**Maven - Build Automation tool**

**What is Build tool?**

A build tool is a tool that automates everything related to building the project to get a piece of software setup. Building software project typically includes below activities:

Generating documentation from the source code.

compiling the source code.

packaging compiled code into JAR files or ZIP files.

Etc. [like generating source code, continuous integration]

And, above activities are required to complete the final software product so how about the automating the above activities?

**Automating Advantages:**

No human errors

faster build of project and deployment

**Various Build Tools**

Shell Scripts.

Gradle.

Make.

Ant.

Maven.

Rake.

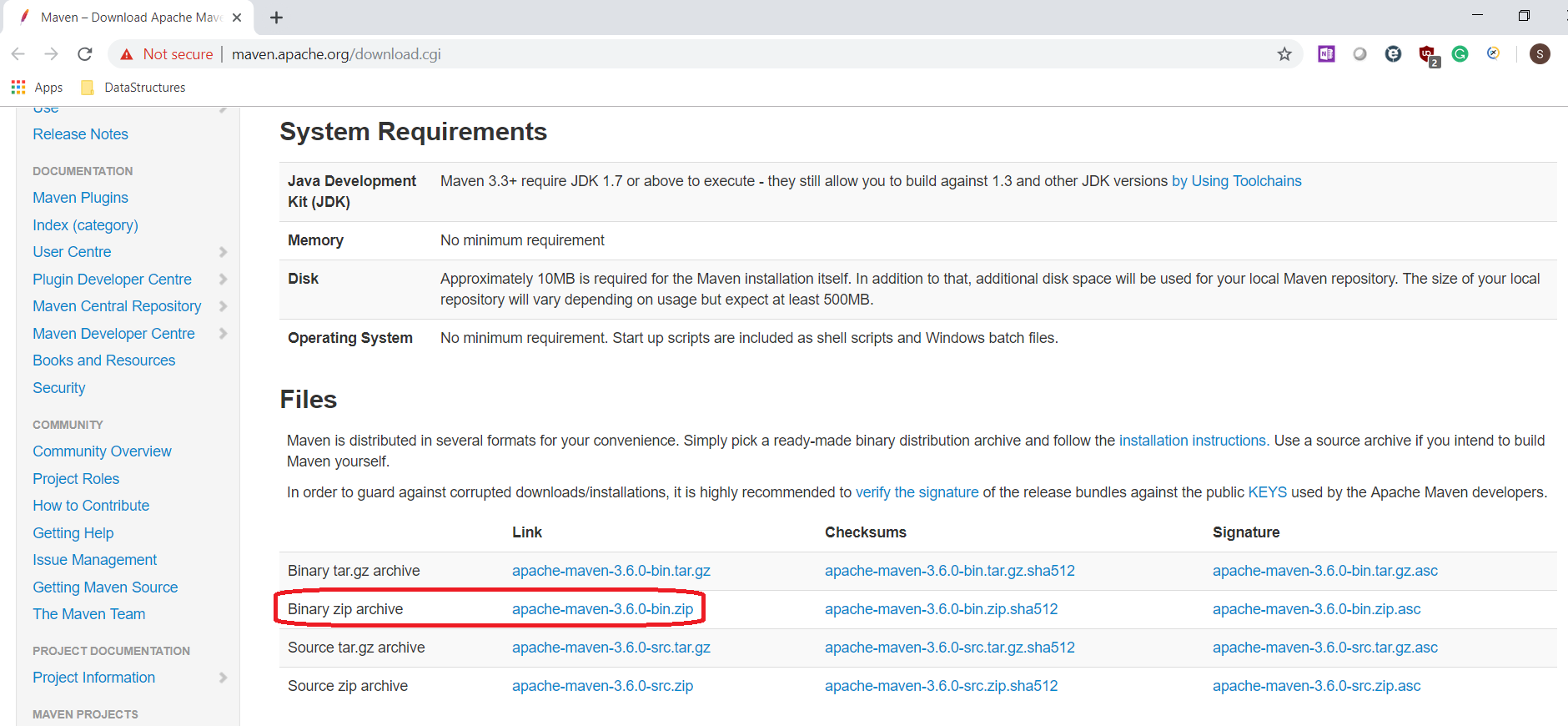
Grunt.

Gulp.

