**Maven Archetype details**

<https://maven.apache.org/guides/introduction/introduction-to-archetypes.html>

**To create Root Project that will have multiple modules**

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

mvn archetype:generate -DarchetypeGroupId=org.codehaus.mojo.archetypes -DarchetypeArtifactId=pom-root -DarchetypeVersion=RELEASE

**To create a sample Project/module inside the parent project**

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

mvn archetype:generate -DarchetypeGroupId=org.apache.maven.archetypes -DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=RELEASE

**To create a web module inside the parent project**

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

mvn archetype:generate -DgroupId=[your project's group id] -DartifactId=[your project's artifa ct id] -DarchetypeArtifactId=maven-archetype-webapp

OR

===

mvn archetype:generate -DarchetypeGroupId=org.codehaus.mojo.archetypes -DarchetypeArtifactId=webapp-javaee6 -DarchetypeVersion=RELEASE

* Dependency Management just control the version of the dependencies, but to assign any dependency we need to define it in dependencies>dependency
* If we define any dependency into the parent POM then all dependency will inherit into all of its sub module

**Optional**

* Install maven
* Install maven plugin in eclipse
* Create a new maven project through eclipse (no archetype), this will be the parent of all modules. Package type should be POM
* Create all sub-module API and EJB from eclipse maven module (no archetype), select the parent and the type of the package. Ear package type should be POM
* For war file use maven web app archetype or don’t select any archetype and select package type war and then finally inside webapp directory create WEB-INF, also WEB-INF/web.xml
* For ejb and api we may need to create META-INF directory into the resources folder
* Dependency Management just control the version of the dependencies, but to assign any dependency we need to define it in dependencies>dependency
* If we define any dependency into the parent POM then all dependency will inherit into all of its sub module
* For example, we can define java ee dependency management into the parent pom and dependency into its submodule. This way sub-module don’t need to worry about the version. Any module that needs jboss-javaee they will define dependency into their own pom file.

Parent POM:

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.jboss.spec</groupId>

<artifactId>jboss-javaee-6.0</artifactId>

<version>3.0.3.Final</version>

<type>pom</type>

</dependency>

</dependencies>

</dependencyManagement>

Sub-Module POM:

<dependency>

<groupId>org.jboss.spec</groupId>

<artifactId>jboss-javaee-6.0</artifactId>

<scope>provided</scope>

<type>pom</type>

</dependency>

* To define the java version to use we can use the following property into the individual POM

<properties>

<maven.compiler.source>1.7</maven.compiler.source>

<maven.compiler.target>1.7</maven.compiler.target>

</properties>

How to use maven EAR plugin

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

* In the POM.xml of the ear package, define all the dependency that needs to be packaged into the ear file and then into the ear plugin define the ejb and web module. See the example below –

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-ear-plugin</artifactId>

<configuration>

<version>${version.jee}</version>

<defaultLibBundleDir>lib</defaultLibBundleDir>

<!-- <archive>

<manifestEntries>

<Dependencies>${project.ear.Dependencies}</Dependencies>

</manifestEntries>

</archive> -->

<modules>

<ejbModule>

<groupId>${project.parent.groupId}</groupId>

<artifactId>first-project-ejb</artifactId>

</ejbModule>

<webModule>

<groupId>${project.parent.groupId}</groupId>

<artifactId>first-project-local-client</artifactId>

<contextRoot>/${project.parent.artifactId}</contextRoot>

</webModule>

</modules>

</configuration>

</plugin>

</plugins>

</build>

<dependencies>

<dependency>

<groupId>${project.parent.groupId}</groupId>

<artifactId>first-project-api</artifactId>

<version>${project.version}</version>

<type>jar</type>

</dependency>

<dependency>

<groupId>${project.parent.groupId}</groupId>

<artifactId>first-project-ejb</artifactId>

<type>ejb</type>

<version>${project.version}</version>

</dependency>

<dependency>

<groupId>${project.parent.groupId}</groupId>

<artifactId>first-project-local-client</artifactId>

<version>${project.version}</version>

<type>war</type>

</dependency>

</dependencies>

* Then the URL for the resource will be - <http://localhost:8080/first-project/client> first-project is the name of the parent project, used as context root and the client is the name of the servlet.
* This ear plugin will create ear file that will include a lib directory. API will be in this lib directory and this api will be in the class path of both ejb and client-web module. Lib is the parent module and everything is the package is the sub-module. So, if we put anything in the lib module will be available for all of its sub module.
* For the above reason, in web module and in the ejb module, we need to define API dependency as PROVIDED as it will be included into the lib directory and controlled by ear plugin.

