Maven Crash Course

Learn Maven Build Validation Tool from Scratch

Trainer – Haradhan Pal

Agenda

- Introduction of Maven, Installation and running sample program
- How to Execute Selenium Script through Maven and TestNG
- Maven Commands and running Maven Project through Command Lines
- Repository and Repository Manager in Maven
- How to implement Headless Browser testing with Maven
- Maven Profiles to Control Build Execution
- Importance of Snapshots in Maven

Introduction of Maven, Installation and running sample program

- Apache Maven is a software project management and comprehension tool. 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.
- Maven is used to define project structure, dependencies, build, and test management.
- Using pom.xml user 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 dependency management tool, i.e., to execute the frameworks; we need the jar files. Maven helps the user to keep the up-to-date jar file on the framework.
- During runtime, the maven will check the version of the jarfiles present in the local system and compares it with the pom.xml file. If the latest version is available online, then it downloads them automatically and replaces the local version and then executes the framework.
- If there is no internet connection or if the latest version is not available, then the maven executes the framework with the jar files available in the local system.
- It provides 'build success' message if no compilation issues in the framework or else provide 'build failure' message.
- Execute program using Maven user need to follow the below steps:
  - Install Maven
  - Add dependency
  - Execute the framework
- Follow below steps install maven in eclipse, but if user are using luna or above version, they might have maven by default installed with their eclipse, if not below steps need to be followed:
  - 1. Click on the Help menu and choose Eclipse Market place.
  - 2. Search for Maven with Eclipse; user will get below software result. As I have installed already, so I will not get an install option; for other case, they will have to click the install option.
  - 3. Accept the agreement and restart the eclipse which might take sometimes.

Introduction of Maven, Installation and running sample program

- Steps for creating Maven project:
- 1. In Eclipse IDE, create a new project by selecting File | New | Other from Eclipse menu.
- 2. On the New dialog, select Maven | Maven Project and click Next
- 3. On the New Maven Project dialog select the Create a simple project and click Next
- 4. Enter Group Id: and Artifact Id: and click finish. The groupId is the id of the project's group. Generally, it is unique amongst an organization. The artifactId is the id of the project. It specifies the name of the project.
- 5. Right-click on JRE System Library and select the Properties option from the menu
- 6. On the Properties for JRE System Library dialog box, make sure Workspace default JRE is selected and click OK
- 7. Select pom.xml from Project Explorer. pom.xml file will Open in Editor section
- 8. Add the Selenium, Maven dependencies to pom.xml in the <project> node.

How to Execute Selenium Script through Maven and TestNG

- Create a Maven Project and add Dependency for TestNG in pom xml file (https://mvnrepository.com/):

```
<dependency>
  <groupId>org.testng</groupId>
<artifactId>testng</artifactId>
<version>6.8</version>
<scope>compile</scope>
</dependency>
```

- Select Package and Create a New Classes. Add code in the classes.
- Right-click on the TestNG file, and select Run As TestNG Test
- Right-click on the Project and select TestNG and then select Convert to TestNG options. Once we select the Convert to TestNG options, it will open the Generate testng.xml window, and click on the Finish
- User need to add the below plugins in the pom xml file (https://maven.apache.org/surefire/maven-surefire-plugin/):
- Maven surefire plugin
- right-click on the MavenProject → Run As → Maven Test

How to Execute Selenium Script through Maven and TestNG

```
<build>
   <pluginManagement>
    <plugins>
     <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>3.0.0-M5</version>
     </plugin>
    </plugins>
   </pluginManagement>
 </build>


<configuration>
        <suiteXmlFiles>
          <suiteXmlFile>testng.xml</suiteXmlFile>
        </suiteXmlFiles>
      </configuration>
```

Maven Commands and running Maven Project through Command Lines

- Go to http://maven.apache.org/download.cgi to download installation file for Maven
- Ensure JAVA_HOME environment variable is set and points to user JDK installation
- Unzip apache-maven-3.6.3-bin.zip
- Select My Computer and select Properties by Right-Clicking on My Computer. Click on Advanced system settings. Click on 'Advanced' tab and then click on 'Environment Variables..'.
- Click on New button under 'System Variables..'
- Write 'MAVEN_HOME' in the Variable name box then enter apache-maven-3.6.3 Maven path in the Variable value box and click OK.
- Add the bin directory of the created directory apache-maven-3.6.3 to the Path environment variable
- Add below maven compiler plugin to the POM XML file

```
<plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
        <configuration>
            <source>10</source>
            <target>10</target>
        </configuration>
    </plugin>
```

Maven Commands

- **Maven Command:**

- Open a command prompt window (Press Windows + R at the same time. Then enter cmd in the Run window and click on the Ok button).
- mvn -version in command prompt to check the vesion of the Maven already installed user system
- Go to the directory where user saved the maven project (cd location of the project)
- mvn clean: This command cleans the maven project by deleting the target directory
- mvn complie: It's used to compile the source Java classes of the project.
- mvn test: This command is used to run the test cases of the project using the maven-surefire-plugin.
- mvn package: This command builds the maven project and packages them into a JAR, WAR, etc.
- mvn help: This command prints the maven usage and all the available options for us to use.

