*MAVEN:*

* *Whenver development team is working on building a application for a project, it can be a webapplication or enterprize application or any other application,*
* *dev team will write the custom source code and push it to git hub,*
* *dev team uses open source code (open source dependencies) which are already available and posted by the development communities worldwide.*
* *this open source code is then integrated by the developers in there custom code and an appplication is created.*
* *the open source dependencies or also called as open source libraries are available in jar format (java archive) which consists or complied java class files, which are ready to be integrated as it is.*
* *The open source code dependencies are available in a repository also called maven central repository from which we can freely download it. Below is the link*

*https://mvnrepository.com/repos/central*

*----------*

*-------------------*

*Maven is a build tool which helps in:-*

*a. downloading open source dependencies.*

*b. compliling custom source code.*

*c. integrating Open source dependencies with the compilied code.*

*d. packaging the full compiled and integrated code in WAR/EAR format as per need.*

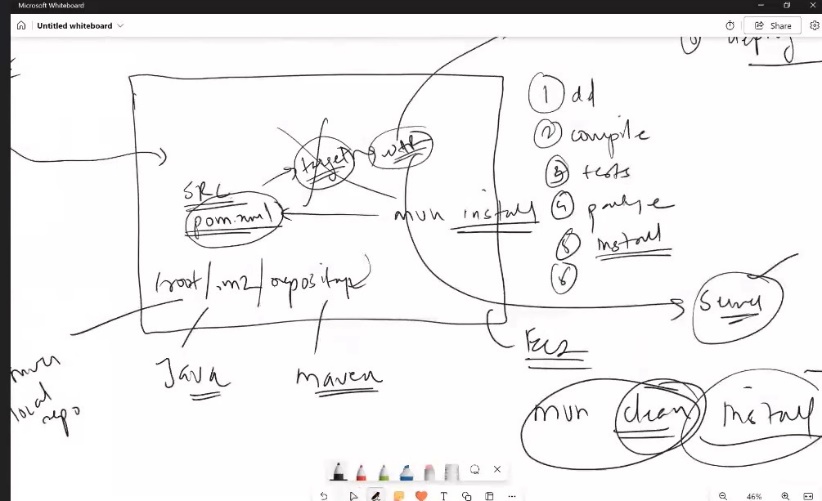
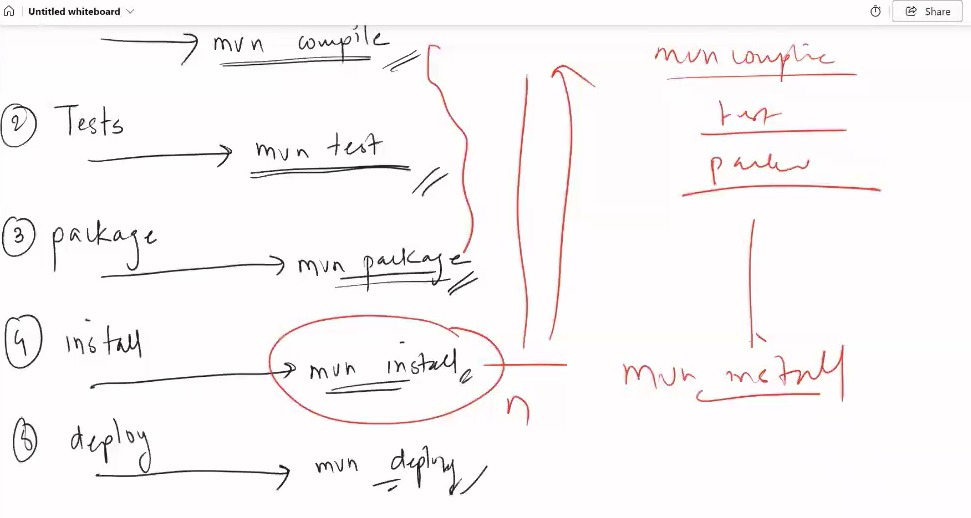
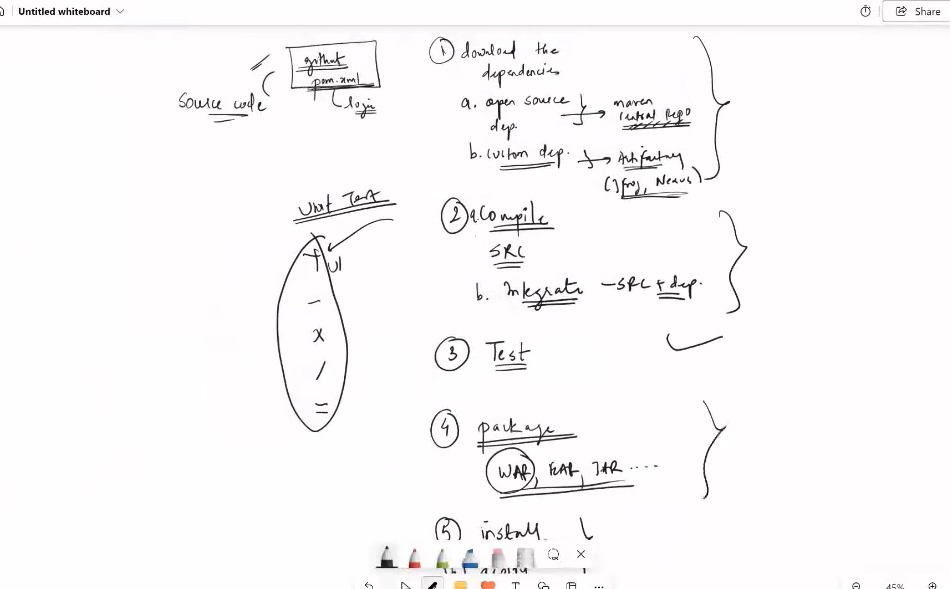
*A diagram of a web page

Description automatically generated with medium confidenceA white board with black text

Description automatically generated*

*WAR --- Web archive which is nothing but used to host web applicationEAR -- enterprize archive which is used to run /host enterprize applications*

*MAVEN-LIFECYCLE*



*1. download the open source dependencies from the maven central repository*

*after checking out the details inside the pom.xml file.These dependencies are downloaded on the machine inside a folder called as maven local repository.*

*2. After the open source dependencies are downloaded from the maven central repo,it will also check for any custom dependencies which are used in your project and it will look for themin the remote repository or the artifactory which is used in your company.*

*Note: Artifactory is same as the central maven repo but it is only used in a organization as it has custom dependencies.*

*3. Once the dependencies are downloaded it will Compile the source code and integrate the source code with the dependencies.*

*Point 1-3 are in the part of COMPILE PHASE Of maven lifecycle.*

***mvn compile***

*4. After the code is compiled the test cases are executed, here these tests are written by the dev team itself to check the code if there are any issues after compilation. These are called as unit test cases. Also, some integration tests are also present.*

*This Phase is the TEST phase of maven lifecycle.*

***mvn tests***

*5. After the tests are done, the compiled code is then packaged into WAR/EAR or JAR as per pom.xml.this phase is PACKAGE PHASE.*

***mvn package***

*6. A copy of the compiled package is stored in the maven local repository.*

*this happens in INSTALL PHASE.*

***mvn install***

7. if we want we can upload a copy of our package to the remote repository or artifactory.

this is called DEPLOY Phase

**mvn deploy**

**What does mvn clean install do?**

**mvn clean install**

**Apache Maven is a popular build tool, that takes your project’s Java source code, compiles it, tests it and converts it into an executable Java program: either a .jar or a .war file.**

**mvn clean install is the command to do just that.**

**You are calling the mvn executable, which means you need Maven installed on your machine. (see How do you install Maven?)**

**You are using the clean command, which will delete all previously compiled Java .class files and resources (like .properties) in your project. Your build will start from a clean slate.**

**Install will then compile, test & package your Java project and even install/copy your built .jar/.war file into your local Maven repository. (see A look at the Maven build lifecycle: phases)**

**Commands:**

**1 clear**

**2 yum install java-11-amazon-corretto.x86\_64 -y**

**3 java -version**

**4 yum install maven -y**

**5 mvn -version**

**6 mvn**

**7 ls -ltr**

**8 pwd**

**9 ls -ltra**

**10 cd .m2**

**11 ls -ltr**

**12 cd repository/**

**13 pwd**

**14 ls -ltr**

**15 cd**

**16 clear**

**17 cd /mnt**

**18 ls -ltr**

**19 yum install git -y**

**20 clear**

**21 git clone https://github.com/Shantanumajan6/project.git**

**22 cd project/**

**23 ls -ltr**

**24 mvn install**

**25 ls -ltr**

**26 cd target**

**27 ls -ltr**

**28 cd /root/.m2/repository/**

**29 ls -ltr**

**30 cd**

**31 cd /mnt/**

**32 ls -ltr**

**33 cd project/**

**34 ls -ltr**

**35 mvn clean**

**36 mvn install**

**37 cd /root/.m2/**

**38 ls -ltr**

**39 rm -rf \***

**40 ls -ltr**

**41 cd**

**42 cd /mnt/project/**

**43 mvn clean install**

<https://github.com/Shantanumajan6/project> ---- project url

**mvn clean**

**The command mvn clean is used to remove the target directory in a Maven project. Here's what it does:**

**mvn: This is the command-line tool for running Maven commands.**

**clean: This goal deletes the target directory, which contains all the build artifacts (compiled classes, JAR files, etc.) from previous builds.**

