Spring Chitkara University Live [SCUL] 📚⚽

BRILLIANT !

Maven : Its a build tool which allows to create Java apps in easier way . Build tool means how you want to package your application . This is like npm for node , maven is for Java .It is used to maintaining dependencies .Build means packaging your application into a jar .

You can check version of maven in your system by opening the cmd/terminal and typing mvn -v

What is **Dependencies** : Its a functionality which you are embedding in your application . Its already created by other developers . Often open source .

By default there is a .m2/ folder which comes with Maven installation . Its a hidden folder inside your user . In windows it will be inside C/Users/<name> . In mac and linux , you will find it inside root folder .**Lets have a look at it .**

Maven has 3 types of repositories : local , central and remote .

What is local repository: Its all the dependencies that your local system have . Dependency is just a piece of code .

Central repository : It has all the dependencies . Its hosted globally . The dependencies we download from here to our system become local repository .

Remote repository : For organisations who cant push their code to WWW on central repository .

Apart from central repository, you may have needed artifacts deployed on other remote locations. For example, in your corporate office there may be projects or modules specific to organization only. In this cases, organization can create remote repository and deploy these private artifacts. This remote repository will be accessible only inside organization.

Spring Chitkara University Live [SCUL] 📚⚽

These maven remote repository work exactly same way as maven's central repository. Whenever an artifact is needed from these repositories, it is first downloaded to developer's local repository and then it is used.

Make sure that whatever java version you have in mvn --version , according to that you are making project in IntelliJ .It should be same .

Lets have a look at IntelliJ now . >Create a new project .Go to Maven section .
What is an archetype ? It is basically the starting point of your application .Its a project structure /layout . So you don't have to write that code again .If we click any of these , it will create a dependency inside our POM.XML .
Currently we dont need any **ARCHTYPE** .
>Click on **Next** . Give name and location . Ok .

You can see that in the background tasks , it is downloading the dependencies from the Central repository and resolving them .

Group Id and Artifact ID in the pom.xml distinguishes your app from other apps . This is indorder to push it globally .

>Go to spring initializer and copy paste the string starter web dependencies to pom.xml by doing explore space there .
If you install parent dependencies , all the child dependencies automatically get installed .

In test folder you write test cases . In src/main/java ,you define your java files . You can define them inside a package or directly inside the java folder . Also remember to add all required things like :

```xml
<dependencies>
  <dependency>
     <groupId>org.springframework.boot</groupId>
     <artifactId>spring-boot-starter-web</artifactId>
     <version>2.5.5</version>
  </dependency>
</dependencies>
```

Finally , reload the maven products to get the jars . For this ,there is a refresh/reload button needs to be clicked .It will download the dependencies from the central repository .
Now in your External Libraries , you will be able to see the MAVEN dependencies .On further exploration , you can see the jars inside it .

Understanding How to build and package your project in a single jar inside target folder :

On the extreme top right , you can see the MAVEN tab . Here you can see the lifecycle and also dependencies .

MAVEN BUILD LIFECYCLE :

- clean
- validate
- compile
- test
- package
- verify
- install
- site
- deploy

Now lets create a simple java class in our java folder and create a main method inside .
Now go to the location of your project using the terminal .
Type mvn clean
Now Type mvn clean package

Explore the new folder target and you can see compiled class files and jar created for deployment in Production . WHAT IS A JAR (its a resource)

Using **mvn clean package** , you can bypass the first five lifecycle methods .

- clean
- validate
- compile

Spring Chitkara University Live [SCUL] 📚⚽

- test
- package

A JAR file is a file with Java classes, associated metadata, and resources such as text and images aggregated into one file.

https://pediaa.com/what-is-the-difference-between-jar-war-and-ear/

Difference b/w mvn clean package and mvn package
-> mvn package will not execute clean phase

Difference b/w mvn clean package and mvn clean install
->mvn clean package checks in local first for pom dependencies . If not present downloads the dependencies defined in the pom from central repository .It does not put the downloaded dependencies into the .m2 folder . The con here is that next time you have to build any other project , you have to again download from the central repository .
Running mvn clean install solves this by following all the steps of the mvn clean package and also putting the downloaded dependencies into the local .m2 folder .This installs the package into the local repository, for use as a dependency in other projects locally

mvn deploy : pushing your changes in pom.xml to remote repo.It's only needed when you have made changes to pom.xml .Running this is going to attempt to deploy the files to a remote repository or server. It copies the final package to the remote repository for sharing with other developers and projects.

Note : All code files , notes and download links have been uploaded to Github .
Please make sure to download Java in your systems , IntelliJ as IDE and Maven in your systems as build automation tool .