**What is a Build Tool?**

A build tool is a tool that automates everything related to building the software project. Building a software project typically includes one or more of these activities:

* Generating source code (if auto-generated code is used in the project).
* Generating documentation from the source code.
* Compiling source code.
* Packaging compiled code into JAR files or ZIP files.
* Installing the packaged code on a server, in a repository, or somewhere else.

Any given software project may have more activities than those needed to build the finished software. Such activities can normally be plugged into a build tool, so these activities can be automated too.

The advantage of automating the build process is ***that you minimize the risk of humans making errors while building the software manually***. Additionally, an automated build tool is typically faster than a human performing the same steps manually.

**Maven Website**

The Maven website is located here:

[**http://maven.apache.org**](http://maven.apache.org/)

From this website, you can download the latest version of Maven and follow the project in general.

**Installing Maven**

To install Maven on your own system (computer), go to the [**Maven download page**](http://maven.apache.org/download.cgi) and follow the instructions there. In summary, what you need to do is:

1. Set the JAVA\_HOME environment variable to point to a valid Java SDK (e.g. Java 8).
2. Download and unzip Maven.
3. Set the M2\_HOME environment variable to point to the directory you unzipped Maven to.
4. Set the M2 environment variable to point to M2\_HOME/bin (%M2\_HOME%\bin on Windows, $M2\_HOME/bin on unix).
5. Add M2 to the PATH environment variable (%M2% on Windows, $M2 on unix).
6. Open a command prompt and type 'mvn -version' (without quotes) and press enter.

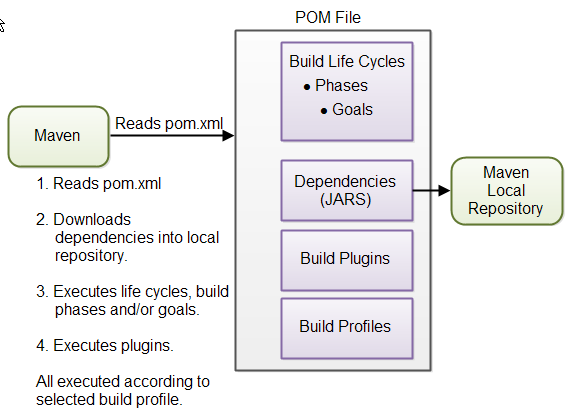
After typing in the mvn -version command you should be able to see Maven execute, and the version number of Maven written out to the command prompt.

Note: Maven uses Java when executing, so you need Java installed too (and the JAVA\_HOME environment variable set as explained above).

**Maven Overview - Core Concepts**

Maven is centered around the concept of POM files (Project Object Model). A POM file is an XML representation of project resources like source code, test code, dependencies (external JARs used) etc. The POM contains references to all of these resources. The POM file should be located in the root directory of the project it belongs to.

Here is a diagram illustrating how Maven uses the POM file, and what the POM file primarily contains:



|  |
| --- |
| **Overview of Maven core concepts.** |

These concepts are explained briefly below to give you an overview, and then in more detail in their own sections later in this tutorial.

**POM Files**  
When you execute a Maven command you give Maven a POM file to execute the commands on. Maven will then execute the command on the resources described in the POM.

**Build Life Cycles, Phases and Goals**  
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.

**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 (I do). You can also specify which remote repository to use for downloading dependencies. All this will be explained in more detail later in this tutorial.

**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.

**Note**: Ant is another popular build tool by Apache. If you are used to Ant and you are trying to learn Maven, you will notice a difference in the approach of the two projects.