**MAVEN Tool**

**=================**

**By sekhar sir**

**q) What is a build process?**

ans:

->If we create a **java class** then its **build process is**

i) **compile** the source file.

ii) **packing** it into a jar(java archive file)file.

->If we create a **servlet class** then its **build process is**

i) **compile** the source file.

ii) **packing** into a war file(web archive file).

iii) **deploy** the war into a servlet container.

iv) **start** the server if required.

-> If we make a change in java class or servlet class then we need to execute its build process again.

-> In real time application development a project contains one or more java classes, and executing its build process again and again will increase the burden on the programmer and also the development team of the project.

**q) Why build tools?**

ans:

-> Build tools are introduced to make the build process of a java project as easy.

-> We can think a build tool like a batch file, when a batch file is executing then all commands of batch file are executed in sequence.

-> When a build tool is executed then all build process of the project is executed in sequence.

**q) Why MAVEN tool?**

ans:

-> Before MAVEN tool we got ANT(Another Neat Tool) for building the java projects.

-> With ANT tool, it is only possible to make a build process of a project. All the remaining work should be done by the programmer.

-> To overcome the problem with the ANT tool the same vendor i.e, Apache Software Foundation has given another tool called MAVEN.

**->MAVEN will do the following.**

i) It creates our project structure automatically.

ii) It will download’s all jars and dependent jars with proper version automatically.

iii) It makes the build process of a project.

iv) It takes care about overall project management.

**MAVEN setup:**

-> Download MAVEN zip file(apache-maven-3.1.1-bin-zip) from maven.apache.org.

-> Extract the downloaded zip file.

-> Set the following two environment variables

i) JAVA-HOME: c:\ProgramFiles\java\jdk1.7

ii) path: d:\apache-maven-3.1.1\bin

-> To test the MAVEN setup, type the following command from command prompt,

c:\>mvn --version

**Important terms in understanding MAVEN project:**

1) archetype.

2) groupId.

3) artifactId.

4) version.

5) package.

**1)archetype**:

--> An archetype is a model through which all projects and all that model types can be created.

--> In other words archetype is a project template, using this a particular java project can be build using MAVEN tool.

--> MAVEN has given a set of predefined archetypes for building different types of java projects.

**ex:**

maven-archetype-quickstart : It is an archetype for building standalone java projects.

maven-arhcetype-webapp : It is an archetype for building web application projects in java.

srtuts2-archetype-starter : It is an archetype for building struts2.x framework application etc...........

**2)groupId:**

--> A groupId is an id given to uniquely represent one or more artifacts generated through MAVEN by an organization.

" an artifact is a output file generated by MAVEN through its build process "

--> One organization can generate multiple artifacts and all those artifacts are identified with a groupId.

**ex:**

org.Springframework : Is a groupId for all the artifacts generated by spring framework organization.

org.apache.struts : Is a groupId for all the artifacts generated by struts framework organization.

**3) Artifact Id:**

--> An artifact Id is a unique id given to identify the artifact generated by the MAVEN tool.

--> An artifact Id is also used as a root folder of a project structure.

Note: Think a group Id as analogue to a java package, and artifact Id as an analogue to a java class.

--> Multiple java classes are identified by the java package name and similarly multiple artifacts are identified by the group Id.

**4) Version:**

--> A version is used to assign a number to the artifact generated by the MAVEN tool.

--> By default MAVEN provides “1.0-SNAPSHOT” as version for an artifact.

--> In MAVEN terms, a SNAPSHOT is a word to indicate that the generated artifact is not final. It may be next added with additional features.

--> After an artifact is finalized, means added with all features then it will be named as RELEASE in real time.

**5) Package:**

-->In real time application, each java class must be placed under the package. So MAVEN also asks a package name for java classes, while createing a project structure.

**Creating a simple java project using MAVEN:**

1) --> Create a folder for storing maven projects created using the tool.

--> Type the following in the command prompt,

>mvn archetype:generate (to create any type of java project using MAVEN)

ex: D:/>MavenExamples>mvn archetype:generate

>Choose a number or apply filter :343:343 (343-indicates an archetype for creating stand-alone applications).

