**What is Maven?**

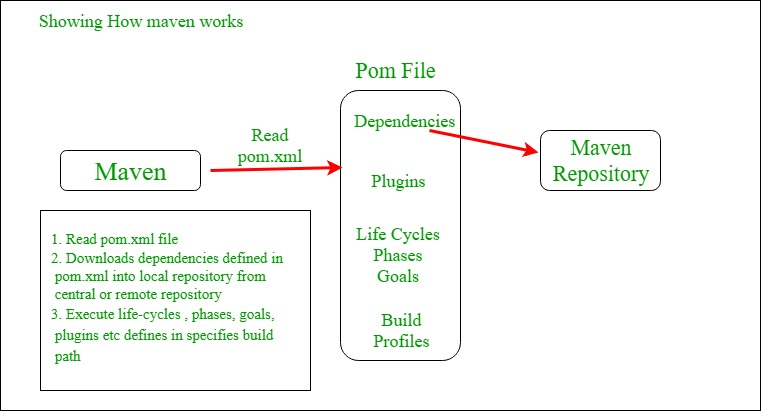
Maven is a powerful project management tool that is based on POM (project object model). It is used for projects build, dependency and documentation. It simplifies the build process like ANT. But it is too much advanced than ANT.  
In short terms we can tell maven is a tool that can be used for building and managing any Java-based project. maven make the day-to-day work of Java developers easier and generally help with the comprehension of any Java-based project.

**What maven does?**

Maven does a lot of helpful task like

1. We can easily build a project using maven.
2. We can add jars and other dependencies of the project easily using the help of maven.
3. Maven provides project information (log document, dependency list, unit test reports etc.)
4. Maven is very helpful for a project while updating central repository of JARs and other dependencies.
5. With the help of Maven we can build any number of projects into output types like the JAR, WAR etc without doing any scripting.
6. Using maven we can easily integrate our project with source control system (such as Subversion or Git).

**How maven works?**

  
**Core Concepts of Maven:**

1. **POM Files:** Project Object Model(POM) Files are XML file that contains information related to the project and configuration information such as dependencies, source directory, plugin, goals etc. used by Maven to build the project. When you should execute a maven command you give maven a POM file to execute the commands. Maven reads pom.xml file to accomplish its configuration and operations.
2. **Dependencies and Repositories:** Dependencies are external Java libraries required for Project and repositories are directories of packaged JAR files. The local repository is just a directory on your machine hard drive. 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.
3. **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. Maven command is the name of a build lifecycle, phase or goal. If a lifecycle is requested executed by giving maven command, all build phases in that life cycle are executed also. If a build phase is requested executed, all build phases before it in the defined sequence are executed too.
4. **Build Profiles:** Build profiles a set of configuration values which allows you to build your project using different configurations. For example, you may need to build your project for your local computer, for development and test. To enable different builds you can add different build profiles to your POM files using its profiles elements and are triggered in the variety of ways.
5. **Build Plugins:** Build plugins are used to perform specific goal. 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.

**Maven pom.xml file**

POM means Project Object Model is key to operate Maven. Maven reads pom.xml file to accomplish its configuration and operations. It is an XML file that contains information related to the project and configuration information such as **dependencies**, **source directory**, **plugin**, **goals etc**. used by Maven to build the project.

The sample of 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/xsd/maven-4.0.0.xsd>">             <modelVersion>4.0.0</modelVersion>           <groupId> com.project.loggerapi </groupId>           <artifactId>LoggerApi</artifactId>           <version>0.0.1-SNAPSHOT</version>           <!-- Add typical dependencies for a web application -->         <dependencies>                 <dependency>                         <groupId>org.apache.logging.log4j</groupId>                         <artifactId>log4j-api</artifactId>                         <version>2.11.0</version>                   </dependency>         </dependencies>       </project> |

Elements used for Creating pom.xml file

1. **project-**It is the root element of the pom.xml file.
2. **modelVersion-**modelversion means what version of the POM model you are using. Use version 4.0.0 for maven 2 and maven 3.
3. **groupId-**groupId means the id for the project group. It is unique and Most often you will use a group ID which is similar to the root Java package name of the project like we used the groupId com.project.loggerapi.
4. **artifactId-**artifactId used to give name of the project you are building.in our example name of our project is LoggerApi.
5. **version-**version element contains the version number of the project. If your project has been released in different versions then it is useful to give version of your project.

Other Elements of Pom.xml file

1. **dependencies-**dependencies element is used to defines a list of dependency of project.
2. **dependency-**dependency defines a dependency and used inside dependencies tag. Each dependency is described by its groupId, artifactId and version.
3. **name-**this element is used to give name to our maven project.
4. **scope-**this element used to define scope for this maven project that can be compile, runtime, test, provided system etc.
5. **packaging-**packaging element is used to packaging our project to output types like JAR, WAR etc.

**Pros and Cons of using Maven**

**Pros:**

* 1. Maven can add all the dependencies required for the project automatically by reading pom file.
  2. One can easily build their project to jar, war etc. as per their requirements using Maven.
  3. Maven makes easy to start project in different environments and one doesn’t needs to handle the dependencies injection, builds, processing, etc.
  4. Adding a new dependency is very easy. One has to just write the dependency code in pom file.

**Cons:**

* 1. Maven needs the maven installation in the system for working and maven plugin for the ide.
  2. If the maven code for an existing dependency is not available, then one cannot add that dependency using maven.

**When should someone use Maven?**

One can use the Maven Build Tool in the following condition:

* 1. When there are a lot of dependencies for the project. Then it is easy to handle those dependencies using maven.
  2. When dependency version update frequently. Then one has to only update version ID in pom file to update dependencies.
  3. Continuous builds, integration, and testing can be easily handled by using maven.
  4. When one needs an easy way to Generating documentation from the source code, Compiling source code, Packaging compiled code into JAR files or ZIP files.