Automation framework - the way of orgonizing the automation project. We orgonize files and folders related to the project in a structured way. When we work on a project we need to create multiple files of different type: page object classes, test cases (test classes), utility files, screenshots, reports, configuration files (xml), so there are a lot of files in the project that we need to orgonize. Instead of keeping all the files in a single place we need to divide them into folders and establish the communication between them. Why? Because it is easier to maintain the project, to add new files, or edit existing files is easier when they are organized.

Advantages of using the concept of framework:

1) Reusability

2) Maintainability

3) Readability. Means when you create something, everyone from your team should be able to understand it.

Mainly we have 2 categories.

1) Built -in. Means framework already available in the market. For example (testNG, Cucumber, J-Unit …). If you choose these type you will get many features (parallel execution, parameterization, grouping …). All these frameworks you may call 3-rd party frameworks. But with use of these frameworks we can not achieve the ready to go framework. Because they provide only the basic/standard options which might not be sufficient for every project.(the project might be uniq).

These frameworks are not subject to change, only for use or not use.

2) customized (user defined). These frameworks are subject to change. They differ from project to project, from company to company. Because of the complexity, or team skills or specific project. Examples are: data driven framework, keyword driven, modulate(модулированная), hybrid (гибридная).

To build the framework, we need to follow certain phases:

1) Analyzing AUT (application under test)

a) Number of web pages

b) What are the different type of elements presented on the pages, how to handle them

c) What to automate / what we can not automate. Only automation is not inaf, manually testing will always needed.

2) Choose test cases for automation

In the project for example 100 test cases, as a rule it is not possible to automate all of them. Let's consider 90% - automatible, 10% - manual. So the automation engineers task is to automate 100%, but these 100% it is 90% of all existing test cases.

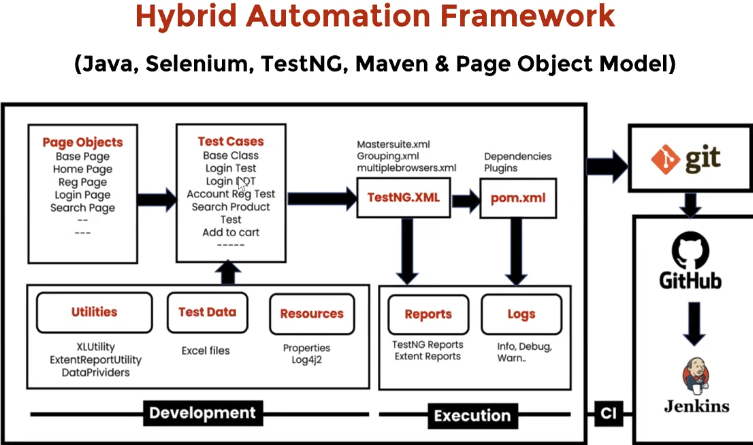
But how to choose test cases for automation?

a) Sanity test cases, means basic functional test cases. How to recognize them? If the test failed you can not proceed further in application. For example login page, or creating account, without account you can not proceed further.

b) Data driven test cases / re-test

c) Regression

3)Design & development of framework



4)Execution. Local and Remote.

5)Maintenance.

Everything should be in a remote repo in order for the team to have access to the updated project and to be able to insert updates and fixes.

There are different types of domain applications.

If you take any specific domain, all kinds of applications that belong to these domain will work in a similar way, UI might be different but main objective will be the same.

Different domains are:

**Banking -** sensitive application (means sensitive data that maintain in these app. + money+transactions)

**Finance -** sensitive application

**Travel**

**Portals**

**Health**

**E-Commerce**  (application allows to buy goods online). There are basically 2 types of operations that are common in that domain, does not matter of the app, it will steal execute 2 types of operations:

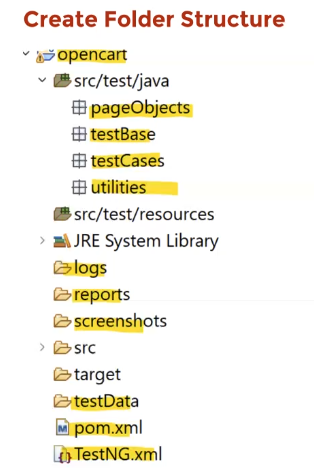
- Front end operations (Example: register the account, log in to app, search for the product…any activities on the page)

- Back end operations (Example: maintain the product description, add products, delete products, maintain customer info, order info, …

**Folder structure**

Всё об приложении походу лежит в src.test.java. в виде **пакетов** (не папок). Потому что это всё java файлы.

Папки сторонних инструментов в корне проекта, чтобы работать с ними можно было отовсюду, и они не .java файлы поэтому для них просто папки.



**Logi**

[**Что такое логи.**](https://docs.google.com/document/u/0/d/1CqXDJj6lBdKpzxV2erGaRhOf4KwTBSuvOjf3en68dxc/edit)

**PROPERTIES FILE in the project**

Добовляю проперти файл src/test/resources/config.properties в нём будут хранится переменные которые динамически не изменяются,

типа url... Динамически изменяемые значения которые используются в Data Driven Tests - это другой концепт (подход) их надо

хранить в Excel наверно. Целью же config.properties файла является хранение захардкодженных значений которые статичны, в

переменных и использование этих переменных в коде когда надо во многих test case - ах. Чтобы загрузить пропертис переменные,

нам нужно создать экземпляр класса Properties там, где нужна переменная из сonfig.properties, в нашем случае это BaseClass

именно он переходит по url и ... смотри --> //loading properties file

в BaseClass-e. Чтобы использовать обьект Properties, вызови метод getProperties.

properties.gerProperties("название\_переменной")