Maven is a tool which is used for building and managing Java Based Projects. Basically to put it in simple words is a way to manage dependency for Java Based Project. Maven can be used when building project with POM (Page Object Model) when working on big projects.

Below are the objectives which can be achieved with maven –

- Easier and Uniform build process.
- Providing quality project information
- Easy Docmentation
- Best practices development
- Manage the dependencies

Lets understand them one by one

- **Easier and Unfirom Build Process** –

  Maven provides pom.xml configuration files where all information such as construction directory, source directory, test source directory set of plugins etc, which are shared by all projects.

- **Providing quality project information** –

  Maven provides information present in pom. It provides Unit test reports, list of mailing lists, dependency lists, cross-referenced sources,etc.

- **Easy Documentation** –

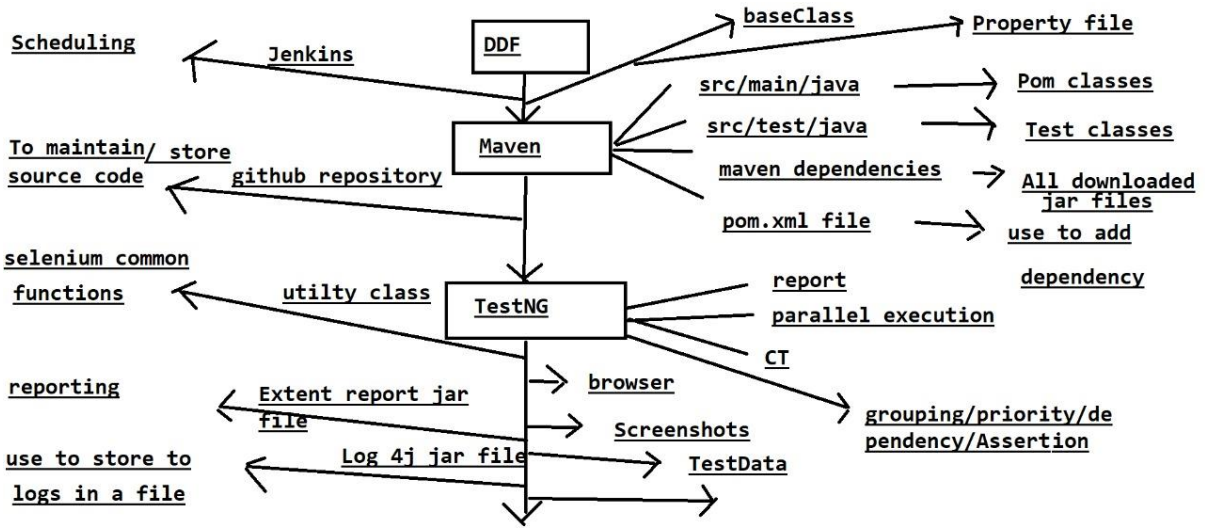
  It will manage your Selenium test project's build compilation, documentation and other related project tasks itself. It helps to create proper project structure,add and manage jar files in project's build path.

- **Best practices development** –

  It easily guides the project by gathering current principles for best practices development. Test source codes are kept in separate but parallel source tree. Naming conventions are used to locate and execute test cases. By a proper project structure, we can easily navigate to other projects that use Maven.

- **Manage the dependencies** –

  According to the webdriver version, Maven downloads all the required files in the local repository called m2. Later if there is any change in version of WebDriver, then only pom.xml needs to be modified. Mavan will automatically download the new version jars and store in local repository.




**Maven build Life cycle:**

Build Lifecycle Basics

Maven is based around the central concept of a build lifecycle. What this means is that the process for building and distributing a particular artifact (project) is clearly defined.

For the person building a project, this means that it is only necessary to learn a small set of commands to build any Maven project, and the POM will ensure they get the results they desired.

There are three built-in build lifecycles: default, clean and site. The default lifecycle handles your project deployment, the clean lifecycle handles project cleaning, while the site lifecycle handles the creation of your project's site documentation.

A Build Lifecycle is Made Up of Phases

Each of these build lifecycles is defined by a different list of build phases, wherein a build phase represents a stage in the lifecycle.

For example, the default lifecycle comprises of the following phases (for a complete list of the lifecycle phases, refer to the Lifecycle Reference):

- validate - validate the project is correct and all necessary information is available
- compile - compile the source code of the project
- test - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
- package - take the compiled code and package it in its distributable format, such as a JAR.
- verify - run any checks on results of integration tests to ensure quality criteria are met
- install - install the package into the local repository, for use as a dependency in other projects locally
- deploy - done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

These lifecycle phases (plus the other lifecycle phases not shown here) are executed sequentially to complete the default lifecycle. Given the lifecycle phases above, this means that when the default lifecycle is used, Maven will first validate the project, then will try to compile the sources,

run those against the tests, package the binaries (e.g. jar), run integration tests against that package, verify the integration tests, install the verified package to the local repository, then deploy the installed package to a remote repository.