MAVEN

DAY-1: ==========

#1

Java Build and Deployment End-To-End Workflow.

#2

Basics

- Java program
- manual compilation

#3

What is Maven? Why we need a build tool?

#4

Installation

- Download JDK 1.8(jdk-8u251-linux-x64.tar.gz) and extract it to your favourite location

Download URL - https://www.oracle.com/in/java/technologies/javase/javase-jdk8-downloads.html

- Download lastest Maven(apache-maven-3.6.3-bin.tar.gz) and extract it to your favourite location

Download URL - https://maven.apache.org/download.cgi

- Setup 'JAVA_HOME' Envi:qronment variable for Maven to locate JDK.
- Setup 'M2_HOME' for PATH variable
- Setup 'PATH' to run jdk and maven commands from any directory in the system.

1. JAVA_HOME
2. M2_HOME
3. PATH

Installing JDK and Maven:

==========

A. open $USER_HOME/.bashrc

B. create below Environment variables

export M2_HOME=/home/gamut/mavensoftwares/apache-maven-3.3.9

export JAVA_HOME=/home/gamut/mavensoftwares/jdk1.8.0_121

export PATH=$M2_HOME/bin:$JAVA_HOME/bin:$PATH

Load .bashrc using below command or open new ternimial

$ source .bashrc

Verify Installation

$ javac -version

$ mvn --version

DAY-2:

#5 TEST YOUR KOWLEDGE

Build and Deployment E2E work-flow and basics

- What is compilation & why we compile the source code?
- Packaging sequence for Java application
- What is Build
- What is Deployment
- Environment
- Dev, QA & DevOps teams Interaction and Collaboration.

#6

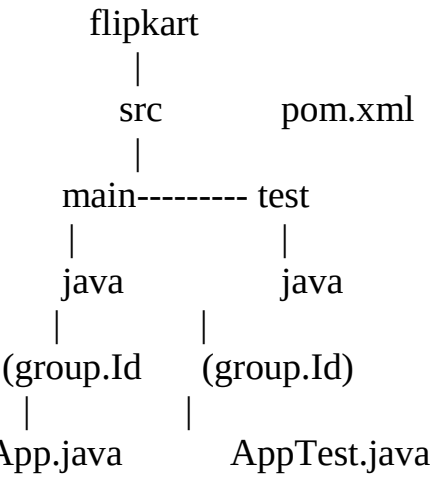
Maven's standard project layout

==========

Project creation:

---

java projects which are created by maven, ideally follows below project folder structure.



flipkart - is called "Project name" / "ArtifactID"
src - Source folder which contains the
application source code
main - Contains application's main functional code
test - Contains application's unit testing code
pom.xml - Maven's build file using which we can
configure build steps such as
compilation, test runs, jar/war creation,
deployments...etc.

DAY-3:
#8
Building the maven project:

# Install git
$ sudo apt-get update
$ sudo apt-get install git
$ git --version --> verify git installation.

#
Clone the code from Git Or create your own Project using below Maven command
Clone:
$ git clone https://github.com/nageshvkn/flipkart.git

#
Building the project using Maven. Use below command.
$ mvn install

$ mvn install - command executes below "build life cycle phases" automatically.

- initialize
prepares project with initial pre-
requisites ex: creating necessary directory structure (i.e. target directory)
- validate
validate project's folder structure
- compile

compiles "main" java code.

- test-compile
  compiles "test" java code
- test
  Runs the test cases and generates test reports.
- package
  creates jar/war.
- install
  copy built artifacts i.e jar/war file into local repository $USER_HOME/.m2
  folder.

#
Verify Built artifacts:

Go to "target" folder and observe below.

target
|
classes test-classes surefire-reports jar/war file

classes: directory contains compiled class files
of main source code
test-classes: directory contains compiled class
files of test source code

surefire-reports: contains test reports.

flipkart-1.0-SNAPSHOT.jar: jar file of the main code

Note:
first time when we run 'mvn install' command, Maven downloads all missing dependencies into .m2 from maven's central repository.
So, we need to have internet when we run 'mvn install' command first time.

#
Understanding pom.xml file structure

#10
Artifact path in local repository .m2
$USER_HOME/.m2/repository/groupId/artifactId/version/jar-OR-war-file

- Package naming convention:
    artifactId-version.jar/war

DAY-4:
#11:
Maven has "Automatic Dependency resolution" feature

- Direct dependency
- Transitive dependency (A -> B, B -> C, A -> B & C)
  (B is direct dependency for A and C is Transitive dependecy for A)
  (To compile AppTest.java, we need Junit.jar. So Junit.jar becomes direct dependency.
  To execute Junit.jar, we need hamcrest.jar. So hamcrest.jar becomes transitive
  dependency)

Maven way of dependency addition, detection from different repositories.

