ACAMICA

Globant
we are ready

Level 2

Introduction to Maven

2.1 - Introduction

Maven is a software tool for the management and construction of Java projects.

Maven uses a Project Object Model (POM) to describe the software project to be built, its dependencies on other modules and external components, and the order of construction of the elements.

2.1 - Installing Maven

To install Maven, we need as a prerequisite that our **JAVA_HOME** is well set up and pointing to our JDK directory. Let's assume we have a running and stable java environment.

As a second step, we download Maven from its official website: http://maven.apache.org/download.cai.

Downloading Apache Maven 3.3.9

Apache Maven 3.3.9 is the latest release and recommended version for all users.

The currently selected download mirror is **http://www-eu.apache.org/dist/**. If you encounter a problem with this mirror, please select another mirror. If all mirrors are failing, there are *backup* mirrors (at the end of the mirrors list) that should be available. You may also consult the complete list of mirrors.

Other mirrors: http://www-eu.apache.org/dist/ Change

System Requirements

Java Development Kit (JDK)	Maven 3.3 requires JDK 1.7 or above to execute - it still allows you to build against 1.3 and other JDK versions by Using Toolchains
Memory	No minimum requirement
Disk	Approximately 10MB is required for the Maven installation itself. In addition to that, additional disk space will be used for your local Maven repository. The size of your local repository will vary depending on usage but expect at least 500MB.
Operating System	No minimum requirement. Start up scripts are included as shell scripts and Windows batch files.

Files

Then we unzip it in any directory of our preference, and we add the BIN directory to our PATH variable.

At this time, if everything went well, we should be able to see in console the installation of Maven by running the following command: **mvn -version**.

```
santiago.hernandez@glb-l0899-2:~/workspace$ mvn -version
Apache Maven 3.1.1 (0728685237757ffbf44136acec0402957f723d9a; 2013-09-17 12:22:22-0300)
Maven home: /usr/local/Cellar/maven/3.1.1/libexec
Java version: 1.8.0_45, vendor: Oracle Corporation
Java home: /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home/jre
Default locale: es_ES, platform encoding: UTF-8
OS name: "mac os x", version: "10.11.6", arch: "x86_64", family: "mac"
```

2.1.1 - Creating a Maven Project

Maven allows us to create a project and manage it in a very simple way. To start creating our project, we simply create an empty directory in our workspace, and then, within that directory, we open a command string and we run:

mvn archetype:generate -DgroupId=com.automation.training -DartifactId=autotraining -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

2.1.2 - Maven Archetypes

Essentially, an archetype is an original pattern or model on which we can develop all those things that are of the same type. The may be considered templates that are parameterized or configured to use certain technologies, used by programmers as a basis for writing and organizing the application code.

For the previous example, we use a generic archetype, since we will not need any special configuration.

2.1.3 - Creating the Project

Once the folder with everything we need has been generated, we enter the directory and ask Maven to compile the project for us. For that, we use the following command: mvn clean install.

```
santiago.hernandez@glb-l0899-2:~/Documents/workspace$ cd autotraining/
santiago.hernandez@glb-l0899-2:~/Documents/workspace/autotraining$ mvn clean install
```

This will download all the dependencies to the .m2 folder, which is where Maven creates the local repository with dependencies. In this way, there is no need to download them again every time we create a project that uses a library that we already use in another, and also, this allows the entire configuration to be managed in a centralized file called POM.xml, which we will see later.

2.2 - Adding our Dependencies

POM file

This file allows us to centralize the entire structure and configuration of our project. Here we can set up the configurations for compilation, environments, proxy, dependencies, profiles, etc.

Once the project has been created and the mvn clean install command has been executed, we will see that within our directory several files have been created. See below:

Two directories, src and target, have been created, within which we will have the source code of our application and the compilations generated, respectively.

