**API Automation Framework**

**What is Framework**

When we try to automate any test case so we have so many types of files , so we have to maintain all types of files,

like we have to maintain different URL and End points, some time we have POJO classes, data driven test cases,

Reports, so we need to maintain all those things properly (Maintain all project related files).

**Main objective of designing the framework**

1 Reusability

2 Maintainability

3 Readability

Reusability : Some time we create/ use any component or class that will be reusable , multiple time we can use it

Maintainability : If we design our framwwork today tommorow someone use that framework

and he want to modify existing test cases then he should abole to do it.

Readability : Once we design the framework the any one can read or understand code/ framework.

**Hybrid Driven Framework**

Hybrid Driven Framework is combination of Data driven and Keyword driven framework

**Phases Comes under developing the Framework**

1 Understanding Requirement

Functional specification(Static Documet)

Swagger (here we know what kind of request and response in expected)

2 Choose Automation Tool

Rest Assured Library

3 Design Framework

What kind of folder structure should be there

What kind of file we have to create , how we manage

4 Development

What ever file we decide, once we design we have to implement those files.

5 Execution + Continues Integration(CI)

**Pre-Requisites**

Step 1 : Create a Maven Project

1 Go to File

2 click on new

3 Select project

4 Inside Maven select Maven Project

5 Select first check box & click on next

6 specify group\_id & artifact\_id and click on finish.

src/main/java : developer keep developing code inside the src/main/java.

src/test/java : If developer wants to test that code, Those test cases they will keep inside the src/test/java.

src/main/resources : Suppose while developing the main code, if we need any resources, like property file, src file,

or any external file they will keep inside the src/main/resources

src/test/resources : while developing the test cases if you need any resources that will keep inside the src/test/resources.

For the testing prespective we need to folder only

1 src/test/java

2 src/test/resources

Step 2 : Update pom.xml with required dependancies

Step 3 : Create Project Structure

Step 4 : Create Routes.java which contains URL's

We have to create routes class inside the endpoints package, this class contains only the required URL.

Step 5 : Create UserEndPoints.java file which contains CRUD methods implementation.

we have to create this class inside the endpoints package.

Step 6 : Create PojoClass.java under api.payload package

here we have to add the require filed or variable, then select all, go to source and select generate getter and setter

Step 7 : Write the Test Cases.

First we have to create one class UserTest.java under api.test package.

Here by using faker labrary we have to create the data & pass it to the POJO class.

And that data will be pass along with the POST request

In this class we have to call all the methods which is created inside the UserEndPoints.java class

Also we have to follow

1 Create a Data Using Faker Libary & pass it to POJO class ie. User.java

2 Then we have to write the test methods for post,put,delete,get & set the priority to all methods

1st we need to set up data using pojo

by using faker we create the data (we are generating our own id)

for post user we are calling create user method by passing userpayload

Step 8 : Create data driven test

Suppose i want to perform data driven testing, whether user able to create multiple types of data we can check.

**Pre-Requisite:**

1 Need to have user data in excel sheet

2 need to use apache poi, we need ExcelUtility file

3 create one folder "testData" inside that folder add your excelsheet

4 DataProvider

we have to create multiple user using our excelsheet data.

Also we need to add two dependancies

apache poi

poi-ooxml

Create on class "XLUtility" inside the api.utilities package

If you have one data to test one data provider is enough but if you have multiple dtadriven test we have to create

dataproviders and these dataprovider we have to keep inside the api.utilities

**Why we use DataProvider**

1 DataProvider will get the data from the excelsheet.

2 DataProvider is one of the annotation method which ever method we provide the dataprovider that method is responsible to

provide data to another method

3 DataProvider get the data from excelsheet and store that data in 2D array.

4 DataProvider provide/pass the data to test case(method)

5 The test case will repeate the number of times based on the data provided by the DataProvider.

6 so we used here XLUtility along with the DataProvider.

7 DataProvider is reusable component so under api.utilities package we have to create "DatatProviders.java" class

Example:

suppose any perticular method is define with @DataProvider anotation

that method generating the data and passing the data to another test method in a project

5 create on e dataDriven class "DDTests.java" under the api.test package

Here in DataDrivenTest class we can create multiple test/user by sending POST request multiple time.

