```
=============
Apache Maven
=============

=> Maven is a build tool

=> Maven is free & open source s/w

=> Maven is developed using java language

Note: Maven is used only for java projects build process.

=> Maven is used to automate java projects build process.

        1) Download required libraries (ex: spring, junit, hibernate, kafka, redis)

        2) Compile source code of the project

        3) Execute Unit test cases of the project

        4) Package application as a jar or war file


=> stand-alone applications will be packaged as jar file

=> web-applications will be packaged as war file.

=> The main aim of maven is to simplify java projects build process.

=========================
Maven Setup in Windows
=========================

### Reference Video : https://www.youtube.com/watch?v=hV1OWzYpzxo


Step-1 : Install Java 17v + Setup JAVA_HOME + Setup Java Path (in Env variables)

                JAVA_HOME = C:\Program Files\Java\jdk-17

                Path = C:\Program Files\Java\jdk-17\bin

Step-2 : Download maven software as zip file & extract it

                URL : https://maven.apache.org/download.cgi

Step-3 : Copy maven folder into C drive (optional)

Step-4 : Setup MAVEN_HOME + setup MAVEN PATH (in Env variables)

                        MAVEN_HOME = C:\apache-maven-3.9.8

                        Path : C:\apache-maven-3.9.8\bin

Step-5 : Verify maven installation in cmd

                        $ mvn -v


=========================
Maven Setup in Linux
=========================
```

```
$ sudo yum install maven -y

==================
Maven Terminology
==================

1) ArcheType : Type of project (quick-start / web)

2) groupId : Company name (which company developing this project) (ex: in.ashokit)

3) artifactId : Project Name

4) version : SNAPSHOT / RELEASE

5) packaging : jar or war

6) dependencies : libraries (jars)

7) repositories : Location where jars are stored (ex: central / remote / local)

8) goals : To perform build process

                - clean
                - compile
                - test
                - package

9) pom.xml : Project Object Model (maven project config file)

============================
Working with Maven Project
============================

=> We can create maven project in 2 ways

                1) Command Prompt (cmd)

                2) IDE (eclipse/ STS / Intelli J)

================================
Maven Project Creation in CLI
================================

# Open cmd and execute below command to create maven project (quick-start)

mvn -B archetype:generate -DgroupId=in.ashokit -DartifactId=my-first-app -DarchetypeArtifactId=maven-
archetype-quickstart -DarchetypeVersion=1.4


# navigate into project directory

cd my-first-app

# execute maven goals

mvn clean
mvn compile
mvn test
mvn package
mvn install

mvn clean package => clean + compile + test + package

mvn clean install => clean + compile + test + package + install
```

Note: install goal is storing our project jar file in maven local repository

Local Repo Location : C://users/username/.m2

==========
Maven Project Creation in IDE
==========

=> By default maven is integrated with all java based IDEs

Ex: Eclipse, STS, IntelliJ IDEA...

=> We can create maven project directley in above IDEs

==========
Maven Dependencies
==========

=> Libraries required to develop our java applications are called as Maven dependencies.

Ex:

a) spring-core
b) spring-jdbc
c) spring-boot-starter
d) jackson
e) junit
f) log4j

=> We need to add required dependencies in maven project pom.xml file.

=> When we add dependencies in pom.xml file then maven will download those dependencies and will add to project build path.

=> Maven dependencies we can find in below website

url : www.mvnrepository.com

```
 <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>6.1.11</version>
        </dependency>
 </dependencies>
```

Note: After adding dependency in pom.xml file then right click on project -> Maven -> Reload Project.

=> Maven will take care of "transitive-dependency" management.

spring-context => core + beans + aop + jcl + context

Advantage: downloading child dependencies automatically

Dis-advantage: downloading unwanted dependencies also (they will waste jvms memory)

=> If we want remove unwanted child dependencies then we need to use dependency exclusion concept like below

<dependency>

```
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>6.2.8</version>
        <exclusions>
                <exclusion>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-aop</artifactId>
                </exclusion>
        </exclusions>
</dependency>


======================
Maven Repositories
======================

=> Mvn Repository is a location where maven dependencies/libraries will be stored.

=> Maven tool will deal with 3 types of repositories

                1) Central Repository (public)

                2) Remote Repository (private -> nexus/jfrog)

                3) Local Repository (in our machine .m2 folder)


=> Local Repository will be available in our machine

                        Path : C://Users/<name>/.m2


=> Local Repository will be created in our machine (.m2 folder)

=> Central Repository will be maintained by apache organization (public)

=> Remote Repository will be maintained by our company to store shared libraries.

Note: To setup remote repositories we will use Nexus / JFrog softwares.

##### To connect with remote repo we will use settings.xml, need to configure our credentials ####


=> When we add dependency in pom.xml file, maven will search for it in local repo. If not available
then it will search in central repo (it will download from central to local).


================================================================

Gradle Tutorial : https://www.youtube.com/watch?v=I84f9Q5bFBA

================================================================
```