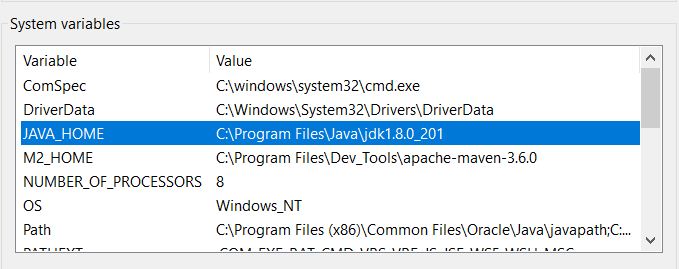
SBT.

**Installing Maven**

To install Maven, go to the Maven page (http://maven.apache.org/download.cgi) and download the file

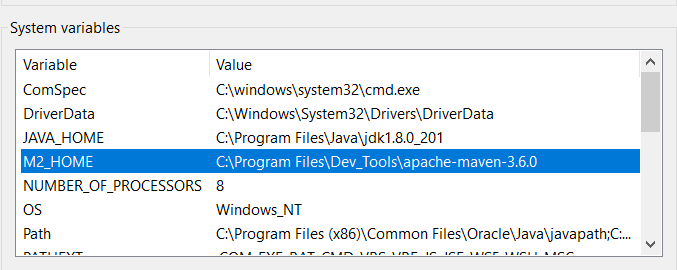


Set the JAVA\_HOME environment variable to point to a valid Java SDK (e.g. Java 8).



**Download and unzip Maven.**

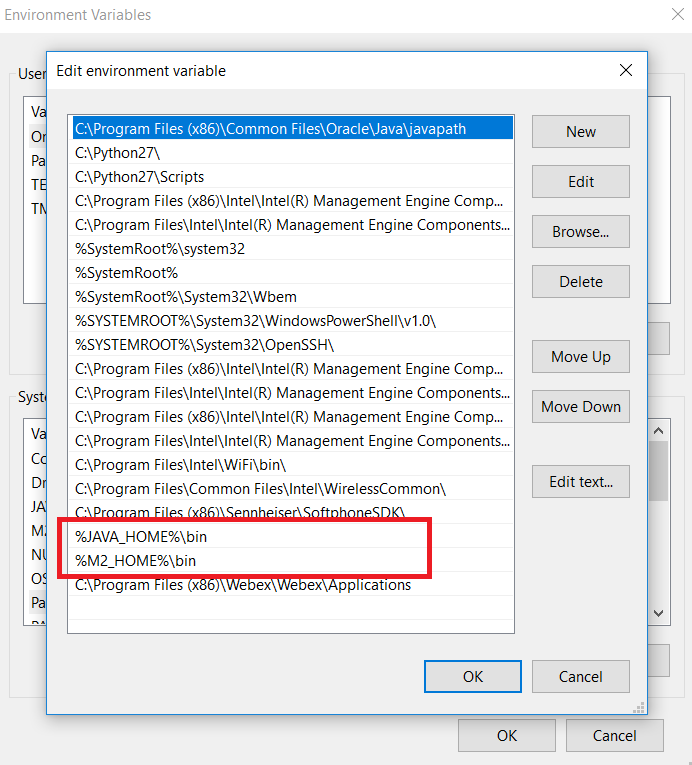
Set the M2\_HOME environment variable to point to the directory you unzipped Maven to.

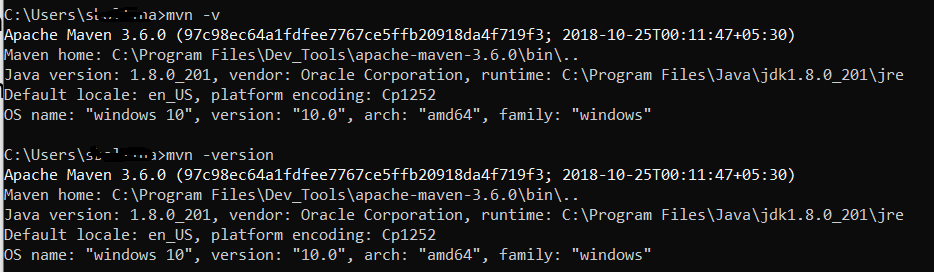


Set the M2 environment variable to point to M2\_HOME/bin (%M2\_HOME%\bin on Windows, $M2\_HOME/bin on unix).