Once it is created we can deal with all the users by passing the user name.

Inside the excelsheet we have no. of users, so 1st we have to pass the entire excelsheet data to the database using POST,

suppose we have 5 user so, we have to create 5 test.

Suppose you want to delete any user so from excelsheet we have to read only username & pass that username as input to delete request.

Then delete the user which is available in the database.

**We have to create two data driven test case**

1 we are create multiple user's by sending POST request

2 we are pass only user name to data provider & delete the user

1 create multiple post method, also we use pojo class

Step 8 : Generate Extent Report

For generating extent report we required

1 Extent report utility class

2 testng.xml file

1 - first we have to create utility file

2 - we have to add extent report dependency in pom.xml

Here we use Listener in TestNG listener is used.

once you execute the test method based on status message you will perform some post action, pass,fail,skip.

Generating test report is one of the post action, once our test method executed then we will create reports.

**Also we have to use 4 methods**

1 onStart

2 onTestSuccess

3 onTestFailure

4 onFinish (once you created everything then we call flush method to make sure everything is ready in the report

if you do not call this method the report will not generate)

---whenever you want to generate extent report using testng.xml file

---create one xml file follow the below steps

1 Right click on api.test package

2 Go to TestNG

3 Select Convert to TestNG

4 the file name if you want

5 click on finish (TestNg cml file will be generated)

Step 9 : Add the Log's

* Logger is like a recording of what happens in a computer program. It helps programmers find and fix issues, see how the program works, and keep track of important events.
* Here we use log4j2 library which is 3rd party library
* By using this log4j2 we will able to generate the log's

1 First we need to have "log4j2.xml" file. this is one of the configuration file

2 Add log4j2 dependency

We have to create this files inside the "src/test/resources"

3 Write code to generate the log's

we have to write this code inside the UserTests.java class which is under api.test package

4 we have to create Logger object inside the UserTests class which is comes under api.test package

5 Then we have to initiate the logger variable inside the setup method In same UserTests class before sending the request we have to write the logger.info("message");

* Then run testng.xml file
* Once we run log's will be generated. inside the log folder, just we have to refresh our log folder.
* We can see all the messages will be generated.
* In XML file if we put "Info" then only information will be displayed but we we write "DEBUG" then
* Debug log/ we can see so many logs.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Another Way To Run the Framework \*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

instead of writing EndPoints, Route class, we have to create on Properties file, inside that Properties we have to place all the URL's

1 Create Properties file

Go to Project

Create properties file under src/test/resources

Right click on src/test/resources,select new, file, provide the name routes.properties

**Properties file :**

* Properties file is like normal text file, we are not follow any proper syntax here, we dont wont any java syntaxes like duoble quotation, single quotation, or semicolon these things wont work, so we can just write plain data in the Properties file.

* In Properties file "#" hash is a comment.

2 Create one File for UserEndPoints, copy UserEndPoints file from api.endpoints and paste it in same package & rename it by UserEndPoints2

* Design this end point to get the data from properties file.
* We need to add one additonal method, which will load your properties file.
* In java we have one class "Resource Bundle" is a special class which we have so by using that class we will able to logs properties file and we will ablle to read the data from the properties file

3 Create one class for UserTests2.java Copy paste UserTests.java in same package change the name for reading the data

4 go to testng.xml

* add your class name there i.e UserTest2
* Run Your testng Suite

5 Execution

Run the test cases using testng.xml you can directly run but if you want to run your test

cases remotely or through jenkins command prompt then we should able to run through ppm.xml

* We need to add dependency/plugins in pom.xml
* We need to add two different plugins.
* We need to add these plugin before the dependencies.

