Maven

Introduction of Maven

- Maven is
  - An Yiddish Word – means “Accumulator of Knowledge”.
  - A project management tool.
  - A complete build lifecycle framework.
  - Based on concept of Project Object Model (POM files)

A POM file is XML representation of project resources.

Environment Setup

- Step 1 :
  - Verify JAVA_HOME environment variable is set and points to JDK installation.
- Step 2 :
  - Download latest version of Maven Archive from http://maven.apache.org/download.cgi
- Step 3 :
  - Extract the Maven Archive
- Step 4 :
  - Set Maven Environment Variables
    - M2_HOME=C:\Program Files\Apache Software Foundation\apache-maven.3.3.9
    - M2=%M2_HOME%\bin

Maven vs Ant

Maven	Ant
Has a Formal Convention to place source code, compiled code etc.	No Formal Convention – provide information of project structure in build.xml file
It is declarative – everything you define in pom.xml	It is procedural – need to provide information about what to do and when to do
It is a Framework	It is a Toolbox
Project Management Tool	Mainly a build tool
It has life cycle.	**No life cycle**

Maven POM File

- An XML file that describes the resources of the project like source code, test code, dependencies(external jars used) etc.
- It contains all the references of all the resources.
- It is located in the root folder of the project.
- Before maven 2, it was named as project.xml file. But, since maven 2 it is renamed as pom.xml.

Elements of POM file

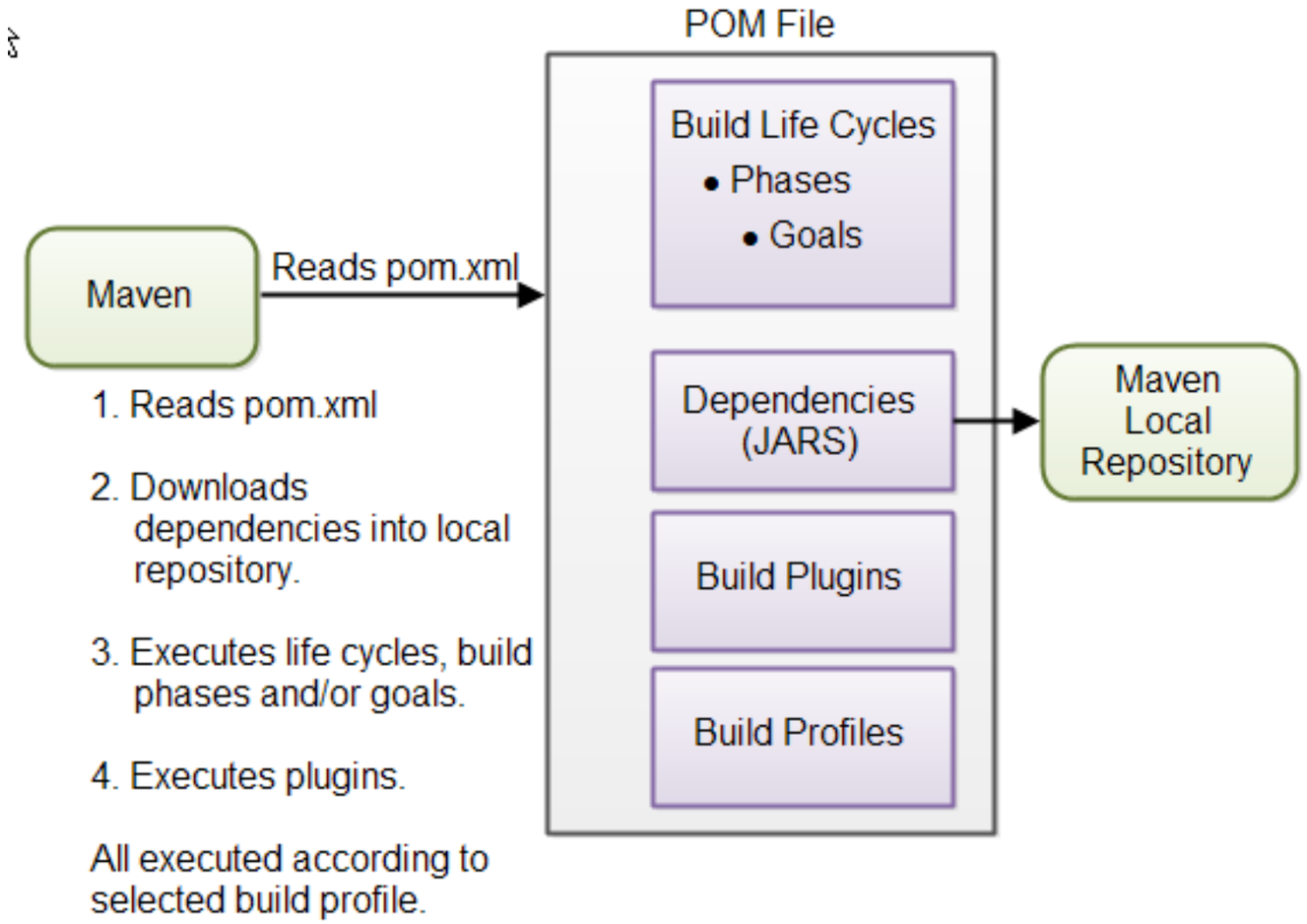
Element	Description
project	Root element of pom.xml file
modelVersion	Sub element of project. It specifies the modelVersion. Example 4.0.0
groupId	Sub element of project. It specifies the id for the project group.
artifactId	Sub element of project. It specifies the id for the artifact (project). Examples of artifacts produced by Maven for a project include: JARs, source and binary distributions, and WARs.
version	Sub element of project. It specifies the version of the artifact under given group.
Packaging	Define package type such as jars, wars etc
name	Name of the maven project
url	url of the project
Dependencies	Define Dependencies of the project
Dependency	Define dependency of the project
Scope	defines scope for this maven project. It can be compile, provided, runtime, test and system.