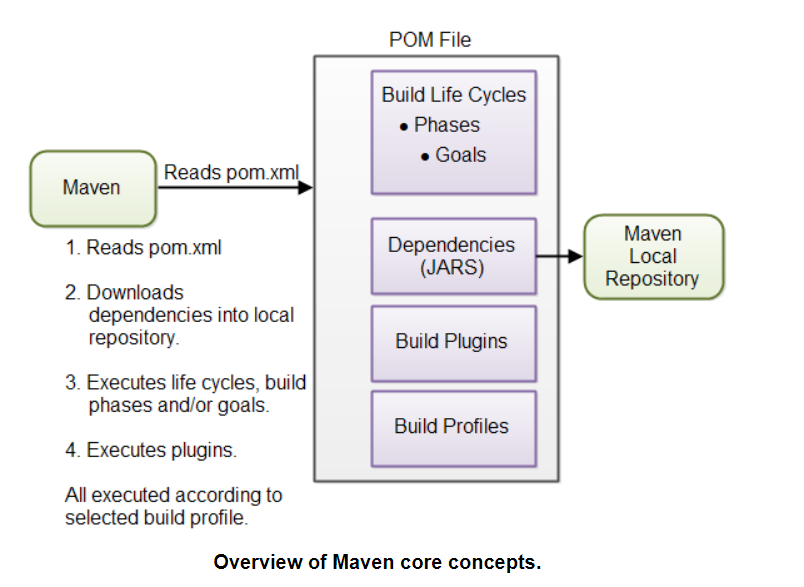
Add M2 to the PATH environment variable (%M2% on Windows, $M2 on unix).

Open a command prompt and type **mvn -version** and press enter.





**Maven Overview**

****

**POM Files -**  uses maven commands to execute the resources described in the POM

A Maven POM file (Project Object Model) is an XML file that describe the resources of the project. This includes the directories where the source code, test source etc. is located in, what external dependencies (JAR files) your projects has etc.

The POM file describes what to build, but most often not how to build it. How to build it is up to the Maven build phases and goals. You can insert custom actions (goals) into the Maven build phase if you need to, though.

Each project has a POM file. The POM file is named pom.xml and should be located in the root directory of your project. A project divided into subprojects will typically have one POM file for the parent project, and one POM file for each subproject. This structure allows both the total project to be built in one step, or any of the subprojects to be built separately.

<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/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.jenkov</groupId>

<artifactId>java-web-crawler</artifactId>

<version>1.0.0</version>

</project>

The **modelVersion** element sets what version of the POM model you are using.

The **groupId** element is a unique ID for an organization, or a project. The group ID is almost similar to the root Java package name of the project.

The **artifactId** element contains the name of the project you are building.

The **versionId** element contains the version number of the project.

**Project Dependencies**

Maven has built-in dependency management, The external libraries specified in the POM will be downloaded to local repository

