MAVEN

It is a project management tool or software build tool.
It is used to build the software application, manage the dependencies, run the tests and create reports.
Instead of downloading jars each time, we can go for maven project since we can get all the predefined libraries using dependencies.
Dependencies is a three-liner code which needs to be mentioned in pom.xml.
When we use jars, if the version got updated means we need to download the updated jar, but using maven, we can just change the version of the dependencies in pom.xml file.
When we change the version in pom.xml, it will clean the old libraries, install the new libraries and then execute. This is also known as Maven Goal (i.e.,) Clean, Install and Test.
In Maven, the project folder structure contains

- src/main/java
- src/main/resources
- src/test/java
- src/test/resources
- JRE system libraries
- Maven Dependencies
- src
- target
- pom.xml

There are some repositories of Maven

- Local Repository - This is a local directory in local where maven stores downloaded dependencies. It's typically located in the .m2 directory in user's home folder.
- Remote Repository - These are repositories hosted on remote servers accessible over the internet. When Maven needs a dependency that's not available in the local repository, it searches remote repositories to download the required artifacts.
- Central Repository - This is the default remote repository for maven. It hosts a vast collection of Java libraries and artifacts that can be freely used by Maven projects.

JUNIT

JUnit is a unit testing framework for the Java programming language. It is used to verify the business logic and easy to identify the pass and failed test cases. Here we can verify and validate the business logics using assert in Junit.

There are certain annotations like

- @BeforeClass used to launch the browser
- @Before used to note start time of the execution
- @Test where business logic should be provided
- @After used to note end time of the execution
- @AfterClass used to quit the browser
- @RunWith and @SuiteClasses used for suite level execution
- @Ignore used to skip a particular test case in execution.

POM

POM is page object model which is a design pattern used to maintain all the locators in page wise using POJO classes.
For example, in my project we have more than 20 pages, for each page we have POJO class which means plain old java object where we will maintain our locators in page wise.
Inside the POJO class we will have a constructor which contains PageFactory.initElements().
PageFactory is a class used to store all the web elements and initElement is a method used to initialize the web elements.
Then there are certain annotations like

- @FindBy used to call one locator
- @FindBys used to call more than one locator which supports AND condition (i.e.) all locators should be satisfied.
- @FindAll also used to call more than one locator which supports OR condition (i.e.) any one locator should be satisfied.
- @CacheLookUp used for memory management.
  In POJO classes, we declare all locators as private, then generate getters and setters.
  Then create objects in main class and indirectly call the locators from POJO classes which implements the encapsulation concepts.
  POM is used to maintain all locators in one place, if any changes are made in DOM structure means we can update the locators in POJO class, so there won't be much rework in scripting.
  And we can handle StaleElementRefException using Page object model.

GIT

What is GIT?

GIT is a source code management tool or version control tool used to maintain our codes in a repository and it is also used to push codes from local to repository and pull codes from repository to local using GIT commands. In our project we are using GITBash tool.

How will you push your codes from local to repository?

In my project, I will create a branch to maintain my codes and then clone the project using **git clone URL**. Then I will import the project in to my local and then checkout the project to my branch using **git checkout -b Branch**, then I will start my scripting in my local. Once I have done with my scripting, I will push the code to my branch using

- **git status** which is used to get the status of the files which is modified
- **git add .** which is used to add the files
- **git commit -m "message"** which is used to commit the files using some message
- And then push codes using **git push -u origin branch**

How will you merge your codes with master?

First, I will check whether any one of my colleagues have merged their codes with master, if no one have done, I will directly give pull request and merge my codes with master.
But if any one of my colleagues have merged their code with master, then I need to take the latest codes from master using **git pull origin master** to avoid the conflicts.
Then I will rearrange the codes in local and merge my codes with master by providing pull request.

What are conflicts?

If I am not taking the latest codes from master and try to give pull request, conflicts will occur in repository, which will create issues. To avoid conflicts, I need to take latest codes from master using **git pull origin master**, then remove header and tail and merge my codes with master by providing pull request.

Other GIT Commands:

git config --global user.email emailaddress used to set the email address
git config --global user.name username used to set the author's name
git init used to start a new repository