Selenium WebDriver is great for browser automation. But, when using it for testing and building a test framework, it feels underpowered. Integrating Maven with Selenium provides following benefitsApache Maven provides support for managing the full lifecycle of a test project.

* Maven is used to define project structure, dependencies, build, and test management.
* Using pom.xml(Maven) you can configure dependencies needed for building testing and running code.
* Maven automatically downloads the necessary files from the repository while building the project.

**Maven** is a build automation tool **used** primarily for Java projects.  **Maven** addresses two aspects of building software: first, it describes how software is built, and second, it describes its dependencies.

Apache Maven is a **software project management and comprehension tool. (Build Management)**

Based on the concept of a project object model (POM), **Maven can manage a project’s build, reporting and documentation** from a central piece of information that is called [Central repository](http://learn-automation.com/github-integration-eclipse-with-selenium/). Maven has its own repository where it keeps all plugin, jars etc. in commonplace in .m2 repository.

Supporters of maven believe that

1. Maven lets you get your package dependencies easily
2. Maven forces you to have a standard directory structure

**Relation with Selenium-**

We can create Maven project for writing script and create dependency-using POM.xml once dependency is set Maven will download all the dependent jar files automatically and in future if any update comes from Selenium or TestNG side it will simply update all the required changes.

**How integrate Selenium with Maven build tool**

Apache Maven is a software project management and comprehension tool. It is formally known as Build tool.  This article will help you to setup project and integrate selenium with maven.Based on the concept of a project object model (POM), Maven can manage a project’s build, reporting and documentation from a central piece of information that is called

Based on the concept of a project object model (POM), Maven can manage a project’s build, reporting and documentation from a central piece of information that is called [Central repository](http://learn-automation.com/github-integration-eclipse-with-selenium/). Maven has its own repository where it keeps all plugin, jars etc. in commonplace in .m2 repository.

[Eclipse](http://learn-automation.com/configure-eclipse-for-selenium/)should be Juno(4.2) or Kepler(4.3) and If you are using Eclipse mars(4.5) or neon (4.6) then Maven comes by default so you can skip initial steps

Selenium Maven artifacts directly in the central Maven repository. We can add it by directly copying/writing dependency OR by selecting it from dialog box (Insert Dependency) from POM.XML

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>3.4.0</version>

</dependency>

artifactId is the name of the jar without version.  Uniquely identify inside group.

groupId will identify your project uniquely across all projects, so we need to enforce a naming schema. It has to follow the package name rules,

Central Repo: (In effective POM)

<repositories>

<repository>

<snapshots>

<enabled>false</enabled>

</snapshots>

<id>central</id>

<name>Central Repository</name>

<url>https://repo.maven.apache.org/maven2</url>

</repository>

</repositories>

<https://repo.maven.apache.org/maven2/org/> (Actual jars available)

<https://mvnrepository.com/> (Jar Dependency xml tags are available)

Once we save the project /XML, the jars (dependencies) added by XML tags are added /saved in C->user->.m2 directory

**Maven Built in build life cycles (Goals)–**

Clean , Test, Install, Site

Clean – To handle cleaning of project

Default – compilimg,testing,building,deployment

Site – Documentation of project

* Each LC has different internal phases/stages
* Phases are executed in sequence order

With below command maven generates java project-

mvn archetype:generate -DgroupId=com.vogella.build.maven.java \

-DartifactId=com.vogella.build.maven.java \

-DarchetypeArtifactId=maven-archetype-quickstart \

-DinteractiveMode=false

Execute jar created using Maven:

cd ////////////got to folder

mvn clean

mvn compile

mvn package

java -cp target/artifactId-version-SNAPSHOT.jar package.Java-Main-File-Name

we need to add

1. maven-compiler-plugin
2. maven-surefire-plugin
3. testng.xml

to pom.xml**.**

The maven-surefire-plugin is used to configure and execute tests. Here plugin is used to configure the testing.xml for TestNG test and generate test reports.

The maven-compiler-plugin is used to help in compiling the code and using the particular JDK version for compilation. Add all dependencies in the following code snippet, to pom.xml in the <plugin> node: