**BDD Cucumber framework**

**Technical knowledge**

+ Java basic for tester

+ Selenium

+ Gherkin

+ BDD

**Framework objective**

* Use to write automation scripts using selenium and gherkin language
* Support for acceptant testing via BDD methodology
* Easy to use for non-technical person at level scripting
* Many forum communities

**How to build this framework**

Design patent:

+ Page Object model: reference this [link](https://www.testingexcellence.com/page-object-framework-java-webdriver/)

Page Object model is the powerful design patent using in structuring project. It is the methodology that shows you how to structure your source in a simple way to be readable and understandable for users as well.

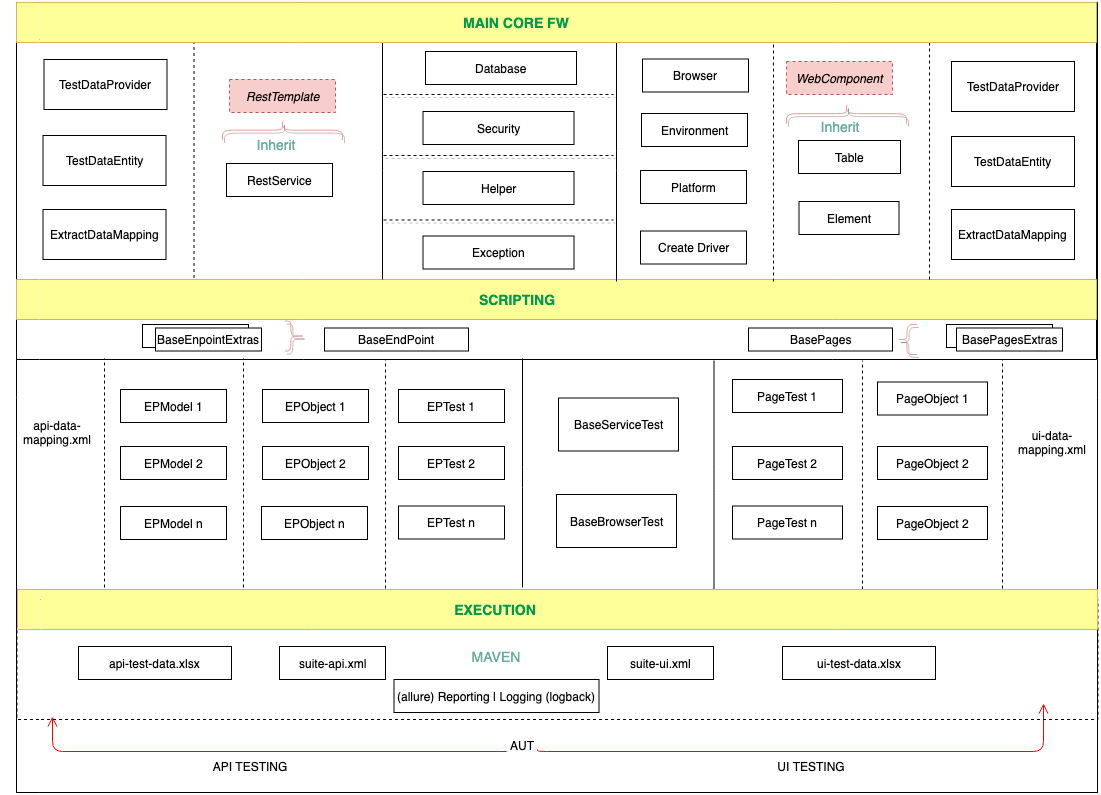
For developer, if you follow this patent, it’s easy to clarify your code base on your business objective, your demand and handy to maintain your source code

+ Java singleton: [reference](https://www.geeksforgeeks.org/singleton-class-java/)

Java singleton model is the technique in java language. By method getInstance(), it allows to declare object only once time and use for all whole project. Developers can create a useful class in java, it means your class do not need follow OOP (a magic of singleton) as well as alter “Static” instant in java language, that will help your code decrease hardcode and save your strorage.

**Framework structure:**

Bellow is the framework’s structure. It’s build to test both API and UI using for both non-technical and technical people. As a person who has non-technical still write automation script because core framework have been defined all the keyword, all of things that you have to do is capture web elements on UI interface, map it with keyword ( Examples: if you want to click on button submit, you just capture submit button on web page (By xpath, index...) “buttonSubmit.click()”. “Click()” function is defined, you just call and use it.).



Main core framework: Contain all things related handling hardly changing things while scripting as configuration, web driver, data, utilities functions, plugins as well as common functions which not related @Test annotation. Only person who have alter know about technical can access and work on layers.

Scripting: This use Page Object model to defind structure, both UI and API have the same structure, just different in handling code.

For UI testing:

+ Base page (Base Object class): Common page and common functions for many page.

+ Page test: extend from Base page: capture interface and write functions

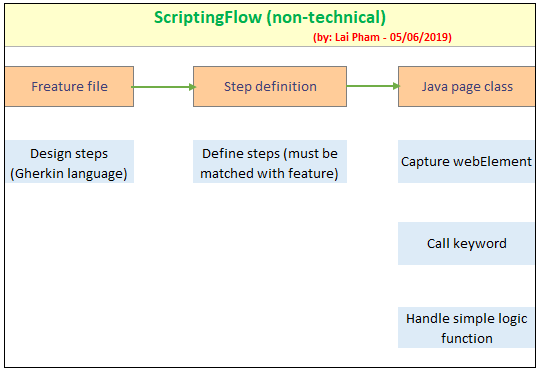
+ Base test: Set up and tear down for each test case. This use to handle which functions run before or after test case

+ Test case: Extend from Base test: Define steps that should be matched with declared functions in Page test

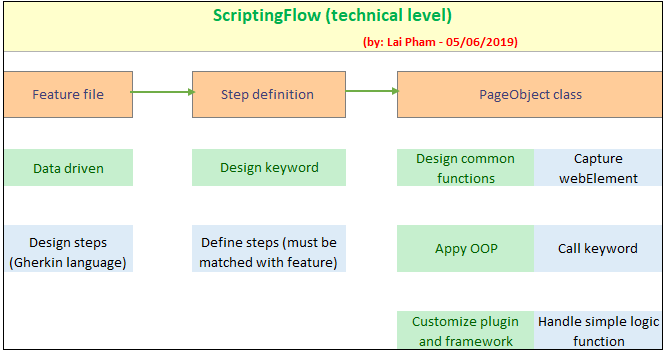
Execution: Configure to run testcase or test suite, it also configs to log the report after finish exciting the scripts

For people who does not know about technical, they still work on Scripting and Execution layers in an easy ways, because all keyword, all business actions have been defined already.

**Workflow of framework**

****

They just get the test case, define the specific steps, then capture interface and then map the element with defined keyword (business actions). The final is verifying the result and write report bug if have bug when exciting test scripts



Up to your demand and your technical knowledge you can access to core framework to customize some functions to suitable with your business objective. You also can maintain framework in case some functions have issues, those are not work as expected.