**Running mvn clean ensures that any previous build artifacts are removed, providing a clean slate for the next build. This can help avoid issues caused by leftover files from previous builds**

**mvn clean install -DskipTests=trueRunning**

**this command will clean any previous build outputs, compile the project, and install the artifacts into the local repository without running the tests. This can be useful when you want to quickly build and install a project without spending time on test execution.**

**mvn install -X**

**-X: This option enables debug output. Maven will provide detailed logging information, which can be very helpful for diagnosing issues during the build process.**

**Running this command will perform a complete build and installation of the project, and it will produce verbose logging output to help troubleshoot any problems that arise during the build.**

**----------------------------------------------------------------------------------------------------------------------------------------**

**[root@ip-172-31-3-109 apache-maven-3.9.6]# ls -ltr**

**total 36**

**-**

**[root@ip-172-31-3-109 apache-maven-3.9.6]#**

**this location is maven home location**

**inside the bin folder we have the mvn command which initially will only run from the bin folder.**

**but what we want is that command should run from anywhere. thus we need to setup an env variable on the machine with root user so**

**go to /root folder open .bash profile file and edit as below, after editing logout and login once.**

**# .bash\_profile**

**# Get the aliases and functions**

**if [ -f ~/.bashrc ]; then**

**. ~/.bashrc**

**fi**

**# User specific environment and startup programs**

**PATH=$PATH:$HOME/bin**

**export MAVEN\_HOME=/mnt/build-tools/apache-maven-3.9.6 ----write this in file**

**export PATH=$MAVEN\_HOME/bin:$PATH ------write this in file**

**export PATH**

**############**

**now mvn command will run from anywhere**

**[root@ip-172-31-3-109 ~]# cd /mnt**

**[root@ip-172-31-3-109 mnt]# mvn**

[INFO] Scanning for projects...

[INFO] ------------------------------------------------------------------------

[INFO] BUILD FAILURE

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 0.228 s

[INFO] Finished at: 2024-05-09T16:28:15Z

[INFO] ------------------------------------------------------------------------

[ERROR] No goals have been specified for this build. You must specify a valid lifecycle phase or a goal in the format <plugin-prefix>:<goal> or <plugin-group-id>:<plugin-artifact-id>[:<plugin-version>]:<goal>. Available lifecycle phases are: pre-clean, clean, post-clean, validate, initialize, generate-sources, process-sources, generate-resources, process-resources, compile, process-classes, generate-test-sources, process-test-sources, generate-test-resources, process-test-resources, test-compile, process-test-classes, test, prepare-package, package, pre-integration-test, integration-test, post-integration-test, verify, install, deploy, pre-site, site, post-site, site-deploy. -> [Help 1]

[ERROR]

[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.

[ERROR] Re-run Maven using the -X switch to enable full debug logging.

[ERROR]

[ERROR] For more information about the errors and possible solutions, please read the following articles:

[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/NoGoalSpecifiedException

[root@ip-172-31-3-109 mnt]#

build failure is coming becase there is no pom.xml file.

[root@ip-172-31-3-109 ~]# ls -ltra

total 28

-rw-r--r-- 1 root root 129 Oct 18 2017 .tcshrc

-rw-r--r-- 1 root root 100 Oct 18 2017 .cshrc

-rw-r--r-- 1 root root 176 Oct 18 2017 .bashrc

-rw-r--r-- 1 root root 18 Oct 18 2017 .bash\_logout

dr-xr-xr-x 18 root root 257 May 9 16:22 ..

drwx------ 2 root root 29 May 9 16:22 .ssh

-rw------- 1 root root 830 May 9 16:27 .viminfo

-rw-r--r-- 1 root root 266 May 9 16:27 .bash\_profile

-rw------- 1 root root 365 May 9 16:27 .bash\_history

drwxr-xr-x 3 root root 24 May 9 16:28 .m2

dr-xr-x--- 4 root root 151 May 9 16:28 .