> Define a value for Property 'groupId' : com.sathya

> Define a value for Property 'Version' : 1.0-SNAPSHOT:

> Define a value for Property 'Package' : com.sathya: com.pack1

y:: y

Then finally we got a message on command prompt

Buid Success

3) --> The directory Strucure created by the MAVEN for the project will be like the following

" see the directory structure which present inside the folder that u created for the MAVEN projects"

4) --> Type the following

D:MAVENExamples/DemoApp>mvn compile

--> Now target folder will be created under the root(DemoApp)

--> Classes folder under target folder will store the generated class files

" see the directory structure once again that u defined"

5) Type the follwoing

D:MAVENExamples/DemoApp>mvn package

--> Now a java file called DemoApp-1.0-SNAPSHOT.jar will be generated and stored in target folder to run the sample app class, type the following command,

D:MAVENExamples/DemoApp>java target/DemoApp-1.0-SNAPSHOT.jar com.sathya.App

Then finally we got the response on command prompt like the following,

maven world

=================================================================================================

**Build Process of a project in MAVEN:**

1) validate

2) compile

3) test

4) package

5) install

--> The above phases are also called the build life cycle

--> Each Phase is also called as a goal and when a goal is executed then all of its above goals are got executed.

--> validate : Is a goal which checks the files in directory structure are proper or not.

--> compile : Compile source files.

--> test : execute test cases

--> package : Creates the jar or war file, according to the project requirement.

--> install : The artifact generated will be installed in a local repository.

--> We install an artifact in a local repository, if another project is depending on that artifact.

=================================================================================================

**MAVEN repositories:**

1) Local repository

2) Central repository

3) Remote repository (this we dont use much)

--> Local repository is a place where MAVEN stores all the downloaded plugins and jars while creating a project.

--> If we create another project of same type then it will load the plugins and jars from local repository.

--> The location of the local repository is,

C:/DocumentsandSettings/Welcome/.m2/repository

--> When a project is creating through MAVEN, all thee required plugins and jars for the first time all downloaded from the central repositroy of the MAVEN, " so to work with MAVEN tool, we need INTERNET connection for the system ".

=================================================================================================

**Creating a Web application using MAVEN:**

1) Type the following command on the command prompt'

D:/MAVENExamples> mvn archetype:generate

> choose a number : 23(type a web application archetype number)

> groupId: com.sathya.web

> artifactId: SampleWeb

> version: 1.0-SNAPSHOT

> package: com.sathya.web

y:y

2) Then see the directory structure created by the MAVEN tool in the current directory.

3) Create a servlet class like the following,

//HelloServlet.java

package com.sathya.web:

import javax.servlet.\*;

import java.io.\*;

public class HelloServlet extends GenericServlet

{

public void service(ServeltRequest req,ServletResponse res)throws ServletException,IOException

{

Printwriter pw=re.getWriter();

pw.println("<h1>Hello friends</h1>");

pw.close();

}

}

4) Open index.jsp and from web-app and add the following code,

<body>

<center>

<h2><a href="srv1">click me</a>

</h2>

</center>

</body>

5) Open web.xml from WEB-INF and configure the servlet.

6) Type the following command in the prompt like the following,

D:MavenExamples/SampleWeb>mvn package

finally we got the following response on the command prompt,

Buid Success

--> A war file is created with the name SampleWeb-1.0-SNAPSHOT.war and it is stored in target folder.

--> Copy the war file into webapp folder of TomCat server, then start the server.

--> Open the browser and type the following request from the address bar.

http://localhost:2013/SampleWeb-1.0-SNAPSHOT/index.jsp

=================================================================================================

**Making a java application as compatable with eclipse IDE:**

1) Open command prompt and enter into DemoApp directory,

D:/MavenExamples/DemoApp>mvn clean

D:/MavenExamples/DemoApp>mvn eclipse:eclipse

2) Open eclipse IDE and enter some workspace name

3) And do the following procedure in eclispe ide,

File->import->expand general->existing project from workspace->select DemoApp->click on finish.

4) To remove problems of the project do the following,

---> Right click on project->buidpath->configure buidpath->librarie->add variables->configure variables->new->enter the following,

variable: M2-REPO

path: C:/DocumentsAndSettings/ABC/.m2/repository->ok->ok.

=================================================================================================

**Making a MAVEN WebApplication as compatable with eclipse IDE:**

---> It is similar to the above standalone application, but we used to pass a parameter from the command prompt

wtpversion

D:/MAVENExamples/SapmleWeb>mvn clean

D:/MAVENExamples/SampleWeb>mvn eclipse:eclipse -Dwtpversion=2.0