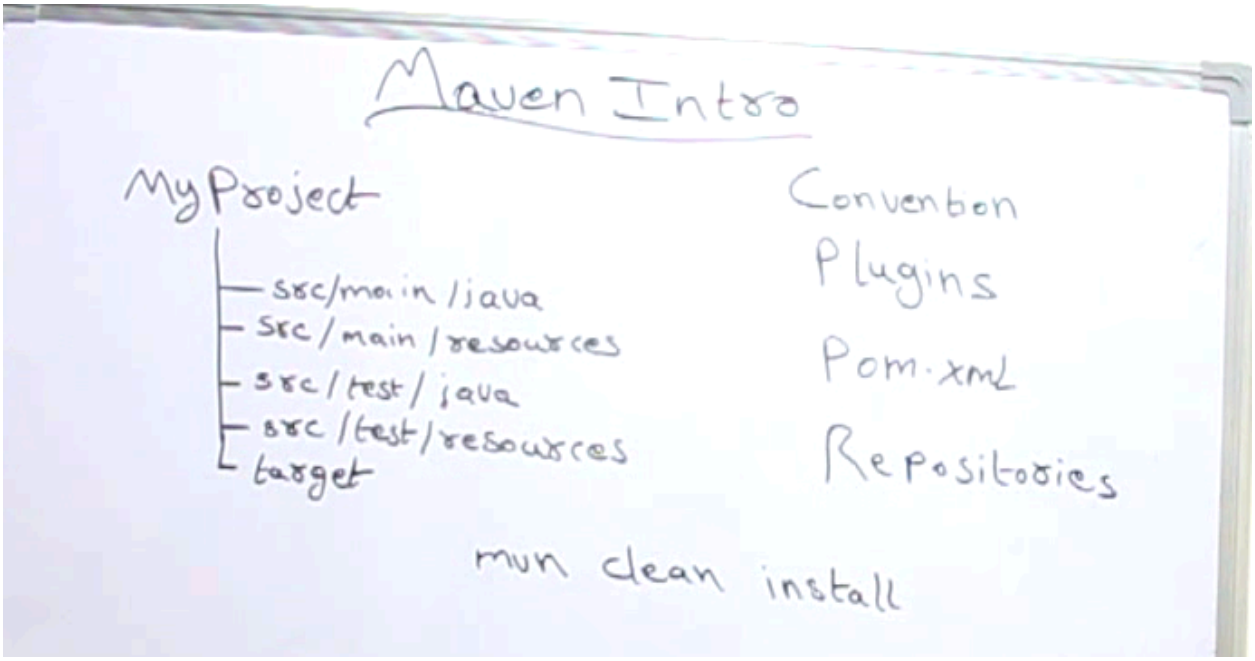
Maven

Maven v1 Maven Introduction (Bharath)

- ✓ Maven is a build automation tool – more than that it's a project management tool
- ✓ Maven focuses on "**Convention over Configuration**" – Build tools like ant uses lots of xml to perform build automation where if you use maven and you follow convention – it could easily perform build automation.
- ✓ Build automation could be build classes - create jar or war files – deploying them to the server – running your test



- ✓ Maven follows convention over configuration – if we build a maven project we need not do many configuration – we need to follow certain set of convention and it will perform all the build automation for us like creating a jar or war, deploying it to the server, perform junit tests
- ✓ POM stands for Project Object Model

Use this for Interview answer about maven:
https://www.youtube.com/watch?v=m_2rUF6GWj8

Maven v2 Maven Lifecycle phases and Goals (Bharath)

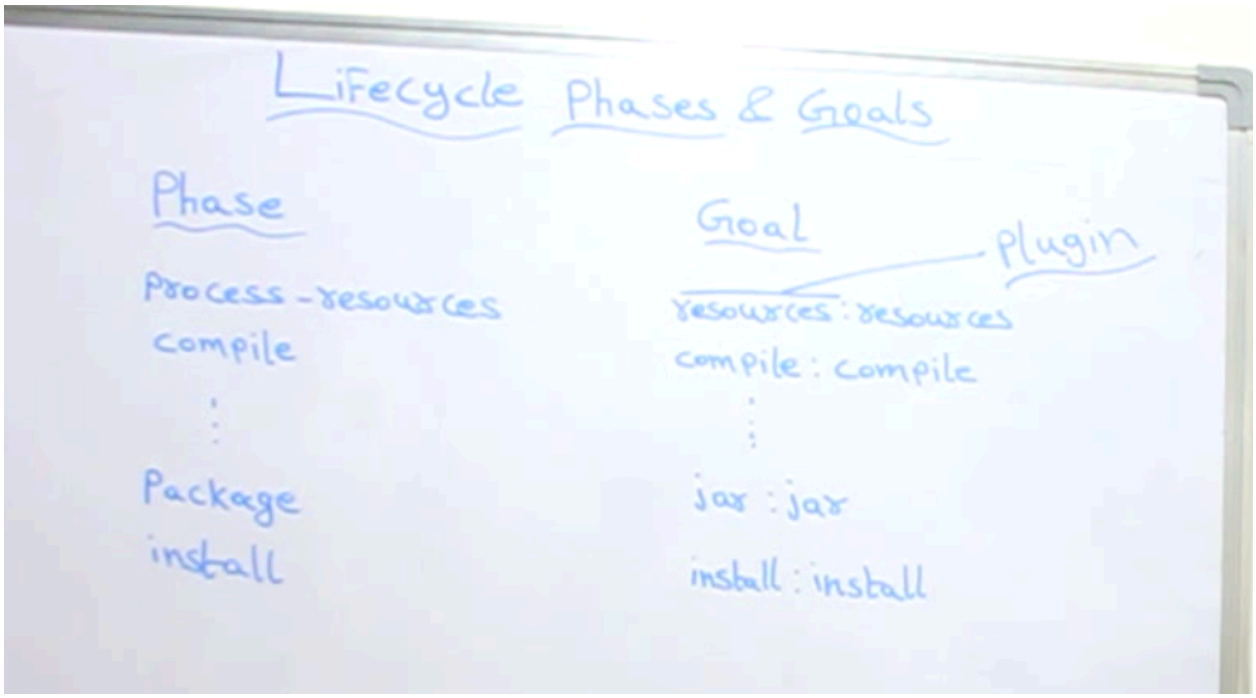
Lifecycle phases of maven:

- Process-resources
- Compile

- Package
- Install

- When we tell maven clean install – we tell maven to execute all the lifecycle phases before install phase

- We can use maven commands using goals also (goals are like tasks in the project) [Ex: maven jar: jar] (it's attached to the package lifecycle phase i.e. equivalent to giving command as maven package)



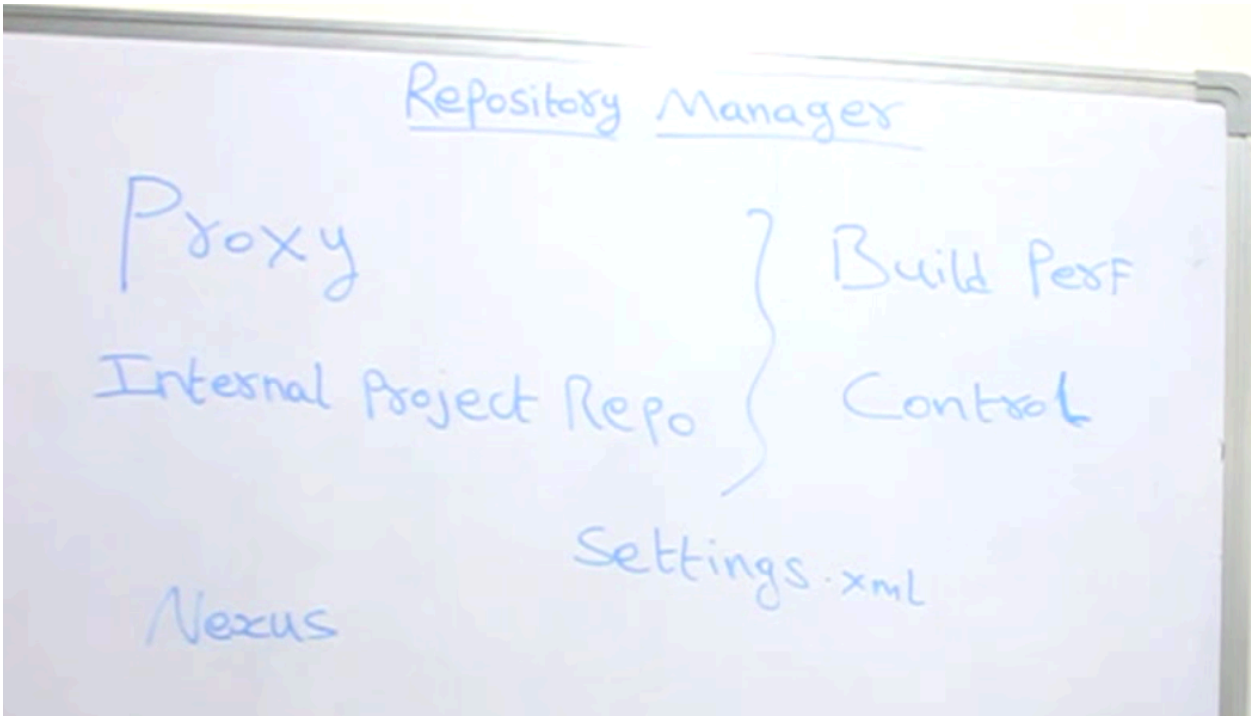
- **Maven will execute all the phases before the phase you want to execute**

- **Maven goals are bound to phases – you can either execute a phase or a goal – both command will work in the same way**

Maven v3 Maven Repository Manager (Bharath)

- Every time maven install or maven build command is given it will download all the jars from a public repository and dump into a local .m2 folder – whenever there is a change the jar in public repository this local repository will sync up with it.

- Nexus – a famous maven repository from sonatype. Uses jetty to configure – once nexus gets configured it can be given in the settings files so that all in company build jar files gets downloaded from the specified nexus repository [gives a lot of control and controls what the organization's developers should use or not]



proxy repository -> improves build performance
Internal Project Repository -> Gives control

==========

Both of them can be mentioned in settings.xml

Gontu Series

VI. Maven Tutorials 01 - Introduction To Maven

What is Maven and what it does?

Maven is a BUILD AUTOMATION tool (also called as a software project management tool)

Compilation of Java Source code

Packaging of code into Jars or Wars

Running Test Cases

Generating Project Documentation (also called as a project website)

Generating Project Reports

What makes Maven so popular around the world?

The ideology behind maven

Maven says, if all projects follow same conventions

like similar directory layout, similar way of compilation of Java Source code, similar way of packaging of code into Jars or Wars, similar way of running test cases etc etc... *then there won't be such a problem -*

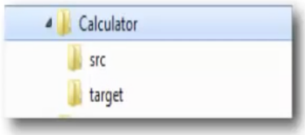
with this approach, everytime a developer moves from one project to another he would not really feel as a newcomer.

The maven project structure

Maven says to all developers to follow below conventions in all Maven based projects:
(Don't worry: I will explain everything which is mentioned below in detail with hands on sessions later)

1. Each Maven based project should have two sub-directories: src and target

Let's say, You are developing a "Calculator" project and the root directory of the project is "Calculator" then according to this convention: "Calculator" directory should have two sub-directories in it with the name src and target
(as shown in this screenshot)



2. src directory should have again two sub-directories in it: main and test

3. main directory should contain a java sub-directory which will contain all Java Source files

4. main directory should contain a resources sub-directory which will contain all resources like an image file, a properties file etc which are needed at runtime by any of the Java Source files.

There are many other conventions which Maven suggest: like how to run test cases, how to compile code, how to package code into Jars or Wars, which sub-directory should contain test classes etc etc which I would talk about in detail with hands on sessions later...

Maven dependency management

Maven has an excellent capability to automate the task of handling all needed project dependencies... any Jar (or a set of Jars or Libraries) which a project needs is called as a Project dependency
e.g if you want to use spring framework in your project then all Jars which you need in order to use spring framework are called as dependencies of that project.

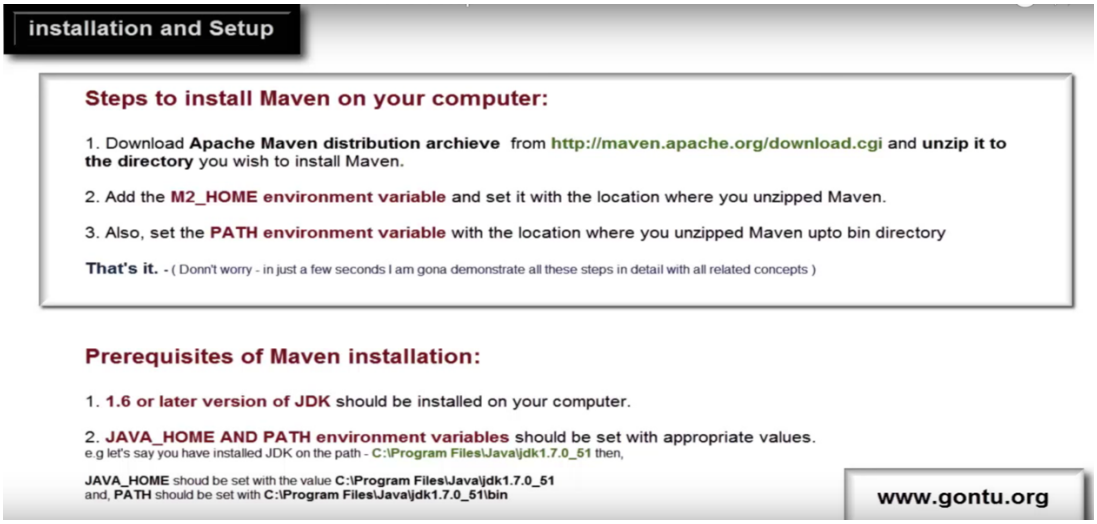
in this example:

All spring Framework related Jars which are required in the project are called as direct dependencies of the project.

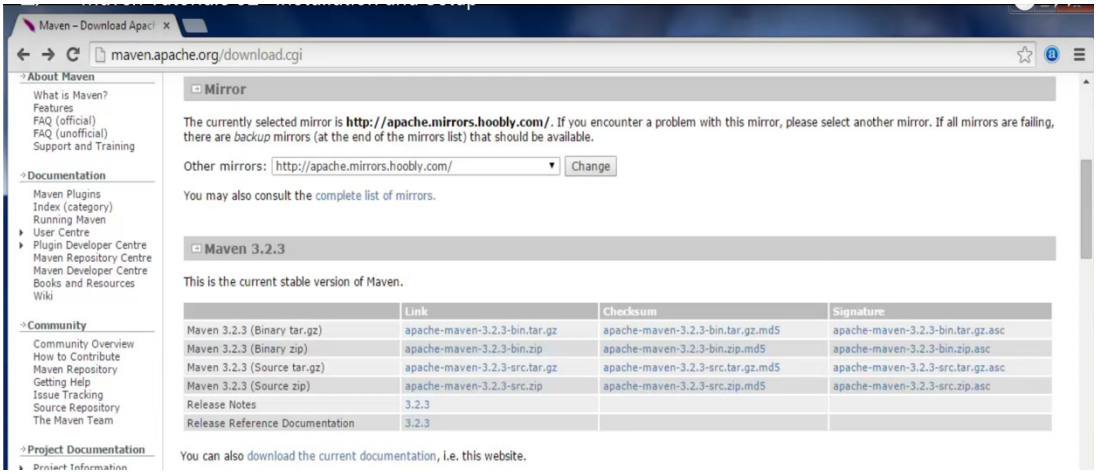
if any of the Spring Framework related Jar further depends on some other Jar or Jars then all those jars are also required in the project in order to use spring Framework and are called as indirect dependencies of the project.

V2. Maven Tutorials 02 – Installation and Setup

Setting up maven in the system – pre Requisites



Maven download



V3. Maven Tutorials 03 – How to use Maven 01 (Archetypes, GroupId, ArtifactId, Version, pom.xml)

Commands:

`mvn –version`

- Tells the version of the maven and other stuffs. Use it to know whether

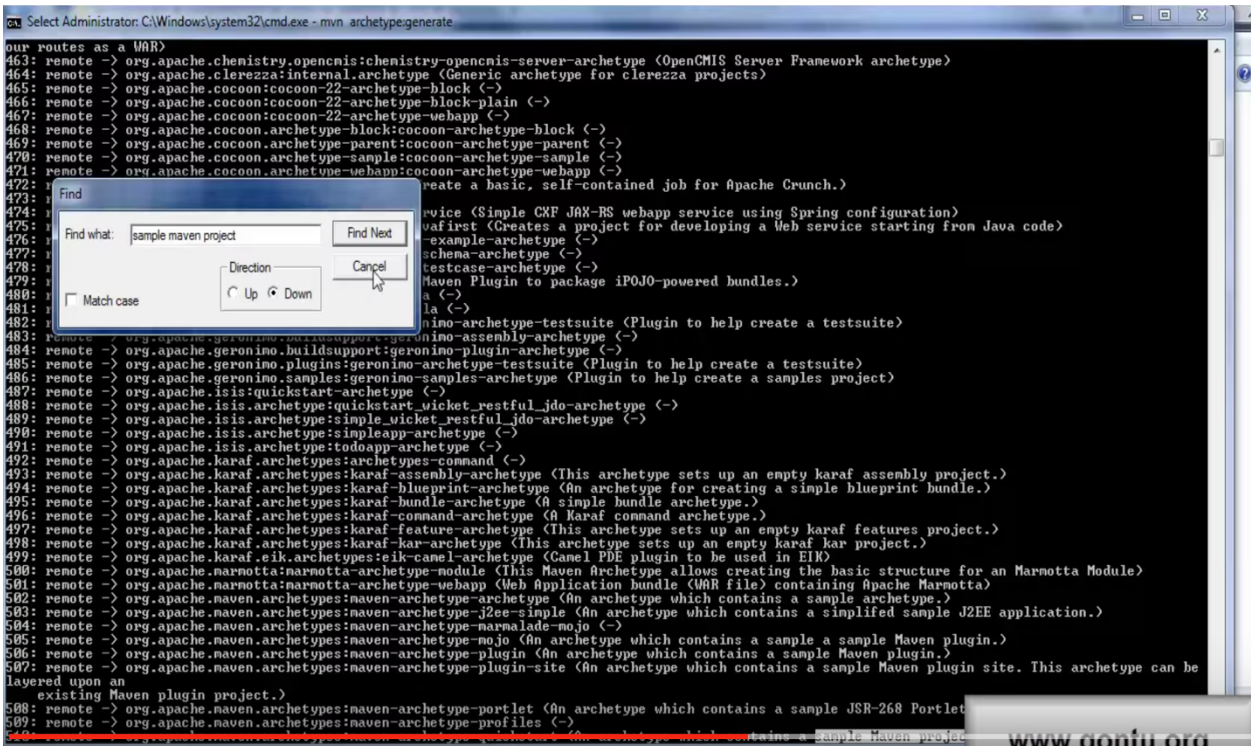
`mvn archetype:generate`

- It asks for information like project structure, type of project and other stuffs. Based upon our input this will create the project structure and all.

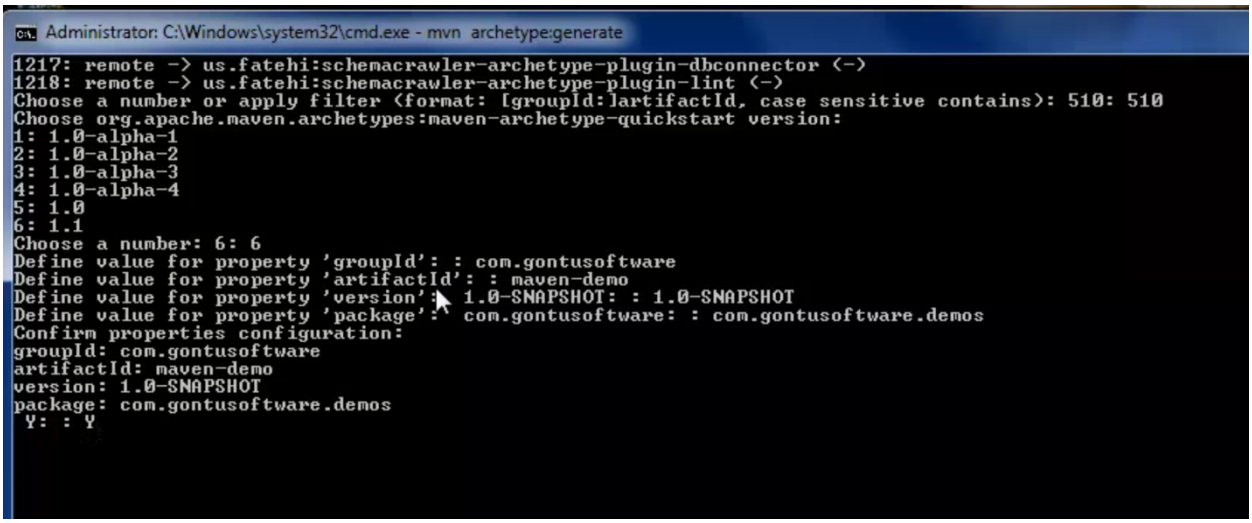
Once this command is given – it download the plugins. The fact is before running any command in maven we need to download specific set of plugins for that command