**[root@ip-172-31-3-109 ~]# cd .m2**

**[root@ip-172-31-3-109 .m2]# ls -ltr**

**total 0**

**drwxr-xr-x 2 root root 6 May 9 16:28 repository**

**[root@ip-172-31-3-109 .m2]# pwd**

**/root/.m2**

**[root@ip-172-31-3-109 .m2]#**

**[root@ip-172-31-3-109 .m2]# cd repository/**

**[root@ip-172-31-3-109 repository]# pwd**

**/root/.m2/repository ---------------------------------------THIS LOCATION IS MAVEN LOCAL REPOSITORY, HERE THE DEPENDENCIES WILL BE DOWNLAODED.**

**[root@ip-172-31-3-109 repository]# ls -ltr**

**total 0**

**[root@ip-172-31-3-109 repository]#**

**Commands:**

**1 clear**

**2 yum install java-1.8.0-openjdk-devel.x86\_64 -y**

**3 java -version**

**4 cd /mnt**

**5 mkdir build-tools**

**6 cd build-tools/**

**7 wget https://dlcdn.apache.org/maven/maven-3/3.9.7/binaries/apache-maven-3.9.7-bin.zip**

**8 ls -ltr**

**9 unzip apache-maven-3.9.7-bin.zip**

**10 ls -ltr**

**11 rm -rf apache-maven-3.9.7-bin.zip**

**12 ls -ltr**

**13 clear**

**14 ls -lrt**

**15 cd apache-maven-3.9.7/**

**16 pwd**

**17 ls -ltr**

**18 cd /bin**

**19 ls -ltr**

**20 pwd**

**21 cd ../**

**22 cd**

**23 clear**

**24 cd /mnt/build-tools/apache-maven-3.9.7/bin**

**25 ls -ltr**

**26 ./mvn**

**27 pwd**

**28 cd**

**29 pwd**

**30 ./mvn**

**31 mvn**

**32 cd /mnt**

**33 mvn**

**34 cd**

**35 pwd**

**36 ls -ltra**

**37 vi .bash\_profile**

**38 mvn**

**39 clear**

**40 cd /mnt**

**41 ls -ltr**

**42 yum install git -y**

**43 git clone https://github.com/Shantanumajan6/project.git**

**44 ls -ltr**

**45 cd project/**

**46 ls -ltr**

**47 mvn clean install**

**48 cd ..**

**49 git clone https://github.com/Shantanumajan6/game-of-life.git**

**50 ls -ltr**

**51 cd game-of-life/**

**52 ls -ltr**

**53 mvn clean install**

**54 ls -ltr**

**55 cd gameoflife-web**

**56 cd target/**

**57 ls -ltr**

**58 pwd**

**59 cd**

**60 cd /mnt**

**61 ls -ltr**

**62 mkdir servers**

**63 cd servers/**

**64 ls -ltr**

**65 wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.89/bin/apache-tomcat-9.0.89.zip**

**66 ls -ltr**

**67 unzip apache-tomcat-9.0.89.zip**

**68 ls -ltr**

**69 rm -rf apache-tomcat-9.0.89.zip**

**70 ls -ltr**

**71 clear**

**72 ls -ltr**

**73 chmod -R 777 apache-tomcat-9.0.89/**

**74 cd apache-tomcat-9.0.89/**

**75 ls -ltr**

**76 pwd**

**77 cd bin**

**78 ls -ltr**

**79 cd ../**

**80 ls -ltr**

**81 cd lib**

**82 pwd**

**83 ls -ltr**

**84 cd ../**

**85 ls -ltr**

**86 cd logs/**

**87 ls -ltr**

**88 pwd**

**89 cd ../**

**90 ls -ltr**

**91 cd conf/**

**92 ls -ltr**

**93 cat server.xml**

**94 cd ..**

**95 ls -ltr**

**96 cd webapps/**

**97 ls -ltr**

**98 [wd**

**99 pwd**

**100 cd bin**

**101 pwd**

**102 cd../**

**103 cd bi**

**104 cd bin**

**105 pwd**

**106 clear**

**107 cd ../**

**108 cd bin**

**109 pwd**

**110 ls**

**111 cd ../logs**

**112 pwd**

**113 ls**

**114 cd ../**

**115 cd lib**

**116 pwd**

**117 ls**

**118 cd ../**

**119 cd conf**

**120 ls**

**121 pwd**

**122 cd ../**

**123 cd we**

**124 cd webapps/**

**125 pwd**

**126 ls -ltr**

**127 cd**

**128 cd /mnt**

**129 ls -ltr**

**130 cd project/**

**131 ls -ltr**

**132 cd target/**

**133 ls -ltr**

**134 cp LoginWebApp.war /mnt/servers/apache-tomcat-9.0.89/webapps/**

**135 cd /mnt/servers/apache-tomcat-9.0.89/webapps/**

**136 ls -ltr**

**137 cd ../**

**138 cd bin**

**139 ls -ltr**

**140 ./startup.sh**

**141 pwd**

**142 cd ../**

**143 cd logs/**

**144 ls -ltr**

**145 tail -100f catalina.out**

**146 cd ../**

**147 cd webapps/**

**148 ls -ltr**

**149 cp -r /mnt/game-of-life/gameoflife-web/target/gameoflife.war .