Repositories and Maven

- Maven repository is a directory where all the packages, JAR files, plugins or any other artifacts are stored with POM.xml. Repository in maven holds build artifacts and dependencies of various types. User need to add multiple dependencies for any framework in Project POM.xml file. Project Object Model (POM) contains all the dependencies that are being used by their current project.
- There are 3 types of maven repository:
  - Local Repository
  - Central Repository
  - Remote Repository
- Maven local repository is located in the local computer system. It is created by maven when the user runs any maven command. The default location is %USER_HOME%/.m2 directory. When maven build is executed, Maven automatically downloads all the dependency jars into the local repository. For new version maven will download automatically. If version declared in the dependency tag in POM.xml file it simply uses it without downloading. User can change the location of maven local repository by changing the settings.xml file. It is located in MAVEN_HOME/conf/settings.xml.
- Maven Central repositories are located on the web. It has been created by the apache maven itself. it contains a large number of commonly used libraries. It is not necessary to configure the maven central repository URL. Internet access is required to search and download the maven central repository. When maven cannot find a dependency jar file for local repository its starts searching in maven central repository using URL: https://repo1.maven.org/maven2/.

Repositories and Maven

- Maven Remote Repository is stored in the organization's internal network or server. The company maintains a repository outside the developer's machine and are called as Remote Repository.

```
<dependencies>
    <dependency>
      <groupId>com.companyname.common-lib</groupId>
      <artifactId>common-lib</artifactId>
      <version>1.0.0</version>
    </dependency>
  <dependencies>
  <repositories>
    <repository>
      <id>companyname.lib1</id>
      <url>http://download.companyname.org/maven2/lib1</url>
    </repository>
    <repository>
      <id>companyname.lib2</id>
      <url>http://download.companyname.org/maven2/lib2</url>
    </repository>
  </repositories>
```

Repository and Repository Manager in Maven

- Maven Dependency Checking Process:
  - Maven will scans through local repositories for all configured dependencies, in case it is found continues with further execution.
  - Maven scans in central repositories for that particular dependency in case of absence of dependencies in local repository. Available dependencies are downloaded in local repositories for future execution of the project.
  - In case dependencies are not found in Local Repository and Central Repository, Maven starts scanning in Remote Repositories.
  - In case dependencies are not available in any of the three Repositories- Local Repository, Central Repository, Remote Repository, Maven throws an Exception "not able to find the dependencies & stops processing".
  - It downloads all found dependencies into the Local Repository.
- A repository manager is a dedicated server application designed to manage repositories. The usage of a repository manager is considered an essential best practice for any significant usage of Maven.
- The followings are the open-source and commercial repository managers are known to support the repository format used by Maven.
  - Apache Archiva
  - CloudRepo
  - Cloudsmith Package
  - JFrog Artifactory Open Source
  - JFrog Artifactory Pro
  - Sonatype Nexus OSS
  - Sonatype Nexus Pro
  - packagecloud.io

How to implement Headless Browser testing with Maven

- Headless browsers are browsers like any other but without a graphical user interface (GUI). The GUI allows user to visualize an aesthetic representation of the page source code (usually in HTML, CSS, and Javascript). Headless browsers fetch the page the same way standard browsers do, and can render the page (screenshot), and save it as an image file if requested.

- The main advantages of using them are that these browsers are more lightweight and therefore require less computational resources, such as memory and processing. They are ideal to use in environments that don't have a GUI: for example, a Continuous Integration server.

- **Maven Dependency:**

```
<dependency>
  <groupId>org.seleniumhq.selenium</groupId>
  <artifactId>htmlunit-driver</artifactId>
  <version>2.27</version>
</dependency>
```

- **Creating WebDriver Instance for HtmlUnitDriver:**

```
WebDriver driver = new HtmlUnitDriver();
```

Maven Profiles to Control Build Execution

- Maven profiles can be used to create customized build configurations, like targeting a level of test granularity or a specific deployment environment. Within each profile element, user can configure many elements such as dependencies, plugins, resources, finalName.
- Maven Profile is an alternative set of configurations which set or override default values. Profiles allow to customize a build for different environments. Profiles are configured in the pom.xml and each profile is identified with an identifier.
- What are the different types of profile? Where is each defined?
  - Per Project
  - - Defined in the POM itself (pom.xml).
  - Per User
  - - Defined in the Maven-settings (%USER_HOME%/.m2/settings.xml).
  - Global
  - - Defined in the global Maven-settings (${maven.home}/conf/settings.xml).
- Build the project with command mvn compile -X. The option -X outputs Maven debug info to console and from the lengthy debug output, From the output, it is clear that Javac debug is still true and optimize is false which means that the production profile is not activated by Maven, and it is still using default configurations.
- To activate any profile use command mvn compile -Pprofile. The option -P activates the profile. From the Maven debug messages, user can confirm that profile is activated and Javac flags debug and optimize are set as configured in the profile.