#12
Maven repositories
- Central
- Private (You can setup using Nexus / Jfrog-atifactrory tool)
- Local (.m2)

#13
$ mvn deploy (uploads jar files to private (nexus) repository)

#14
"mvn clean" & Build types
- Complete Build
- Incremental Build
- Daily Build

- Nightly Build

# Skip test cases
$ mvn install -DskipTest (skip test cases execution)
$ mvn install -Dmaven.test.skip=true (skips compilation and test case execution)

$ mvn compiler:testCompile (Compiles only testing code)
$ mvn surefire:test (Executes the test cases and generates reports)

#18
Where Maven's coordinates / GAV parameters are helpful?
- To decide artifact storage path in local repository.
- To decide jar/war name
- To define a dependency in pom.xml file.

#19
PROJECT-02: WEB Application Build and Deployment.

Goal:
- Create a project for flipkart web application and perform end-to-end build and deployment
- Handling build and deployment for any web application

Steps:

# Install git
$ sudo apt-get update
$ sudo apt-get install git
$ git --version --> verify git installation

#
Clone the code from Git
$ git clone https://github.com/nageshvkn/iflipkart.git
OR
Create a new project:
$ mvn archetype:generate -DgroupId=com.flipkart -DartifactId=flipkart -Dversion=1.0-SNAPSHOT -DinteractiveMode=false -DarchetypeArtifactId=maven-archetype-webapp

#
Building the project using Maven. Use below command.

```
$ mvn install
```

#
Check final artifact i.e flipkart.war file in target directory

```
$ ls target
```

#
Set-up tomcat for deployment
- Download tomcat *.tar.gz and JDK.
- Extract to your favouriate location
- Make sure JAVA_HOME environment variable is set

#
Deploy flipkart.war into tomcat deployment path

```
$ cp target/flipkart.war $TOMCAT_HOME/webapps
```

#
Start tomcat server

```
$ cd $TOMCAT_HOME/bin
$ ./startup.sh
```

#
Launch application with below URL.
http://localhost:8080/flipkart

Syntax:
[http://TomcatServerIP:Port/WarFilename]

#26
Project:2 (Real-time End-to-End Build and Deployment Process)
==========
Goals:
- Building the War for large scale real-time kinda application
- Learning Deployments with a dedicated tomcat Server

Steps:

#
Install GIT
$ sudo apt-get update
$ sudo apt-get install git

# Clone gamutkart application source code into the build server from below "gamutkart" github repository.
$ git clone https://github.com/nageshvkn/gamutkart2.git

3.
Build "gamutkart" application using below command.
$ mvn install

4.
Deploy 'gamutkart.war' application to remote server deployment location(i.e $TOMCAT_HOME/webapps) using below command.

$ cp target/gamutgurus.war $TOMCAT_HOME/webapps

4A.
After the deployment, we need start the server using below tomcat startup comamnd.

$ cd $TOMCAT_HOME/bin
$ ./startup.sh

5.
Tomcat by default runs on port 8080. So application can be accessed with below URL.

ex: http://IPAddress:8080/gamutgurus
http://localhost:8080/gamutgurus (If application is deployed in local machine)

6. NOTE:
In case there is any issue in the application, errors will be logged in "$TOMCAT_HOME/logs/catalina.2017-03-24.log" file.

We can check this file and if there are any errors/exceptions, we provide this information to developers.

7.
Note: If you want to change the port number, Go to below file and change the port number where you see something like this. (port="8080" protocol="HTTP/1.1")

$ vim $TOMCAT_HOME/conf/server.xml

#
Interview Notes:

Explain Maven Life cycle phases:
==========
- When we execute '$ mvn install' command..
  - By default maven reads 'pom.xml' file and runs the given target(i.e install). before running 'install' target, It will read GAV paramerters, packaging attribute for creating jar/war file and also downloads declared dependencies from central/remote repositories.
  - According to Maven's build lifecycle, maven executes all required build steps such as:
    initialize - creates all necessary folder structure
    validate - validates project structure
    compile - compiles main code
    test-compile - compiles test code
    test - executes test cases and generates reports
    package - creates jar/war file
    install - copies jar/war file into local repository i.e ~$USER_HOME/.m2

Qns:
1. How do you continue the build even after compilation failures.
   for Main & Test
failOnError=true

2. Maven & ANT comparison