sharing Scenario Context***](https://toolsqa.com/selenium-cucumber-framework/share-data-between-steps-in-cucumber-using-scenario-context/)

*Scenario Context* is a class which holds the test data information specifically. It actually use the *Test Context* to travel the information between various steps. With in this *ScenarioContext* class, you can create any number of fields to store any form of data. It stores the information in the key value pair and again, value can be of any type. It can store *String, Boolean,* Integer or may be a *Class*.

**dataProviders:**

[***Config File Reader***](https://toolsqa.com/selenium-cucumber-framework/read-configurations-from-property-file/)

It is dangerous to store hard coded values in the project, also it is against the coding principles. And so far we have been using a lot of hard coded values in our code. With the help of properties file, we will be eliminating these hard coded value one by one.

* ***getBrowserWindowSize()*** : Retrieve the property using getProperty method of Properties class. Null check is performed and in case of null by default value is returned as true. In case of not null, String value is parsed to Boolean.
* ***getEnvironment()*** : EnvironmentType.Local is returned in case of Null and if the value is equal to Local. Which means that in case of missing environment property, execution will be carried on local machine.
* ***getBrowser()*** : Default value is returned as DriverType.Chrome in case of Null. Exception is thrown if the value does not match with anything.

[***JSON Data Reader***](https://toolsqa.com/selenium-cucumber-framework/data-driven-testing-using-json-with-cucumber/)

The simplest explanation of [*data-driven testing*](https://toolsqa.com/selenium-webdriver/data-driven-testing-excel-poi/) is this: data that is external to your functional tests is loaded and used to extend your automated test cases. To be sure that the application works as expected you can alter the test to accept variables, entering those variables into the data fields. Data driven can be done using *Excel, XML, JSON* etc but in this series of *Cucumber Automation Framework*, we will use *JSON* to pass data to tests.

**Enums:**

**Driver Type**

**Environment Type**

**managers:**

[***Page Object Manager***](https://toolsqa.com/cucumber/junit-test-runner-class/)

Duty of the *Page Object Manger* is to create the page’s object and also to make sure that the same object should not be created again and again. But to use single object for all the step definition files.

[***File Reader Manager***](https://toolsqa.com/selenium-cucumber-framework/read-configurations-from-property-file/)

Sometimes it’s appropriate to have exactly one instance of a class. These are accessed by disparate objects throughout a software system, and therefore require a global point of access. In our case, we have *ConfigReaderFile*, which should be accessed globally. But later on in this *Selenium Cucumber Framework* series we will be having many more file readers. So it is better to have a File Reader Manager above all the File Readers. And it is better to make the manager class as *singleton*.

[***WebDriver Manager***](https://toolsqa.com/selenium-cucumber-framework/design-webdriver-manager/)

**We are getting access to driver.**

**Page Objects**

***Selenium PageFactory*** : Page Factory is an inbuilt Page Object Model concept for Selenium WebDriver and it is very optimized. To learn more on Page Factory, please visit our tutorial [***Page Object Pattern using Selenium PageFactory***](https://www.toolsqa.com/selenium-webdriver/page-factory-in-selenium/).

***PageFactory*** is used to ***Initialize Elements*** of a Page class without having to use ‘FindElement‘ or ‘FindElements‘. Annotations can be used to supply descriptive names of target objects to improve code readability.

### ***@FindBy Annotation***

**selenium:**

[***Wait Utility for Ajax Wait***](https://toolsqa.com/selenium-cucumber-framework/handle-ajax-call-using-javascriptexecutor-in-selenium/)

***AJAX*** is a technique to do an XMLHttpRequest (out of band Http request) from a web page to the server and send/retrieve data to be used on the web page. ***AJAX stands for Asynchronous Javascript And XML***.

***Means: Ajax is way for the client-side browser to communicate with the server (for example: retrieve data from a database) without having to perform a page refresh***.

***JQuery*** (website) is a javascript framework that makes working with the DOM easier by building lots of high level functionality that can be used to search and interact with the DOM. ***Part of the functionality of jQuery implements a high-level interface to do AJAX requests***.

* JQuery is a lightweight client side scripting library while AJAX is a combination of technologies used to provide asynchronous data transfer
* JQuery and AJAX are often used in conjunction with each other
* JQuery is primarily used to modify data on the screen dynamically and it uses AJAX to retrieve data that it needs without changing the current state of the displayed page

**Sorce->test->java**

**Features**

**Drivers**

**Runners**

We are triggering framework form runners class.

Runners class ALLOWS TO RUN DIFFERENT TYPES OF TESTING, SUITS BY HELP OF TAGS.

FEATURE: path of feature file

GLUE: path of step definitions

[*Cucumber Report Plugins*](https://toolsqa.com/selenium-cucumber-framework/cucumber-reports/):

[*Extend Reports with Screenshots*](https://toolsqa.com/selenium-cucumber-framework/cucumber-extent-report/)

Last chapter of [*Cucumber Reports*](https://toolsqa.com/selenium-cucumber-framework/cucumber-reports/)  we got to know about the Plugins which are provided by Cucumber itself to generate various kind of reports like *HTML, JSON, XML* etc. Although those reports are enough to give you the overall execution results with detailed time logs and other things. But there are many third party plugins also available in market, which helps you to produce awesome reports which has much more capability to produce test logs and better visual graphics. This chapter focus on *Extent Report* plugin to create beautiful and meaningful Cucumber Test's report.

Cucumber gives us the capability to generate reports as well in the form of ***HTML, XML, JSON & TXT***. Cucumber frameworks generate very good and detailed reports, which can be shared with all stakeholders.

RERUN PLUGIN

FOR LOCATING FAILED TEST SCENARIOS. IT GENERATES FILE WITJ RERAN.TXT UNDER TARGET FOLDER

FAILED RUNNER CLASS:

ALLOWS TO RUN FAILED TEST CASES

**Stepdefinitions**

[*Before and After Hooks*](https://toolsqa.com/selenium-cucumber-framework/how-to-use-hooks-in-selenium-cucumber-framework/)

Cucumber supports *hooks*, which are blocks of code that run *before* or *after* each scenario. You can define them anywhere in your project or step definition layers, using the methods *@Before* and *@After*. *Cucumber Hooks* allows us to better manage the code workflow and helps us to reduce the code redundancy. We can say that it is an unseen step, which allows us to perform our scenarios or tests.