**

**150 ls -ltr**

**151 cd ../**

**152 cd logs/**

**153 ls -lrt**

**154 tail -100f catalina.out**

**155 cd ../**

**156 cd webapps/**

**157 ls -ltr**

**158 history**

**Links:**

https://maven.apache.org/download.cgi

https://dlcdn.apache.org/maven/maven-3/3.9.7/binaries/apache-maven-3.9.7-bin.zip

https://tomcat.apache.org/download-90.cgi

<https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.89/bin/apache-tomcat-9.0.89.zip>

Apache Tomcat is a Webserver used to deploy/run/host java based web applications eg WAR file.

to download apache tomcat wget [https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.89/bin/apache-tomcat-9.0.89.zip unzip apache-tomcat-9.0.89.zip](https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.89/bin/apache-tomcat-9.0.89.zip%20unzip%20apache-tomcat-9.0.89.zip)

chmod -R 777 apache-tomcat-9.0.89

/mnt/servers/apache-tomcat-9.0.89 ---- this is TOMCAT HOME

[root@ip-172-31-45-200 bin]# pwd

/mnt/servers/apache-tomcat-9.0.89/bin ---- here we have all binaries i.e startup and shutdown scripts

[root@ip-172-31-45-200 bin]# ls

bootstrap.jar commons-daemon-native.tar.gz makebase.sh tomcat-juli.jar

catalina.bat configtest.bat setclasspath.bat tomcat-native.tar.gz

catalina.sh configtest.sh setclasspath.sh tool-wrapper.bat

catalina-tasks.xml daemon.sh shutdown.bat tool-wrapper.sh

ciphers.bat digest.bat shutdown.sh version.bat

ciphers.sh digest.sh startup.bat version.sh

commons-daemon.jar makebase.bat startup.sh

[root@ip-172-31-45-200 bin]#

[root@ip-172-31-45-200 logs]# pwd

/mnt/servers/apache-tomcat-9.0.89/logs ---- tomcat logs are generated here

[root@ip-172-31-45-200 logs]# ls

[root@ip-172-31-45-200 lib]# pwd

/mnt/servers/apache-tomcat-9.0.89/lib -- here all the tomcat libraries are present

[root@ip-172-31-45-200 lib]# ls

annotations-api.jar el-api.jar tomcat-dbcp.jar tomcat-i18n-ru.jar

catalina-ant.jar jasper-el.jar tomcat-i18n-cs.jar tomcat-i18n-zh-CN.jar

catalina-ha.jar jasper.jar tomcat-i18n-de.jar tomcat-jdbc.jar

catalina.jar jaspic-api.jar tomcat-i18n-es.jar tomcat-jni.jar

catalina-ssi.jar jsp-api.jar tomcat-i18n-fr.jar tomcat-util.jar

catalina-storeconfig.jar servlet-api.jar tomcat-i18n-ja.jar tomcat-util-scan.jar

catalina-tribes.jar tomcat-api.jar tomcat-i18n-ko.jar tomcat-websocket.jar

ecj-4.20.jar tomcat-coyote.jar tomcat-i18n-pt-BR.jar websocket-api.jar

[root@ip-172-31-45-200 lib]#

[root@ip-172-31-45-200 conf]# ls

catalina.policy jaspic-providers.xml server.xml web.xml

catalina.properties jaspic-providers.xsd tomcat-users.xml

context.xml logging.properties tomcat-users.xsd

[root@ip-172-31-45-200 conf]# pwd

/mnt/servers/apache-tomcat-9.0.89/conf --- here all the tomcat configuration files are present

server.xml is the main tomcat config file in which we can also the the connector port 8080 which we can use to connect tomcat.

[root@ip-172-31-45-200 webapps]# pwd

/mnt/servers/apache-tomcat-9.0.89/webapps --- deploy/copy the war file in this location

[root@ip-172-31-45-200 webapps]#