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

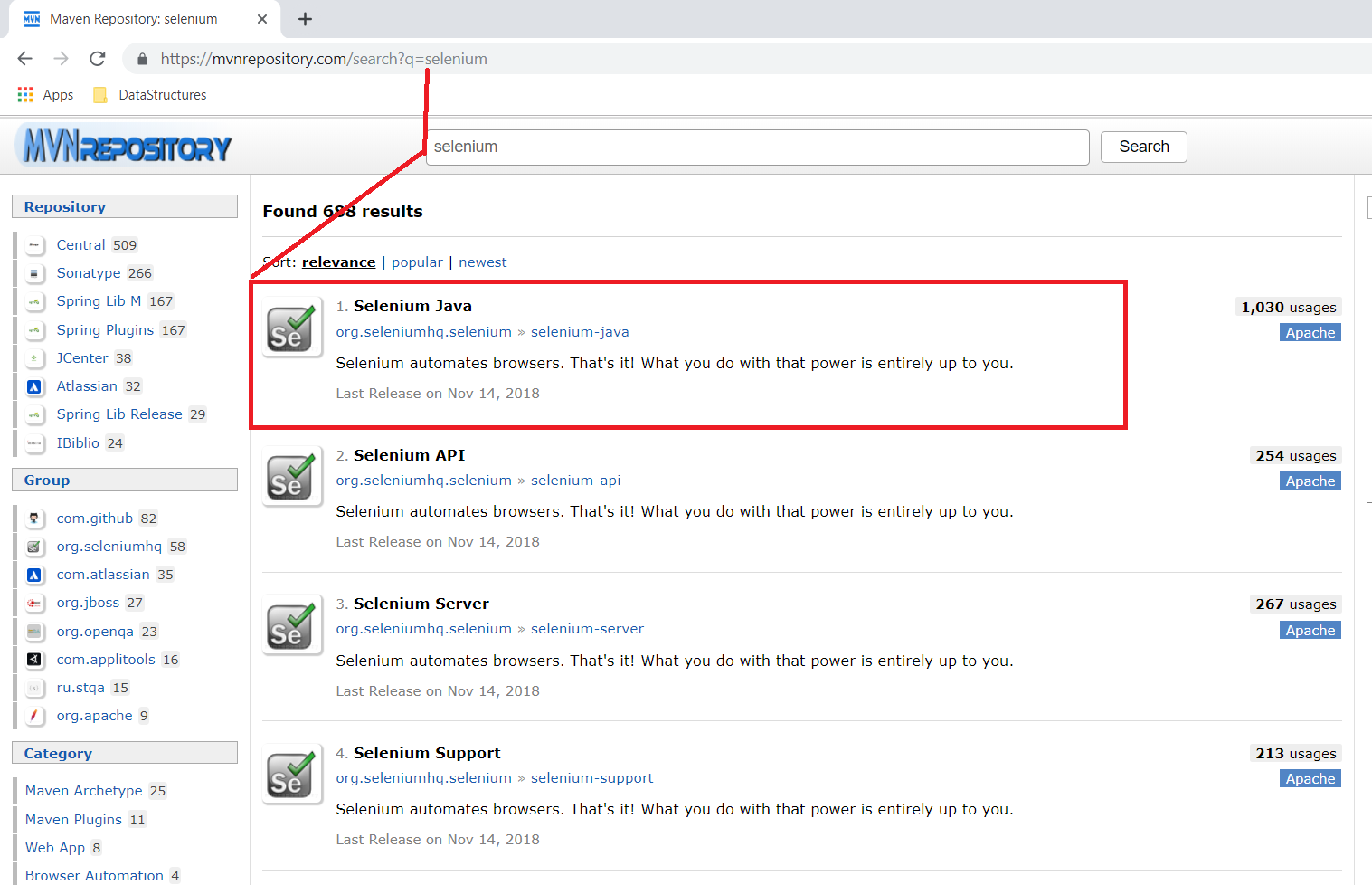
<version>4.8.1</version>

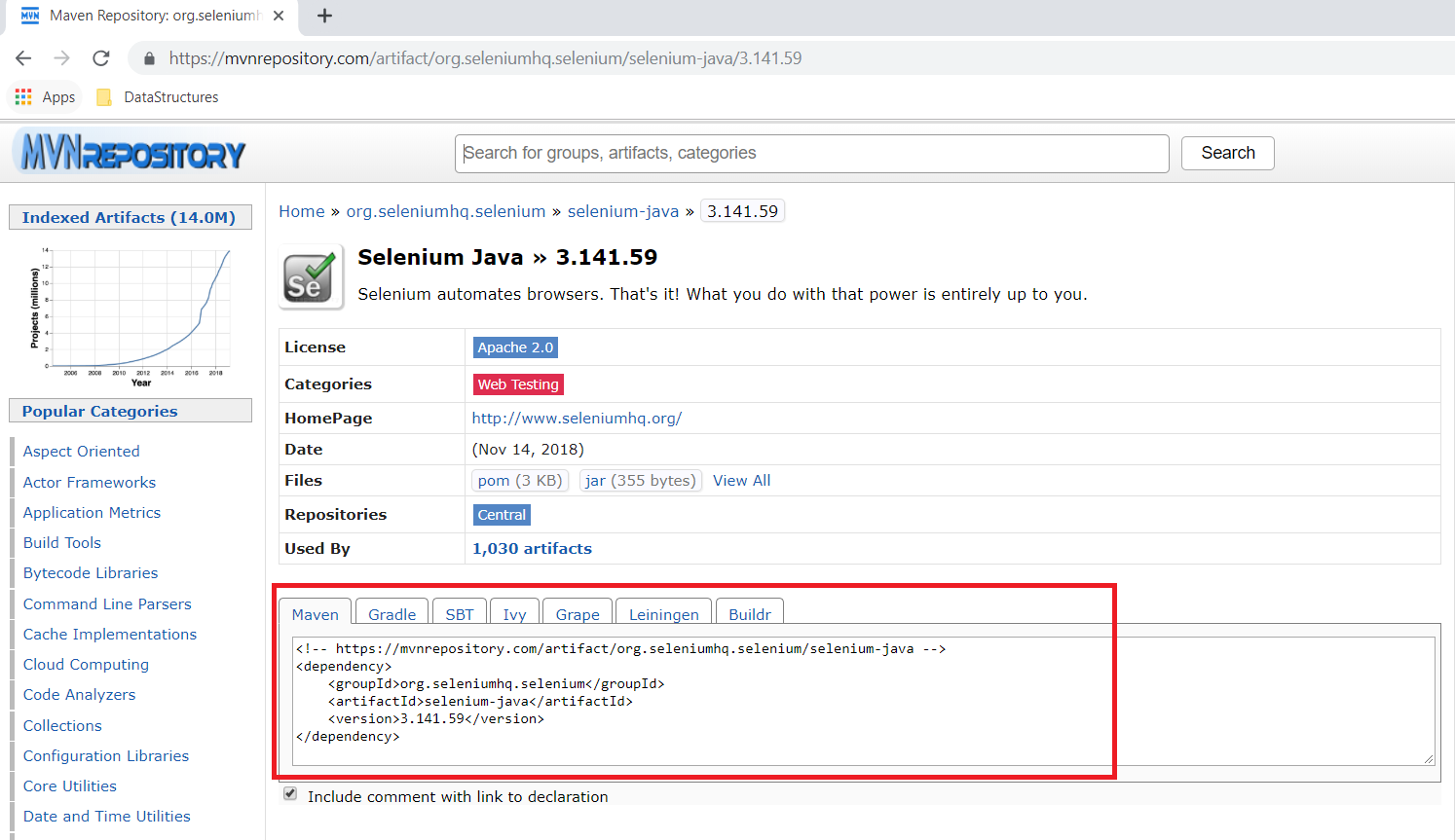
<scope>test</scope>

</dependency>

</dependencies>

**Maven Repo**

****

****

**<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->**

<dependency>

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

<artifactId>selenium-java</artifactId>

<version>3.141.59</version>

</dependency>

**Build Phase Description**

validate Validates that the project is correct and all necessary information is available. This also makes sure the dependencies are downloaded.

compile Compiles the source code of the project.

test Runs the tests against the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed.

package Packs the compiled code in its distributable format, such as a JAR.

install Install the package into the local repository, for use as a dependency in other projects locally.

deploy Copies the final package to the remote repository for sharing with other developers and projects.

**Build Life Cycles, Phases and Goals –** [commands clean compile install test and others]

The build process in Maven is split up into build life cycles, phases and goals. A build life cycle consists of a sequence of build phases, and each build phase consists of a sequence of goals. When you run Maven you pass a command to Maven. This command is the name of a build life cycle, phase or goal. If a life cycle is requested executed, all build phases in that life cycle are executed. If a build phase is requested executed, all build phases before it in the pre-defined sequence of build phases are executed too.

The build Maven Life Cycle is divided into 3 types:

**Default Maven Life Cycle** is the general model in build process, which builds, tests and distributes the project.

**Clean Lifecycle** is the simple life cycle in Maven that has 3 phases. All the three are invoked one by one when clean life cycle command mvn clean is executed.

**Site Life** Cycle helps in project documentation and reporting process.