1 Maven compiler plugin

2 Maven surefire plugin

1 Go to Chrome

2 Search maven surefire plugin is rewrite to run our test case through pom.xml

3 click on apache maven i.e maven.apache.org

4 Go to Usage

5 copy paste dependency/plugin in pom.xml

6 go to search maven compiler plugin

7 click on apache maven i.e maven.apache.org

8 Go to Usage

9 copy paste dependency/plugin in pom.xml

10 also we need to add <build> <pluginManagement> tag before Plugin

11 In same plugin there is on comment

<!-- put your configurations here -->

we need to add testng xml name

<suite>testng.xml</suite>

12 inside the configuration we need to add one more tag

<configuration>

<!-- put your configurations here -->

<suiteXmlFiles>

<suiteXmlFile>testng.xml</suiteXmlFile>

</suiteXmlFiles>

</configuration>

we need to add above tag under the surefire plugin.

12 Save all the Plugin/dependency

Then run pom.xml

pom.xl is related to maven we have to run pom.xml file throght maven

**How to run pom.xml**

1 right click

2 select run as

3 select maven test

* Our all the test cases run successfully & it shows BUILD SUCCESS, it meand all our tests are successfully executed throught pom.xml.
* So the configuration whcih we have to do in the pom.xml so that we are able to run our test through pom.xml
* Also this is same process we have to follow when we use maven, for selenium, restassured

-- we need to add maven plugin

-- specify the configuration

6 Run pom.xml Remotely

* we are running our test within our Eclipse locally But Suppose tomorrow i move this project to somewhere else i dont have an eclipse and without Eclipse also we shouls able to run our project.
* We have to run our code through Jenkins , we are able to run our test in the remote environment that means through jenkins we will able to trigger

**First of all we have to Check**

1 Our scripts are working without having Eclipse.

2 Without using Eclipse can we run our scripts through command prompt

3 we have to run this pom.xml through command prompt

* In Eclipse we already have Maven Plug-in by default,we will get a maven plugin so that's the reason we are able to create maven project, able to update pom.xml, we are able to run pom.xml

* Whenever we want to run our project outside the Eclipse.
* Then we need to install maven at the OS level that means if you are working on MAc or Window you have to download the maven software and we have to install it So, Maven is available in the two different Flavors

1 Eclipse plugin

2 Software in the OS system level we have to install on maven software so we will able to run pom.xml in our command prompt

**Steps:**

1 Close your Eclipse

2 Install the Maven on Window

3 First download the software

4 Go to Chrome search Download Maven

5 Select Maven apache org, i.e download apache maven

6 As per your window select file for win select Binary zip archive "we need to do small configuration here"

7 Once download extract that zip file

8 After extracting copy this folder and paste it in c drive

9 Open the folder and set the path

10 click on folder open the bin folder & copy the above path

11 go to my pc , go to properties

12 go to advanced system settings

13 click on environment variable

14 select path variable and click on edit

15 click on new , paste the url click on ok, ok

**Cross check maven configuration**

1 Open cmd

2 Write mvn -version

3 after running above command maven is added in your machine or not

4 Then we run the project through cmd

**How to Run project on CMD**

1 type cd "location of your file"

2 type dir 'it shows all project related files'

3 type mvn test (this will fetch all the pom.xml information)

4 Then it shows the result (if java version is not compatible error will be occured)

* Suppose if the error occured then add the tag in pom.xml inside the maven-compiler-plugin

<configuration>

<source>10</source>

<source>10</source>

</configuration>

* As well as we need to add a java 12 or 13 or we need to install version which is above java 11
* Then need to set java environment variable
* type mvn test in cmd

**Need to remember below points:**

1 Execute tests using pom.xml within eclipse.

2 Execute tests using pom.xml in command prompt.

3 Commit code in local repository (Git repository)

4 Push your code to remote repo (GitHub repo)

5 Run project in Jenkins (From git github) (jenkins will get code from the latest code from the GitHub every time

* is everyday task whenever you do some changes whatever you add new test cases or update / delete anything we will always make sure those changes should be part of local repository & remote repo).
* If we are executing successfully using maven command then only we will able to run our project remotely through Jenkins from Git.
* we will able to run it because in Jenkins also we will specify the maven command.

-- Jenkins also run your project by taking the help og the same command.