In addition, we see the **pom.xml** file has been created which, as we said, has project information. Let’s see its content:

```
santiago.hernandez@glb-l0899-2:~/Documents/workspace/autotraining$ cat pom.xml
<project 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/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.automation.training</groupId>
  <artifactId>autotraining</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>autotraining</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

As we can observe, the file is an xml where we have the following values, as configured when we created the project:

GroupId: It is the group ID previously created
ArtifactId: It is the artifact ID previously created
Version: It is the compiled version of the developing application
Dependencies: It is where we will specify the dependencies that we need in our project.

2.2.1 - Adding the Necessary Dependencies

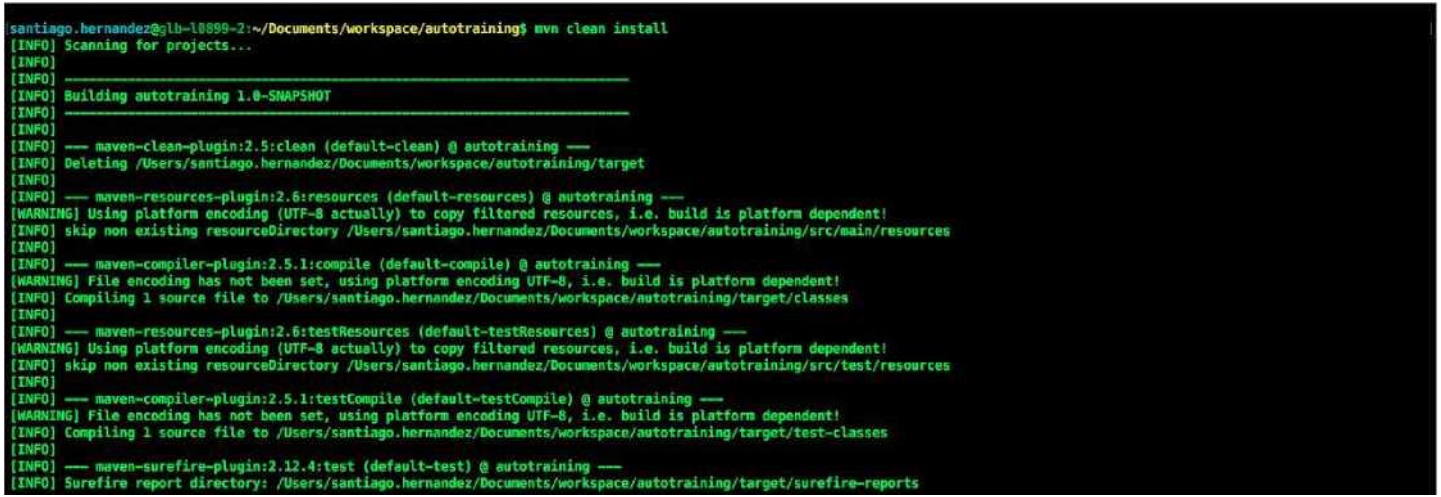
We will add the dependencies that we need in order to work in our automation project, which initially will be selenium webdriver and testng. Then we will see what each of them does and how they are used. For now, let’s configure our project:

For this, we will edit the **pom.xml** and we will add both dependencies; it should look like this:

```
santiago.hernandez@glb-l0899-2:~/Documents/workspace/autotraining$ cat pom.xml
<project 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/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.automation.training</groupId>
  <artifactId>autotraining</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>autotraining</name>
  <url>http://maven.apache.org</url>
  <dependencies>
  <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>3.0.1</version>
    </dependency>
    <dependency>
          <groupId>org.testng</groupId>
          <artifactId>testng</artifactId>
          <version>6.8</version>
        </dependency>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

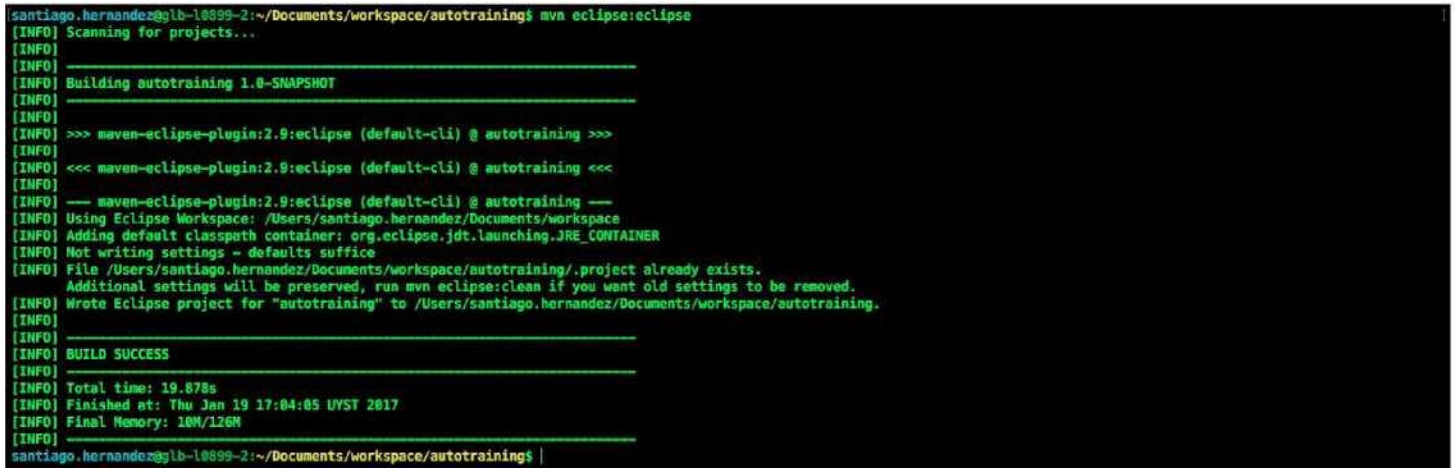
2.3 - Compiling the Project

To compile the project we run the command previously learned; if you do not have the selenium and testng libraries already, it is going to take a little more than usual since each of them will be downloaded:



2.4 - Creating a Project for Eclipse

To create our project in eclipse, what we need to do is run a Maven command that will take care of having our environment working:



Once this step is completed, we will import our project in Eclipse.

Summary of the Level

In this level, we have learned to use Maven’s basic commands to create a project, add its dependencies and compile it.

This helps us have all the necessary setup to start with Selenium WebDriver.