Page No.
Date

MAVEN

Project Management Tool (Java)
Build Tool
Manage Dependancies

Java project structure

→ source code
→ Test code
→ project structure (assets, directories, Resources)
→ Dependancies / Library
→ configuration
→ Task Runner → build / Test / Run
→ Reporting

```
└xyz
  └abc
    └.java
```

MAVEN → Maven is an Automation and project Management tool developed by Apache software foundation. It is based on pom (project object Model).

Maven can build any no. of projects into desired output such as .jar, .war, metadata

Mostly used for Java based projects.

It was initially released on 13th july 2004.

Maven is written in Java

Meaning of Maven is " Accumulator of knowledge "

Maven helps in getting the right ~~jar~~ file for each project as these may be different version of separate packages.

To doconload dependancies it is no more needed to visit the official website ob each software. It could now be easily done by visiting "mvnrepository.com"

Dependencies → It refers to the Jar libraries that are needed for the project

Repositeries → Refers to the directories ob packaged Jar files.

Build tools

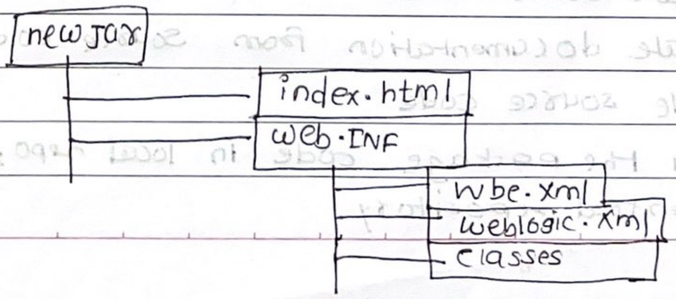
C, C++	Make file
.Net :	visual studio
Java :	Ant, Maven gradle.

Problems without Maven

1. Adding sets ob jars in each project →
In case ob structs, spring, we need to add jar files in each project it must include all the dependancies ob jars also

2. creating the right project structure →
we ~~want~~ must create the right project structure in servlet, struct etc, Otherwise it will not be executed.

For ex: • war file layout



POM (project Object Model)

POM refers to the XML files that have all the informⁿ regarding project and configuration details.

Main configuration file is pom.XML

It has the description of the project details regarding the versioning and configuration management of the project

The xml file is in the project home directory.

POM.xml contains

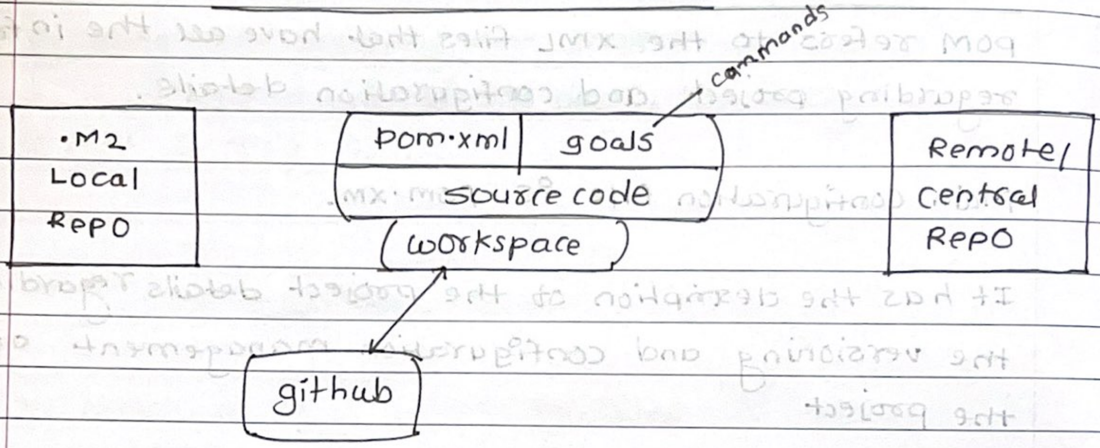
→ Metadata
→ Dependancies
→ Kind of project
→ Kind of output (.jar, .war)
→ Description

one project → one workspace → one pom.xml

Requirnment to Build

- source code (present in workspace)
- compiler (Remote repo → local repo → workspace)
- Dependancies (Remote repo → local repo → workspace)

Architecture of Maven



Maven Build Life-cycle

Goals :-

1) Genarate Resources (Dependancies)
2) compile code
3) Unit Test
4) package (Build)
5) Install (into local repo & artifactory)
6) Deploy (to server)
7) Clean (delete all run time files)

eg: mvn install

mvn clean package

1 to 6 → Default and sequence order

7 → Not default and It won't allow sequence.

- Build lifecycle consist of a sequence of build phases and each build phase consist of a sequence of goals.
- Each goal is responsible for a particular task.

- When a phase is run all the goals related to that phase and its plugins are also compiled.

  eg: mvn install

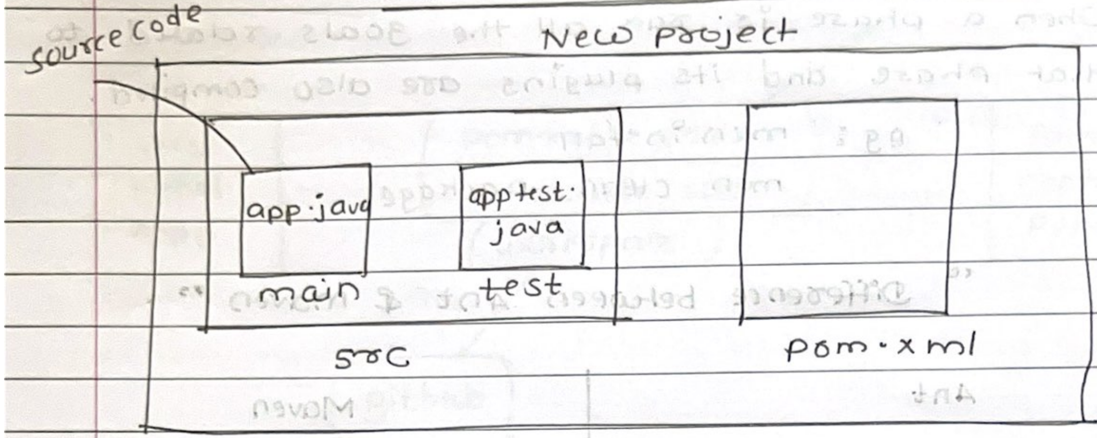
  mvn clean package.

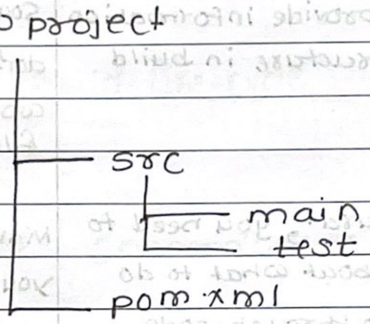
"Difference between Ant & Maven"

Ant	Maven
Ant does not has ~~formal~~ convention so we need to provide information of the project structure in build xml file	Maven has a convention to place Source code, compiled code etc. so we don't need to provide information about the project structure in pom.xml file
Ant is procedural, you need to provide info about what to do and when to do through code.	Maven is a declarative, everything you define in the pom.xml file
There is no Lifecycle in Ant	There is a lifecycle in maven
It is a tool box	It is a framework
It's mainly a build tool	It is mainly a project Management tool
It is less prefered than Maven	It is more preffered.

Page No.
Date

Maven Directory Structure


source code
New project
app·java
main
app test·java
test
src
pom·xml


new project
src
main
test
pom.xml