**MAVEN**

Build is two step process --> 1st compilation, 2nd assembling

compilation : raw source code -> object files

assembly : creating jar/war/ear files from object files

converting .java files into .class files in java we call it as compilation

then we need to assemble all object files so that we have final product(output)

for java assembly file’s extension would be (jar,war,ear)

jar --> (java archive)

war --> creating for web application (web archive )

ear --> to provide license for web application (enterprise archive)

build management is a process of compiling and assembling the code

**ADVANTAGES:**

repitative tasks can be eliminated

we can track history

old tool of java was ANT (open source)

maven is java based tool

maven takes care of entire project --> it devides project into several objects so that functionality would be easy

**maven follows set of standard:**

it provides lifecycle

maven takes care of dependencies --> java code sometimes need other dependencies which can be handled by maven

maven called as project management tool

maven is not just a build tool it is super set of various features like dependencies management tool

every project have build confg files which contains info about what to do ,how to do and all complete details of that project

that confg file we call it as "pom.xml"

maven needs some plugin to perform some actions like compilation ,creating delivery or deleting files etc.

in maven executable is mvn just like in git we have git

instruction we passing in maven we call is goal

so we are giving goal which tell maven what maven has to do

mvn -> goals ->plugins -> task (jar) jar files are nothing but maven plugins

if particular plugin is not present locally then it downloads plugin from apache maven remote repository

**MAVEN build follows a lifecycle:**

**default lifecycle of maven:**

- generate-source/generate-resource (list )

- compile

- test

- package (assembling(creating deliverable war files ))

- install (copying deliverable from other project) --> suppose you have two projects (A and B)--> for both project there are two different delirable files like for A it would be A.jar and for B it would be B.jar --> suppose for creating B.jar file the project might need A.jar file then what maven does it search for A.jar locally but its not available then maven goes to remote repo but again its not available there also --> so A.jar is organization based or project based file its not available locally as well as remote --> so to use A.jar for craeting B.jar maven uses install

- deploy

# there also clean and site not part of lifecycle

clean delete runtime files

maven follows derectory layout

workspace

===========

src/main/ --> under this path there should be all java files present

/java --> all java files present

/resources --> all depencies or library present

/confg -->

src/test/ --> all test files present here which tests all java files present in src/main

ex.

src/main

/java/1.java

/resources

src/test

/java/1\_test.java -->which tests 1.java file in /src/main/java folder

/resources

in the same way entire project will be designed

for java project you will see /src/main

pom.xml