**Dependencies and Repositories**

One of the first goals Maven executes is to check the dependencies needed by your project. Dependencies are external JAR files (Java libraries) that your project uses. If the dependencies are not found in the local Maven repository, Maven downloads them from a central Maven repository and puts them in your local repository. The local repository is just a directory on your computer's hard disk. You can specify where the local repository should be located if you want to. You can also specify which remote repository to use for downloading dependencies.

**Build Plugins**

Build plugins are used to insert extra goals into a build phase. If you need to perform a set of actions for your project which are not covered by the standard Maven build phases and goals, you can add a plugin to the POM file. Maven has some standard plugins you can use, and you can also implement your own in Java if you need to.

**Build Profiles**

Build profiles are used if you need to build your project in different ways. For instance, you may need to build your project for your local computer, for development and test. And you may need to build it for deployment on your production environment. These two builds may be different. To enable different builds you can add different build profiles to your POM files. When executing Maven you can tell which build profile to use.

**Selenium Project Dependencies**

<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->

<dependency>

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

<artifactId>selenium-java</artifactId>

<version>3.11.0</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

</dependency>

<!-- Extent Reports -->

<dependency>

<groupId>com.vimalselvam</groupId>

<artifactId>cucumber-extentsreport</artifactId>

<version>3.0.2</version>

</dependency>

<dependency>

<groupId>com.aventstack</groupId>

<artifactId>extentreports</artifactId>

<version>3.1.2</version>

</dependency>

**References:**

<https://maven.apache.org/>

<https://www.mkyong.com/tutorials/maven-tutorials/>

<https://maven.apache.org/guides/index.html>

<https://www.splessons.com/lesson/maven-tutorial/>