**4. Maven**

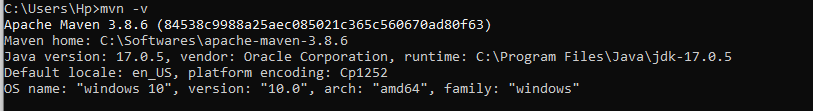
1. Why do we need a build tool like maven?

When a developer implements an application, there are many things to do. He has to write the source code, compile the code, test the code and deploy. And also, if that code needs some dependencies, developer also has to download them and add them into the project. Let`s say the developer manually added those dependencies as jar files and after some time we want to update the versions of that jar files. So, then the developer has to download the newer versions of that jar files and replace them with the existing jars. So, what will happen next? Those versions can be incompatible with each other. Then there will be many errors displayed in the console and the developer has to resolve them one by one.

So, if that developer used a build tool like maven, he wouldn`t need to do those things manually and the build tool will automate those things and then the developer can focus on implementing the business logic without concerning about other things.

Maven is mainly used to build java-based projects. Maven helps us to compile our code, generate documentations, package compiled code into jar files and deploy our application. Using maven, we can get the right jar file for our project and we can simply visit ‘mvnrepository’ and find all the libraries from there. And also, maven helps us to create the right project structure.

1. Display output of maven version



1. What is the pom.xml file?

Project Object Model (POM) file is an XML file which located inside the home directory of our project. When we execute a task, maven, searches for a pom.xml file. Pom file responsible for keeping details of the project such as versioning and configuration management. We can add the properties, plugins and dependencies as xml tags inside the pom file regarding our project.

1. Explain these tags found in pom.xml files?

|  |  |
| --- | --- |
| groupId | Id of the project group. It`s unique amongst an organization. The group Id format is ‘com.companyname.projectgroup’. |
| artifactId | Id of the project (name of the project) |
| version | The version of the project. |
| packaging | Packaging type : it specifies the type of the package that the project needs to be converted. Ex :- jar, war |
| dependencies | Inside the dependencies tag we list all the dependency details which were needed for our project. |
| dependency | Use to add the dependency details. Mainly add group id, artifact id and version of the external library we wanted for our project inside the dependency tags. |
| properties | We can add value placeholders inside the properties tag and we can access them using ${x} notation inside our pom file |

1. Run unit tests with maven. What is the command you used? **mvn test**
2. Explain the usage of these commands

|  |  |
| --- | --- |
| mvn clean | Used to clean the project by removing the generated files from previous build. It deletes the target directory from project folder. |
| mvn install | Used to install the package into the local repository, then we can use that package as a dependency in other projects locally. |
| mvn package | Used to do the packaging for the compiled code into the given format such as jar, war etc. |

1. Explain 3 types of maven repositories

* Local repository

Maven local repository is located in our local computer. If we run mvn command, it will search dependencies inside the local repository.

* Central repository

Central repository located in the web. If our project pom file has a dependency which cannot find inside the local repository, then maven will find it inside the central repository.

* Remote repository

Remote repository is contained the private artifacts and that can be accessible only inside the organization. We can configure the location for the remote repository in our pom file and if our project needed a dependency which cannot find inside our local repository or central repository maven will find it inside the given remote repository locations and will download them.