Maven Glossary

This document describes some of the most common terms encountered while using Maven. These terms, that have an explicit meaning for Maven, can sometimes be confusing for newcomers.

- **Project**: Maven thinks in terms of projects. Everything that you will build are projects. Those projects follow a well defined "Project Object Model". Projects can depend on other projects, in which case the latter are called "dependencies". A project may consistent of several subprojects, however these subprojects are still treated equally as projects.
- **Project Object Model (POM)**: The Project Object Model, almost always referred as the POM for brevity, is the metadata that Maven needs to work with your project. Its name is "project.xml" and it is located in the root directory of each project.
- **Artifact**: An artifact is something that is either produced or used by a project. Examples of artifacts produced by Maven for a project include: JARs, source and binary distributions, WARs. Each artifact is uniquely identified by a group id and an artifact ID which is unique within a group.
- **GroupId**: A group ID is a universally unique identifier for a project. While this is often just the project name (eg. commons-collections), it is helpful to use a fully-qualified package name to distinguish it from other projects with a similar name (eg. org.apache.maven).
- **Dependency**: A typical Java project relies on libraries to build and/or run. Those are called "dependencies" inside Maven. Those dependencies are usually other projects' JAR artifacts, but are referenced by the POM that describes them.
- **Plug-in**: Maven is organized in plugins. Every piece of functionality in Maven is provided by a plugin. Plugins provide goals and use the metadata found in the POM to perform their task. Examples of plugins are: jar, eclipse, war. Plugins are primarily written in Java, but Maven also supports writing plug-ins in Beanshell and Ant Scripting.
- **Mojo**: A plugin written in Java consists of one or more mojos. A mojo is a Java class that implements the org.apache.maven.plugin.Mojo interface. This means that a mojo is the implementation for a goal in a plugin.
- **Repository**: Refer to Introduction to Repositories
- **Snapshots**: Projects can (and should) have a special version including SNAPSHOT to indicate that they are a "work in progress", and are not yet released. When a snapshot dependency is encountered, it is always looked for in all remote repositories, and downloaded again if newer than the local copy. The version can either be the string SNAPSHOT itself, indicating "the very latest" development version, or something like 1.1-SNAPSHOT, indicating development that will be released as 1.1 (i.e. newer than 1.0, but not yet 1.1).
- **APT**: APT is a wiki-like format of documentation that Maven currently understands. For information on how to create APT files, refer to the Guide to creating a site document.
- **XDoc**: XDoc is the format of documentation that Maven currently understands. It is quite simple, and allows embedding XHTML within a simple layout that is transformed into a uniform site. For information on how to create XDoc files, refer to the Guide to creating a site document.