Maven Profiles to Control Build Execution

- By defining a profile user can override the values defined by project's Effective POM. Maven allows following elements in an Profile.

```
<project>
  <profiles>
    <profile>
     <id>smoke</id>
      <build>
        <defaultGoal>...</defaultGoal>
        <finalName>...</finalName>
        <resources>...</resources>
        <testResources>...</testResources>
        <plugins>...</plugins>
      </build>
      <reporting>...</reporting>
      <modules>...</modules>
      <dependencies>...</dependencies>
      <dependencyManagement>...</dependencyManagement>
      <distributionManagement>...</distributionManagement>
      <repositories>...</repositories>
      <pluginRepositories>...</pluginRepositories>
      <properties>...</properties>
    </profile>
  </profiles>
</project>
```

- To activate a profile, add -P option: mvn package -Pprofile
- If user always want to execute a profile, we can make one active by default:

```
<profile>
  <id>defaulttestingprofile</id>
  <activation>
    <activeByDefault>true</activeByDefault>
  </activation>
</profile>
```

- Then, user can run mvn package without specifying the profiles, and they can verify that the defaulttestingprofile profile is active.

Maven Profiles to Control Build Execution

- Open Maven settings.xml file available in %USER_HOME%/.m2 directory where %USER_HOME% represents the user home directory. If settings.xml file is not there, then create a new one.
- Add test profile as an active profile using active Profiles node as shown below in example.

```
<settings xmlns = "http://maven.apache.org/POM/4.0.0"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <mirrors>
    <mirror>
      <id>maven.dev.snaponglobal.com</id>
      <name>Internal Artifactory Maven repository</name>
      <url>http://repo1.maven.org/maven2/</url>
      <mirrorOf>*</mirrorOf>
    </mirror>
  </mirrors>
  <activeProfiles>
    <activeProfile>test</activeProfile>
  </activeProfiles>
</settings>
```

Importance of Snapshots in Maven

- During development of a large scale application, the process involves a series of changes to be made in the application till it is confirmed to be ready as a final release. Because, there could be multiple teams working on an application across different set of modules.
- Consider 2 teams A & B working on 2 different modules and lets say team B will be dependent on the services provided by team A. Say team A is involved in working on some major critical fixes and those fixes will be checked in every day. So team B has to be notified in a way that there are still some pending work by team A and the final version has not yet been released by the team A. This is where the maven's SNAPSHOT comes into the picture.
- Snapshot is a special version which indicates the current development copy of the project which is being worked on. For each build, maven always checks out for a SNAPSHOT of the project. Hence, whenever maven finds a newer SNAPSHOT of the project, it downloads and replaces the older .jar file of the project in the local repository.
- Snapshot version always gets some updates/changes being made on the project. Also, Snapshot should exist only during development phase and it will not be having a release version. This implies that the build of the project will be changed at any time and it is still under the development process.
- **Snapshot Vs Version:**
  - In case of SNAPSHOT, Maven will automatically fetch the latest SNAPSHOT (data-service: 1.0-SNAPSHOT) every time Maven Test project is built.
  - In case of Version, maven once downloaded the mentioned version say JavaSamples:1.0, then it will never try to download a newer 1.0 available in repository. To download the updated code, Maven Test Project version is to be upgraded to 1.1.

Importance of Snapshots in Maven

- A snapshot version in Maven is one that has not been released.
- The idea is that before a 1.0 release (or any other release) is done, there exists a 1.0-SNAPSHOT. That version is what might become 1.0. It's basically "1.0 under development". This might be close to a real 1.0 release, or pretty far (right after the 0.9 release, for example).
- The difference between a "real" version and a snapshot version is that snapshots might get updates. That means that downloading 1.0-SNAPSHOT today might give a different file than downloading it yesterday or tomorrow.
- Usually, snapshot dependencies should only exist during development and no released version (i.e. no non-snapshot) should have a dependency on a snapshot version.
- When user build an application, Maven will search for dependencies in the local repository. If a stable version is not found there, it will search the remote repositories (defined in settings.xml or pom.xml) to retrieve this dependency. Then, it will copy it into the local repository, to make it available for the next builds.
- There are two types of builds available in Maven:
  - Snapshot builds: SNAPSHOT is the special version that indicate current deployment copy not like a regular version, maven checks the version for every build in the remote repository so the snapshot builds are nothing but development builds.
  - Release builds: Release means removing the SNAPSHOT at the version for the build, these are the regular build versions.
- Although, in case of SNAPSHOT, Maven automatically fetches the latest SNAPSHOT on daily basis, user can force maven to download latest snapshot build using -U switch to any maven command.

mvn clean package -U

- Maven will start building the project after downloading the latest SNAPSHOT of data-service.