Maven downloads the plugins into .m2 folder to later check for locally

architype – is the type of the project along with the option no displayed by mvn command



providing groupId, ArtifactId, version and package details



SNAPSHOT – means the project is still under development
groupId – opposite of domain name [whoever is creating the project]
Artifact id – project name

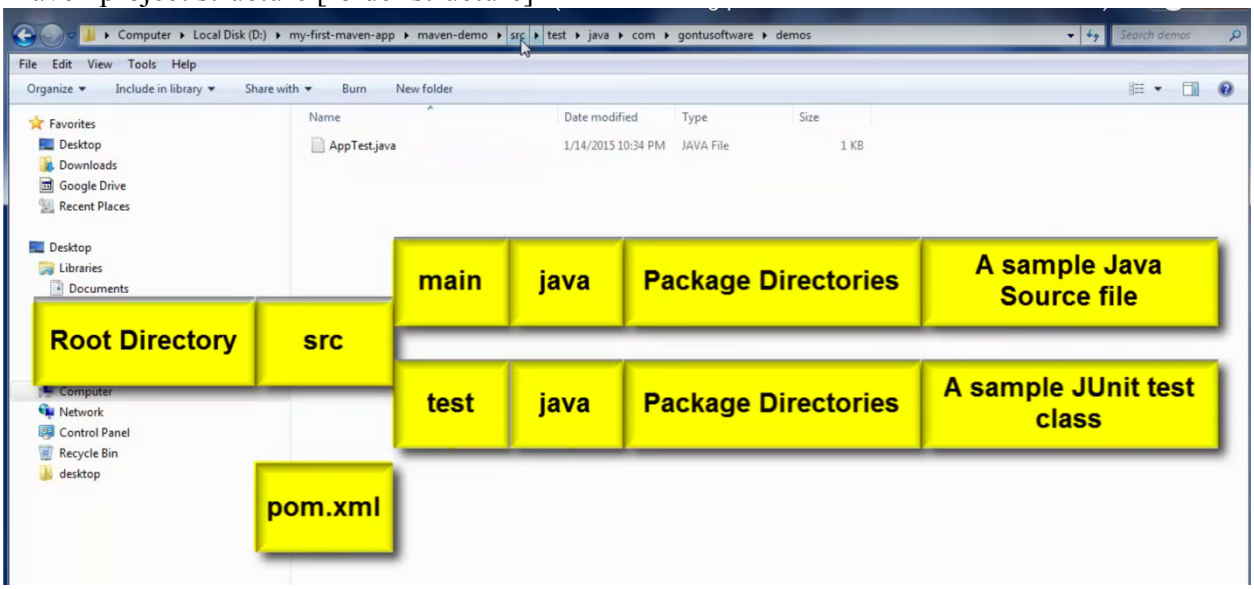
Providing groupId

Providing ArtifactId and version no

Jar will be created in this format [artifactId-version.extension]

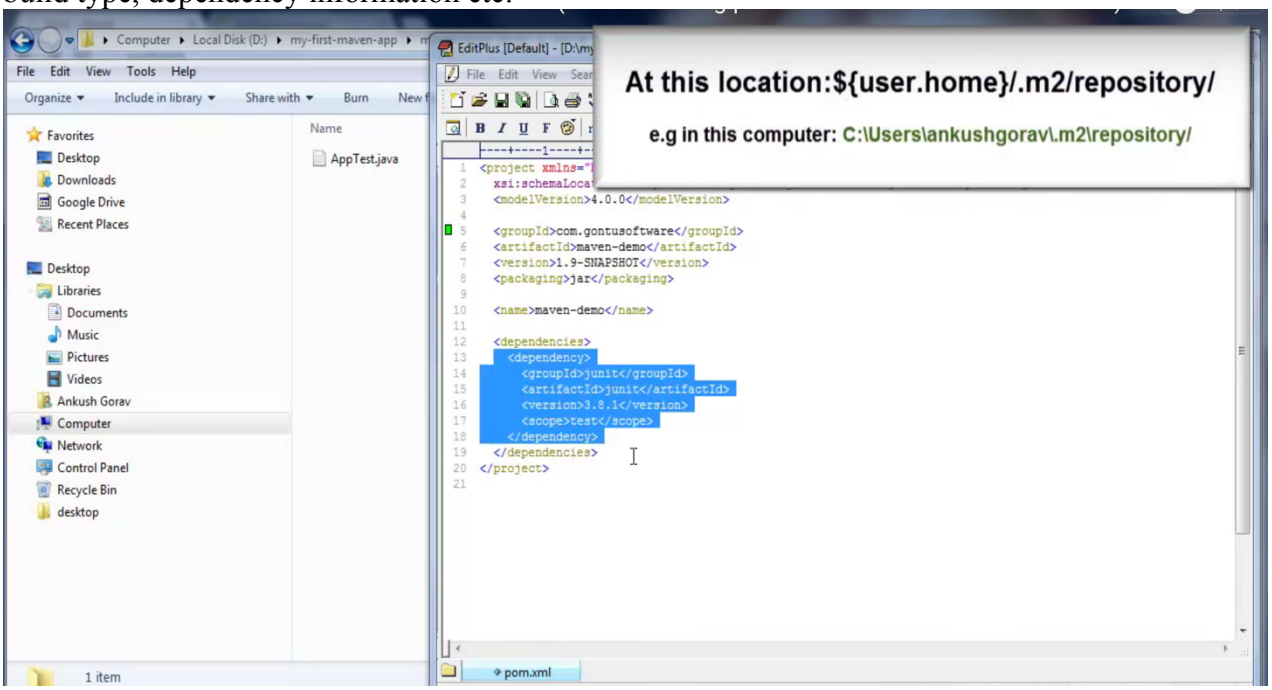
V4. Maven Tutorials 04 – How to use Maven 02 (understanding pom.xml in detail and some more...)