**What is CI (Continuous Integration)**

* There are multiple dev and tester so developer , there are multiple developer are working on multiple things.
* So, at the end of the day what they do is they checkin the code in the common tepository this repo is global repository.
* But before sending the code to global repository they are also having their own local repositories this is call as a git, git is local repository.
* So they will commit their code first inside the their local repositories and then they will push their entire code from their local repository to remote repository.
* Once the entire code is availale in the global repository this is Github Repository.
* Git is local repository and GitHub is global repository.Now devops team will do here is devops team will create "Build" by taking all this code build or package or what ever related to code, this we will validate by using testing code, so whatever developer is doing parallely the Tester also do the same things.
* So What ever automation code we have developed at the end of the day all the tester they will keep their code in their local repository which is git repository and from therre they will push the code in the GitHub repository & in the devops environment they will use the Maven to create a build from the developers and from the GitHub they will get the automation code using jenkins they will try to exit or automation code against the build.
* So here it self basic sanity and regression is completed before getting the build. all this process called Continuous Integration process one side the developer will do their own tasks and other side thetester also will do their own tasks so developer's, tester's & devops these teams will work together collaboratively to achieve the one goal

**Why we need to run project so many times**

* Above is actual CI process that is the reason we should able to execute our project first in the command prompt before directly jumping into the Jenkins, in the local environment we are testing whether our project is successfully working fine or not.

**So we need two verification**

1 We should able to run our project within our eclipse

2 We should able to execute project within command prompt

* Once they are successful then we will able to commit our code into Git
* Then push it to the GitHub
* Then using Jenkins we will able to run our project All this is a complete process.
* Then commit the code in the local repository

**CI (Continuous Integration) Process:**

**Pre-requisites :**

1 Install git software

2 Create an account with github and create empty repository.

Repo URL : <https://github.com/genzsoln/Sujata_API_Automation_Framework.git>

**Git Installation:**

**Steps 1 : Install Git**

1 Go to The Chrome

2 Search Git bash

3 Click on first link i.e Download <https://git-scm.com/downloads>

4 click os Download as per your OS win, mac, linux

5 Go to Download in you system

6 Click on Git install the software just clicking next next.

* After you install Git Bash, you can right-click inside a folder. You'll see an two options 'Open Git Bash' 'Open GIT GUI'

When we open 'GIT Bash' then you will able to see the git command prompt. You can then type and run all different commands in that window.

**Step 2 : We need GitHub Account.**

* Go to github.com
* create an account with github.

**Following are the commands we need to use**

1 **git init -** create local repository in your project location

Go to your project location

Right click on any place

select 'git bash'

Git command prompt will be open

After git init one folder will be created in your project location i.e 'git'

**2 git config --global user.name "your name"**

**git config --global user.email "abc@gmail.com"**

* Once we create our local repository & we have to provide our information to the local repository.
* because 1st time when we're working with a git command first time we have to tell our username and email address to the git repository
* Because whatever changes we do or whatever updations modifications you do

everything well be remembered by suing our name anf mail address So we have to use these command

* only we have to use these 2 command when we use git first time.

**3 git status**

* we have to check the status like is there any files are ready to add or not every time we can check the status.
* So whether files are there in the working directory or not, already commited already added or not already committes,pushed or not
* We can verify by using status commands.
* after this command it will shows how many files or what we need to add this files inside staging area.

**4 git add -A** If you want to add all the command into staging area.

**git add filename.ext** If you want to add specific file the directly write the name of that files.

**git add \*.ext** If you want to add multiple files with the same extension which are

having same extension then we can write this simple regular expression \* start

It means all the files which ever is having have extension will be added to the staging

**git add foldername** If you want to add any folder.

**5 git commit -m "message"**

* what ever changes you done you have to add inside the message.
* after this command our entire code commited to the local repository with our git repository.
* after executing above command what ever shows in white color it will commites files.
* Once we have files in the git repository then we have to push our code into the remote repository.
* For that we need to create empty repository.
* then we have to push code.

* Go to your Github account, create one empty repository.

1 Go to home page

2 there is "new" button click on that.

Or there is one '+' button click on that there is one option 'new repository' click on that.

