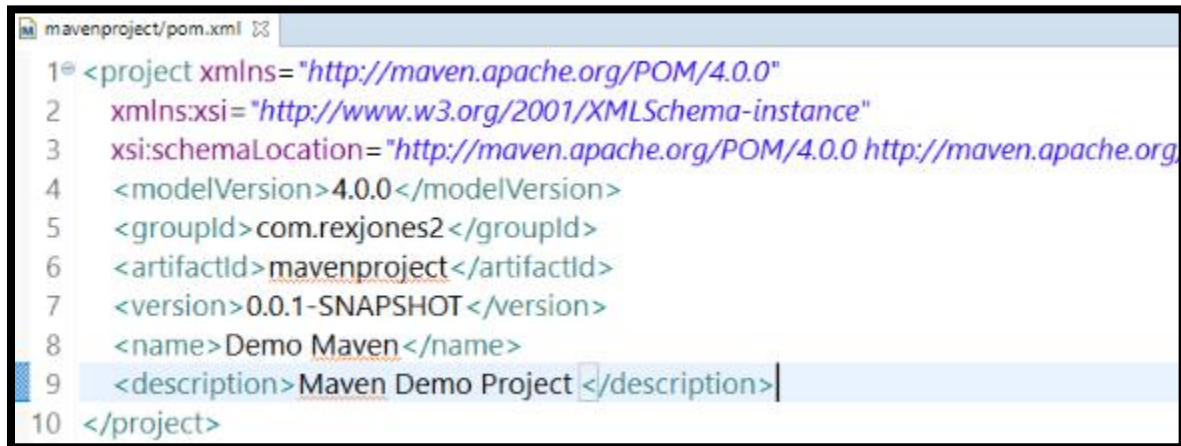


## (Transcript)

# Add Dependencies & Execute First Test

### Add Maven Dependencies & Plugins

To add a dependency for Maven, we double click the pom.xml file then navigate to the pom.xml tab. Let's breakdown this file. The project element defines the basic schema settings. modelVersion contains the model version of the Project Object Model. Here are the details after creating this project: groupId, artifactId, version, name, and description.



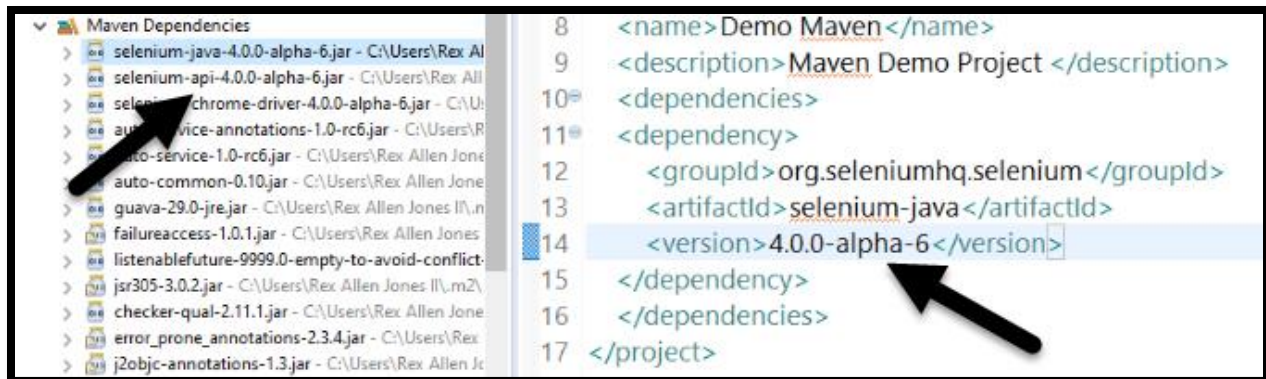
```
1 <project xmlns="http://maven.apache.org/POM/4.0.0"
2   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org
4   <modelVersion>4.0.0</modelVersion>
5   <groupId>com.rexjones2</groupId>
6   <artifactId>mavenproject</artifactId>
7   <version>0.0.1-SNAPSHOT</version>
8   <name>Demo Maven</name>
9   <description>Maven Demo Project</description>
10  </project>
```

Now, let's add the dependencies element by clicking CTRL + SPACE. Navigate to dependencies and the description shows "This element describes all of the dependencies associated with a project". For starters, we know our project will depend on Selenium and it will depend on TestNG. The advantage of a pom.xml file is to bypass downloading the jars then configuring the build path to include those jars. Some people perform that same process over and over whenever their project requires updated jars.