Maven project structure [folder structure]



pom.xml

this file holds all the configuration information about the project like project name, version no, build type, dependency information etc.

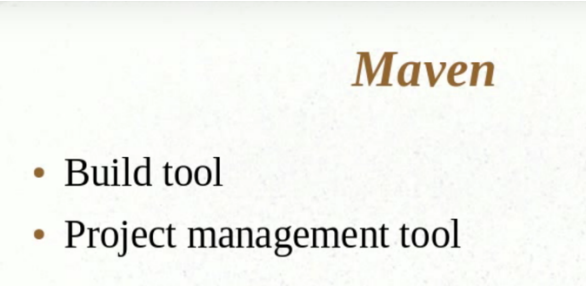


a sample pom.xml file with groupId, ArtifactId, version, packaging info and dependency information

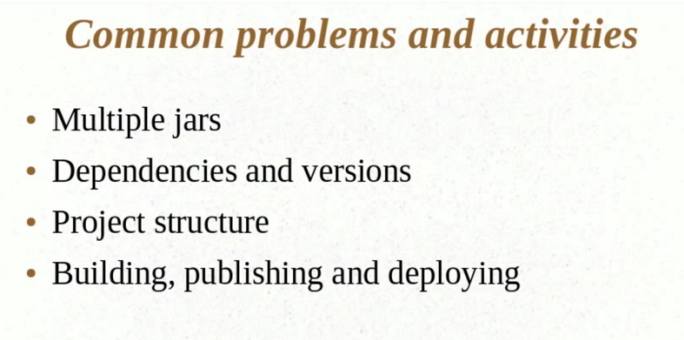
Java Brains

V1. Maven Tutorials 01 - Introduction and Setting Up

What is maven?



Some current problems with the other build systems:



Archetype – it's a model how your project structure will get set - maven comes with various predefined archetypes

Pom.xml – has information about application itself and what are the dependencies

V2. Maven Tutorials 01 Part 2 - Introduction and Setting Up

Commands:

`mvn compile`

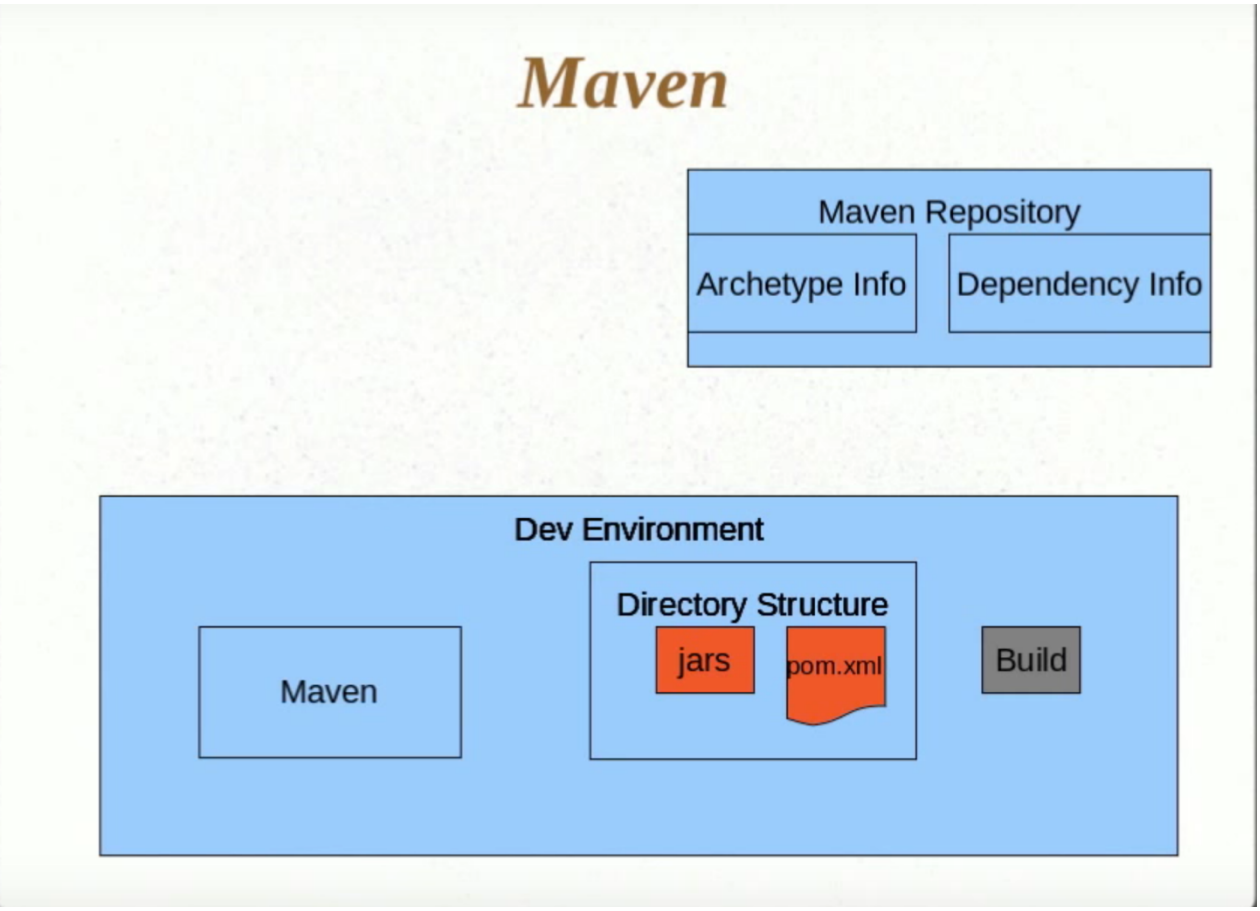
- It will download all the related dependencies and compile java classes

`mvn package`

- This will package our projects in a jar file.

In all the cases, the plugins will get downloaded in the first instance when we will run the command

How maven works:



Archetype Info – contains the project structure info
Dependency Info – contains the info about various dependencies and versions (jar info)

- When we give *mvn archetype:generate* command it goes to maven repository and gets corresponding archetype info and creates the project structure in out dev environment.
- Once the directory structure gets created we specify the dependencies in the pom.xml (by default junit dependency will be there).
- Then when we give mvn compile or mvn package command it downloads all the dependencies from the maven repository.
- And builds the jar using those.

V3. Maven Tutorials 02 – Understanding Archetypes and pom.xml

What maven does for us - Provides

Maven

- Project template
- Build

Maven archetype is responsible for creating

Maven archetype

- Folder Structure
- pom.xml

archetype:generate contains

archetype:generate

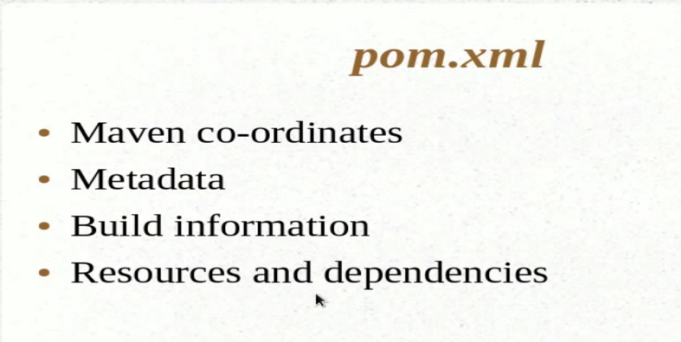
- Archetype
- Group ID
- Artifact ID
- Version
- Package