3 Give the repository name

4 Click on create repository.

5 This will create an empty remote repository also there our repository URL will be displayed.

**Push the code to remote repository**

we have to execute one command here

**6 git remote add origin MainBrnach**

This cmd just establish the connection we have to execute this command only one time, it will connect local repo to remote repo

**7 git push -u origin MainBranch**

Go to your remote repo refresh your page, we will able to see all the files.

* Suppose you already done something in one round & in second round we added some new test cases in my workspace,
* So new changes is not part of our local repo and staging & also which not part of our remote repo , so we have to run all the command from the beginning.

**so we need to run below command**

* git add
* git commit
* git push

1 git add -A

2 git commit -m "msg"

3 git push -u origin MainBranch

* Once the files are available in the remote repo our Jenkins should able to execute these files in their environment
* we need to Install Jenkins.
* In GitHub repo we dont do any direct changes, whatever changes we want to do that should be done only through our project.
* If we delete something that also automatically committed, again we need to run the commands add, commit, push.

**Download Jenkins**

Refer this link how to download and install jenkins : <https://www.youtube.com/watch?v=0RVbTDLXk50>

Step 1 : Go to website jenkins.io

Step 2 : Click on Download

<https://www.jenkins.io/download/thank-you-downloading-windows-installer-stable/>

Step 3 : Go to download section click on windows (as per your OS) (installation started)

Step 4 : Once download completed install the jenkins.

* Logon Type select first option
* Click on Test Port
* after clicking next it will showa error if you installed above java 17 version
* because jenkins only supports java 11 to 17
* download java 11
* click on Java SE 11 Archive Downloads

<https://www.oracle.com/in/java/technologies/javase/jdk11-archive-downloads.html>

* for installing java 11 we need to have oracle account we need to create oracle account then download will be started.
* after download we need to paste that dolder c drive--program file--java
* set the environment variable for java 11
* do the same installation process
* set the java 11 path click on next and install Jenkins
* go to chrome- search localhost 8080
* There is one URL click on that you will get one link copy that link C:\ProgramData\Jenkins\.jenkins\secrets\initialAdminPassword
* Go to c drive and paste that path
* Then it will shows an option where you want to open that password select notepad

76ab9b4d40e54f9b98b02956c911a297

cc66a953b3f440d8b35d0a8a9cc70979

* Paste that password on the above link page click on continue.
* Select install suggested plugins
* wait till everything done, after installing all plugins click on continue.
* add your all details to create account name, email, password etc
* click on save and continue.
* It will provide your Jenkins URL : <http://localhost:8080/>
* Click on Save & Finish.
* Click on start using Jenkins.

**Jenkins :**

* Jenkins automates tests when code changes, speeds up testing using multiple machines, and provides reports for quick issue detection.
* Jenkins automates tests when code changes, makes testing faster, and helps deploy code.
* Jenkins will run on localhost 8080
* Login to your jenkins account <http://localhost:8080/>

When you want to run your program in Jenkins follow the below steps :

1 On Jenkins there is one option Manage Jenkins

2 Go to global tool configuration or tools (we have to set maven path and jenkin path correctly)

* Add JDK 11 path
* Add Git Path along with executable file C:\Program Files\Git\bin\git.exe
* click on maven installation

**We have to install maven for windows**

* + Go to Chrome
  + Search Maven
  + Click on offical apache maven website <https://maven.apache.org/>
  + click on download
  + click on Binary zip archive
  + Extract that file, go to c drive, program file & paste that folder.
  + Now set the maven path
  + First set bin path inside environment variable

3 On dashboard there is on option New Item click on that

* Write name of your project.
* Select the Maven

4 One page will be open

* Inside the General section we have to select some options
* Source code management is "Git"
* Specify remote repo URL
* In Build section in goals and option write "clean test"
* apply & save
* Now our project will created in Jenkins.

**How to Run project in Jenkins**

* In dashboard we can see our project.
* write click on 3 dot near our project
* select Build now

OR

* Directly click on project, there is option build now.
* First time it will pull project in jenkins from github repo it will start execution project.