Example : 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.javatpoint.application1</groupId>
  <artifactId>my-application1</artifactId>
  <version>1.0</version>
  <packaging>jar</packaging>

  <name>Maven Quick Start Archetype</name>
  <url>http://maven.apache.org</url>
```

```
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.8.2</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

</project>
```

How Maven uses POM file?



Running Maven

- To build all the modules and to install it in the local repository. The local repository is created in your home directory (or alternative location that you created it), and is the location that all downloaded binaries and the projects you built are stored.

mvn clean install

Maven Directory Structure

-src

  -main

    -java

    -resources

    -webapp

  -test

    -java

    -resources

-target

Maven Repositories

- A Maven repository is a directory to store all the project jars, library jar, plugins or any other artifacts.

- There are three types of Maven repository.
  - Local
    - Is a local folder in your machine
    - Is created when any maven command is given first time
  - Central
    - Is repository managed by Maven Community
    - Contains a large number of commonly used libraries
  - Remote
    - Custom repository contains required libraries (not able in central repository)

Project Dependencies

- Maven has built-in dependency management.
- Specify in the POM file what external libraries your project depends on, and which version, and then Maven downloads them for you and puts them in your local Maven repository.
- Specify project dependencies inside the **<dependencies>** element in POM file.
- External Dependency
  - It is a dependency (jar file) which is not located in a Maven repository (neighter local, remote or external)
  - It may be located some where at you local hard disk.
  - External means external to Maven Repository System

Maven Goal

- A **goal**
  - represents a specific task which contributes to the building and managing of a project.
  - may be bound to zero or more build phases.
  - not bound to any build phase could be executed outside of the build lifecycle by direct invocation.

Maven Build Life Cycle

- A Maven build follow a lifecycle. Maven has following three standard lifecycles:
  - Clean – handles project cleaning
  - Default (or build) – handles project deployment
  - Site – handles creation of project's side documentation
- Default lifecycle
  - generate-sources/generate-resources
  - compile
  - test
  - package
  - integration-test (pre and post)
  - Install
  - deploy

Example Maven Goals

- To invoke a Maven build you set a lifecycle “goal”
- mvn install
  - Invokes generate and compile, test, package, integration-test, install
- mvn clean
  - Invokes just clean
- mvn clean compile
  - Clean old builds and execute generate, compile
- mvn compile install
  - Invokes generate, compile, test, integration-test, package, install
- mvn test clean
  - Invokes generate, compile, test then cleans

Creating Web Application by using Maven

- To create Web Application using Maven, execute **archetype:generate** command of **mvn tool** at command prompt**.**
- **Syntax :**

mvn archetype:generate -DgroupId=<project-packaging> -DartifactId=<project-name>
-DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=<booleanValue>

- **Example :**

mvn archetype:generate -DgroupId=com.myweb -DartifactId=MyProject
-DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false

Introduction to PMD

- PMD stands for Programming Mistake Detector.
- It is a free source code analysis tool which helps you to find the bugs in your java code and improve the code quality.
- PMD helps to find empty try/catch/finally statements, while/if statements, unnecessary object creation etc.

- PMD warning and problems are not exactly errors, but shows that particular code need to be improved

Configuring PMD

- For configuring with Maven, Apache Maven PMD Plugin is required
  - The PMD Plugin allows you to automatically run the PMD code analysis tool on your project's source code and generate a site report with its results.
  - It also supports the separate Copy/Paste Detector tool (or CPD) distributed with PMD.

Introduction to Checkstyle

- Checkstyle
  - is a development tool to help programmers write Java code that adheres to a coding standard. It
  - automates the process of checking Java code to spare humans of this boring (but important) task.
  - ideal for projects that want to enforce a coding standard.
  - highly configurable and can be made to support almost any coding standard.
    - An example configuration files are supplied supporting the Sun Code Conventions, Google Java Style.

Introduction to FindBugs

- Findbugs
  - is an open source project for a static analysis of the Java bytecode to identify potential software bugs.
  - provides early feedback about potential errors in the code. This helps the developer to access these problems early in the development phase.

- The Findbugs analysis can be integrated into the Eclipse IDE via an additional software component.

- We can integrate FindBugs into our build process by using the **FindBugs Maven plugin**.