A sample pom.xml file

pom.xml

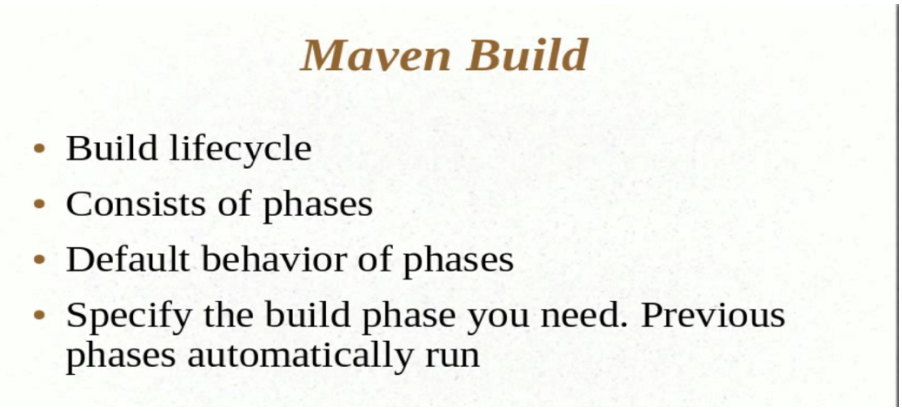
```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.santanusetu</groupId>
  <artifactId>MavenTestApp</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>MavenTestApp</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

In short pom.xml contains information about following things:

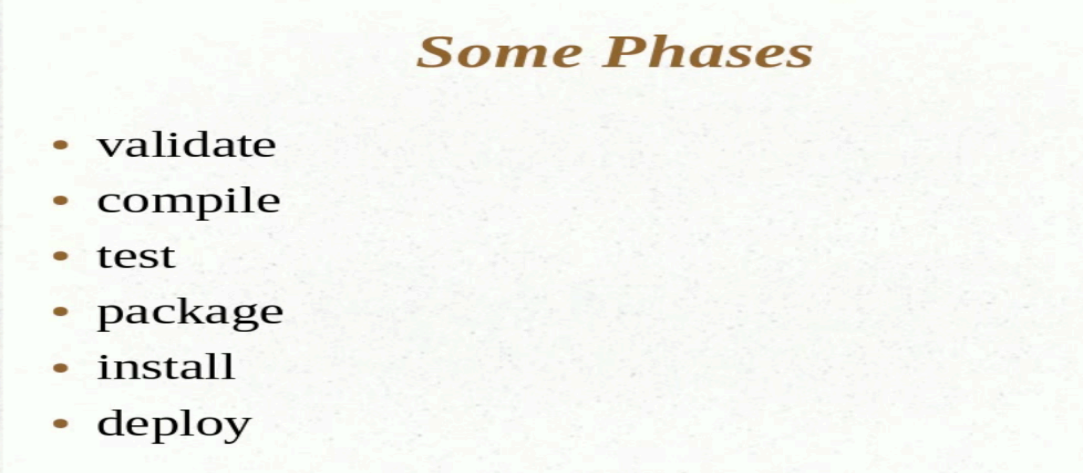


V4. Maven Tutorials 03 – Maven Build Phases

Features of Maven Build



Some maven build phases



compile – compiles the class files
test – runs the junit test cases

package – packages the jar or war depending upon the project type

install – installs the packages in the local .m2 repository [if some other project defines it as its dependency – it will first search in local repository – then go to central maven repository]

deploy – deploys the package in local repository

```
mvn compile
mvn test
mvn package
mvn install
```

- Any phase command given will execute all the previous commands and then it will execute itself

V5. Maven Tutorials 04 – Adding a Dependency

```
mvn clean – clean is a phase that removes all the classes and distributions generated before
```

To search for a dependency in maven repository – type search maven repository in Google and type the dependency name in the search bar. Click on the version no – it will give the complete text that we need to add in the pom.xml

```
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>1.7.21</version>
</dependency>
```

mvn compile

Default scope value is compile – that means that jar will be available for only compile stage

V6. Maven Tutorials 05 – A Web Application using Maven

We can take a sample j2ee project and deploy in tomcat – it will run as normal

V7. Maven Tutorial 06 - Introduction to Plugins with the Maven Compiler Plugin

```
mvn clean install
```

- Cleans and then installs the package in local repository

V8. Maven Tutorial 07 – Using the jetty plugin

https://www.youtube.com/watch?v=6sIhJuaZhF0

V9. Maven Tutorial 08 - Eclipse Plugin for Maven and Maven Plugin for Eclipse

https://www.youtube.com/watch?v=xE2F4Z_wKCU

Maven Interview Questions

1.What is Maven?
In simple terms, Maven is a build automation tool or Project Management Tool.

2.How Maven is different from other build tools?
Maven makes building a project very simple. The complex part of the build process is gathering all the required jar files which maven takes care automatically through a concept called dependency management.

Also Maven comes up with pre-defined packaging mechanisms for Jar, War, Ear which makes the build process really simple.

3.What is a POM file?
Maven needs a configuration file called POM (Project Object Model) to know the information about the project to be build.

Maven proposes standard way of storing the source code, test code and configuration files. If the project is structured this way, then lot of things are automatically taken care by Maven.

4.What are the basic elements in a POM file?
POM file is a XML file. The basic POM file contains the GroupId, ArtifactId, Version – GAV

5.What is the standard maven project structure ?
-src
--main
---java
--test
---java
--resources

6.What is a Maven Plug-in ?
Each of the life cycle activities in the build process is done through a different components. These components are called plug-ins in Maven terms. For example, compiling is done through a plug-in called compiler plug-in. Creating a war file is done through maven war plug-in.

7.What is dependency management?
Any java code need or depend on additional Jar files apart from what is available in JDK.
Maven makes building a project really easy by automatically downloading these jar files once the required Jar files are declared in the POM file.

8. What are the mostly used Maven commands on a day to day basis.
Mvn install - Compiles the projects, Run the tests if there are any and generates the build artifacts.
Mvn clean - Cleans the project and removes all the build artifacts created.

9.How to integrate Maven with Source control system ?
The SCM tag in the pom.xml file.

Mvn Release:prepare - Prepares to release the project.
Mvn Release:perform - Uploads the artifacts to the repository

10.What are the best practices in Maven setup
1)Use Parent POM with out exception and define all the common things there.

Common things here are
- properties
- dependency management
- SCM
- distribution management
- repositories

2) Don't explicitly mention the version numbers of the dependency jar files in the POM files except in the parent pom.xml files. Mention version numbers in the parent POMs.

In the parent pom.xml file the dependency management tag is used to define all the dependency jar files with version numbers. If this parent pom.xml file is used in all the other pom.xml files in the project then you don't have to change the version number in each and every pom.xml file.

11.What are the different types of scope available in Maven ?
Compile
Test
Provided