### Add Selenium Dependency

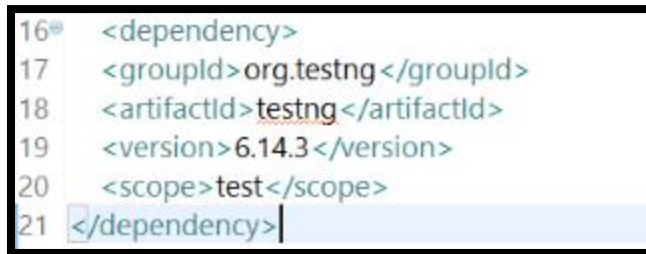
With this pom.xml file, all we do is click CTRL + SPACE, navigate to Insert dependency, type Selenium and we see a list of Selenium options. Expand selenium-java then select the version. I'm going to select 3.141.5. We see the Group Id, Artifact Id, Version, and Scope is compile. Click OK, then we Click Save. Maven Dependencies show Selenium Java 3.141.5 jars.

We can always update the version number. Highlight the version and click CTRL + SPACE then select a different version "Selenium 4-alpha-6". Watch what happens when I click Save. Automatically Maven Dependencies update to show Selenium 4 jars.



## Add TestNG Dependency

Let's add the TestNG dependency which is a Test Framework for Java. This time, we are going to Maven's Repository and search for TestNG, select TestNG. I'm going to use the last 6 version: 6.14.3. After selecting the version, we copy the dependency. Go back to Eclipse and paste the dependency.



## Add Maven-Compiler Plugin

We also have to add the Compiler plugin for Maven. Go to Maven's Repository then Search for compiler plugin and select it. The most recent version is 3.8.1. Copy the GroupID, artifactId, and version. I did not select dependency because it's really a plugin. Go back to Eclipse and after dependencies. We click CTRL + SPACE, select build, CTRL + SPACE again, select plugins, for the last time we click CTRL + SPACE then plugin, and paste. CTRL + SHIFT + F format this pom.xml file. Let me show you something else. Do you see how JRE System Library shows Java SE-10? We must add the release after the version. Click CTRL + SPACE, select Configuration, CTRL + SPACE for the last time, select Release, then enter 10 and save. The purpose of Maven's Compiler Plugin is to compile the sources of our project.

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.1</version>
      <configuration>
        <release>10</release>
      </configuration>
    </plugin>
  </plugins>
</build>
</project>
```

The purpose of Maven's  
Compiler Plugin is to compile  
the sources of our project

## Execute First Maven Project

### Run From Eclipse

Now, let's start with creating a Test Script. We go to src/test/java > right click > New > Class. Class is GoogleTitleTest. WebDriver driver / @Test / public void testGoogleTitle (). Import WebDriver, now we write

```
public class GoogleTitleTest {
    WebDriver driver;

    @Test
    public void testGoogleTitle () {
        System.setProperty("webdriver.chrome.driver",
            "C:\\Users\\Rex Allen Jones II\\Downloads\\Drivers\\chromedriver.exe");
        driver = new ChromeDriver ();
        driver.manage().window().maximize();
        driver.get("https://www.google.com");
        System.out.println("Page Title: " + driver.getTitle());
        driver.quit();
    }
}
```

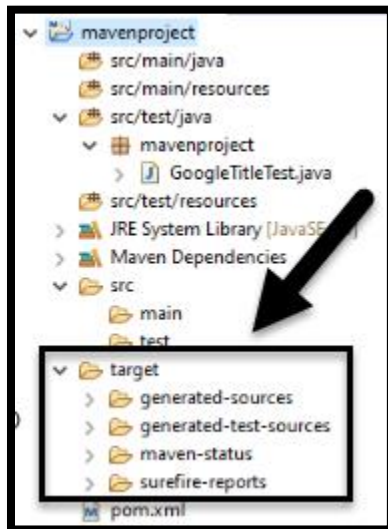
This code sets the property for chromedriver, create a new ChromeDriver, maximize the window, load google search page, print the title and quit the driver. We can always Run this test by clicking the green

arrow but let's run this test as a Maven Test. Also, let me update the project, by right clicking Maven, Update Project and click OK. Now, let's run this Test as a Maven Project by right clicking, go to Run As, Maven Test. There's 1 more thing I want to show you.

Notice how the target folder is empty. However, it will populate with content after running because it stores all output. In the console, we see Page Title: Google, Test Run 1, No Failures, No Errors, and nothing Skipped.

```
Page Title: Google
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 16.976 sec
```

Now, let's open the target folder and it has data.



That's it for running the test as a Maven Test, Executing our First Project, also adding the Selenium and TestNG Dependencies.

## Social Media Contact

- YouTube <https://www.youtube.com/c/RexJonesII/videos>
- Twitter <https://twitter.com/RexJonesII>
- LinkedIn <https://www.linkedin.com/in/rexjones34/>
- GitHub <https://github.com/RexJonesII/Free-Videos>
- Facebook <http://facebook.com/JonesRexII>