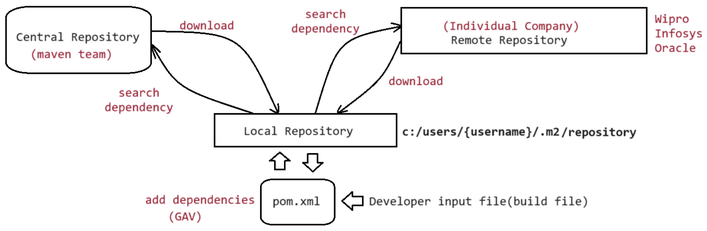
1. Maven Build tool(Developer/Expert)

   a. Creates a Standard folder structure for our project based on developer requirement.
   b. Downloads the dependency from central/remote repository to Local Repository(.m2 folder of our local system).
   c. Adds the required jars to classpath of our project.
   d. Compilation + Execution + Packaging(jar/war) would be done based on developer needs(goals)

Note: Developer while using project management build tool, he would focus only on "Business Logic/Development".



Working with Maven(Only for Java Development)
 a. CLI
 b. IDE's

CLI : Command Line Interaction
 To work with CLI in Maven
  Set up Java in our system (already done)
  Set up Maven in our system

Download maven software from
   a. https://maven.apache.org/download.cgi apache-maven-3.9.7-bin.zip
   b. Extract the file and keep the apache-maven-3.X version in C drive of your local system
   c. Set up environment variable to inform O.S about maven
        MAVEN_HOME = C:\apache-maven-3.9.1
        PATH   = C:\apache-maven-3.9.1\bin | Set using GUI mode of windows os
   d. run the command from commandline

```
C:\Users\nitin>mvn -version
Apache Maven 3.9.1 (2e178502fcdbffc201671fb2537d0cb4b4cc58f8)
Maven home: C:\apache-maven-3.9.1
Java version: 18.0.1.1, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-18.0.1.1
Default locale: en_IN, platform encoding: UTF-8
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"
```

Creating Standalone project using Maven in CLI Mode

Maven Terminologies(GAV)
  a. groupId ⇨ companyName
  b. artifactId ⇨ ProjectName
  c. version ⇨ version of development
  d. packaging ⇨ jar/war

Every dependency used in project is identified as "GAV"

Note: Developer should inform maven about what type of project we need to build.
      To say this to maven we need to for "archetype"[2500+]
        1. standalone :: maven-archetype-quickstart
        2. webapp   :: maven-archetype-webapp

Creating a maven project in CLI mode

(standalone)

```
D:\maventools>mvn archetype:generate -DgroupId=pwskills -DartifactId=01-Maven-App -DarchetypeArtifactId=maven-archetype-quickstart -Dpackage=in.pwskills.nitin.main -Dversion=1.4 -DinteractiveMode=false
```

```
01-Maven-App
   |-> src/main/java
   |       |-> in.pwskills.nitin.main | Development code
   |                 |-> App.java
   |-> src/test/java
   |       |-> in.pwskills.nitin.main | Testing code
   |                 |-> AppTest.java
   |-> pom.xml
         (build file)
```

Life Cycle of Maven

Maven has 3 Life Cycle.Each Life Cycle contains number of Phases

- Clean
- default
- site

(ready made/standard plugins)

```
  clean :: old .class files it will clean (if generated previously)
compile :: Generates the .class file for the source code.
package :: Zip the .class files and creates a jar/war file
   test :: run the test cases and generates the report(testing)
install :: our project will be used as a jar/war in other projects
           (installed locally as a dependency)
```

To run java application using maven standard plugin support not available we use extra plugin in pom.xml file to get the support.
exec-maven-plugin

```
02-Maven-App
   |-> src/main/java
   |       |-> in.pwskills.nitin.main | Development code
   |                 |-> App.java
   |-> src/test/java
   |       |-> in.pwskills.nitin.main | Testing code
   |                 |-> AppTest.java
   |-> pom.xml
         (build file)
```

>> mvn test(generates the test cases output in console)
>> mvn site(generates the test cases output in target/site/index.html)
>> mvn surefire-report:report(generates the test cases output in target/site/surefire-report.html)

```
03-JDBC-Maven-App
   |-> src/main/java
   |       |-> in.pwskills.nitin.main | Development code
   |                 |-> MainApp.java
   |-> src/test/java
   |       |-> in.pwskills.nitin.main | Testing code
   |                 |-> AppTest.java
   |-> pom.xml
         (build file)
```

downloaded and kept in .m2 folder

```
<dependency>
    <groupId>com.oracle.database.jdbc</groupId>
    <artifactId>ojdbc6</artifactId>
    <version>11.2.0.4</version>
</dependency>

<!--Dependency from another project-->
<dependency>
    <groupId>pwskills</groupId>
    <artifactId>02-Maven-App</artifactId>
    <version>1.4</version>
</dependency>
```

use as a service

```
02-Maven-App
   |-> src/main/java
   |       |-> in.pwskills.nitin.main
   |                 |-> Arithmetic.java | Development code
   |-> src/test/java
   |       |-> in.pwskills.nitin.main | Testing code
   |                 |-> AppTest.java
   |-> pom.xml
         (build file)
```

mvn install

mvn package

```
CONNECTION object created...
STATEMENT object created...
RESULTSET object created...
SID     SNAME   SAGE    SADDRESS
10      schin   49      MI
7       dhoni   41      CSK
18      kohli   35      RCB
45      rohith  38      MI
1       rahul   31      LSG
Getting extra service
The sum is :: 30
Closing the resources...
```

Service used from another project

Creating a webapplication using maven in CLI Mode

```
D:\maventools>mvn archetype:generate -DgroupId=pwskills -DartifactId=04-Maven-webApp -DarchetypeArtifactId=maven-archetype-webapp -Dpackage=in.pwskills.nitin.main -Dversion=1.0 -DinteractiveMode=false -Dpackaging=war
```

package not created

```
04-Maven-webapp
   |-> src/main
   |       |-> java
   |       |       |-> in.pwskills.nitin.controller
   |       |                 |-> WishServlet.java
   |       |                       (/wish)
   |       |-> resources
   |       |-> webapps
   |               |-> WEB-INF
   |               |-> index.jsp
   |-> pom.xml
         (build file)
```

mvn package   :: creates a war, deploy it manually in tomcat9 folder
mvn tomcat7:run :: creates a tomcat environment and runs our war file(logical binding)

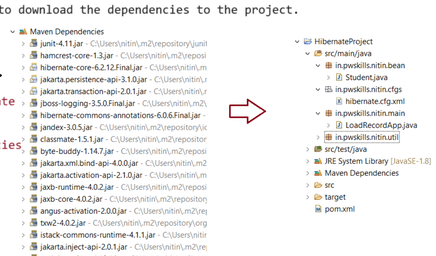
Working with Maven in Eclipse

Note: no need to install any plugin for maven, by default eclipse provides maven supoprt

maven will follow transitive dependency management to download the dependencies to the project.

```
<dependency>
    <groupId>org.hibernate.orm</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>6.2.12.Final</version>
</dependency>
```

hibernate + other dependencies



developer ⇨ version mismatch problem (which version will match if we work with multiple dependencies)

solution

1. Maven inheritance
2. Maven MultiModule

SpringBoot