**Maven WAR Plugin**

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-war-plugin</artifactId>

<version>3.1.0</version>

<configuration>

<warName>remote</warName>

<archive>

<manifest>

<addClasspath>true</addClasspath>

</manifest>

</archive>

</configuration>

</plugin>

</plugins>

</build>

<dependencies>

<dependency>

<groupId>${project.groupId}</groupId>

<artifactId>first-project-api</artifactId>

<version>${project.version}</version>

</dependency>

</dependencies>

* War name in the plugin will be used as URL - <http://localhost:8080/remote/test>
* Maven will automatically generates META-INF and WEB-INF
* API shouldn’t be declare as Provided, as we want it to package within war

**Jboss-as plugin to deploy in jboss**

<plugin>

<groupId>org.jboss.as.plugins</groupId>

<artifactId>jboss-as-maven-plugin</artifactId>

<version>7.9.Final</version>

<executions>

<execution>

<phase>install</phase>

<goals>

<goal>deploy</goal>

</goals>

</execution>

</executions>

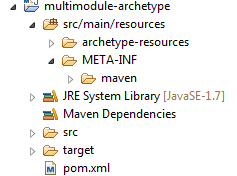
</plugin>

* Add this configuration into build along with ear/war plugin
* Jboss needs to be up running
* Go to ear module Then trigger deployment by the command “mvn clean install”, this will deploy the ear into the jboss in maven install phase.

**How to create an archetype**

1. **Form eclipse create a maven project- search by maven-archetype-archetype**
2. Remember the artefact and group id of this project

Project structure will look like below –



<groupId>com.maksud</groupId>

<artifactId>multimodule-archetype</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>Archetype - multimodule-archetype</name>

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

<properties>

<maven.compiler.source>1.7</maven.compiler.source>

<maven.compiler.target>1.7</maven.compiler.target>

</properties>

1. **In the Meta-INF/maven change archetype.xml to archetype-metadata.xml**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<archetype-descriptor xsi:schemaLocation=*"http://maven.apache.org/plugins/maven-archetype-plugin/archetype-descriptor/1.0.0 http://maven.apache.org/xsd/archetype-descriptor-1.0.0.xsd"* name=*"service-framework-ear"*

xmlns=*"http://maven.apache.org/plugins/maven-archetype-plugin/archetype-descriptor/1.0.0"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*>

<fileSets>

<fileSet encoding=*"UTF-8"*>

<directory>.settings</directory>

<includes>

<include>\*\*/\*.prefs</include>

</includes>

</fileSet>

<fileSet filtered=*"true"* encoding=*"UTF-8"*>

<directory></directory>

<includes>

<include>.project</include>

</includes>

</fileSet>

<fileSet encoding=*"UTF-8"*>

<directory></directory>

<includes>

<include>README.TXT</include>

<include>.gitignore</include>

</includes>

</fileSet>

</fileSets>

<modules>

<module id=*"${rootArtifactId}-api"* dir=*"\_\_rootArtifactId\_\_-api"* name=*"${rootArtifactId}-api"*>

<fileSets>

<fileSet filtered=*"true"* packaged=*"true"* encoding=*"UTF-8"*>

<directory>src/main/java</directory>

<includes>

<include>\*\*/\*.java</include>

</includes>

</fileSet>

<fileSet filtered=*"true"* encoding=*"UTF-8"*>

<directory>src/main/resources</directory>

<includes>

<include>\*\*/\*.xml</include>

</includes>

</fileSet>

<fileSet encoding=*"UTF-8"*>

<directory>.settings</directory>

<includes>

<include>\*\*/\*.prefs</include>

</includes>

</fileSet>

<fileSet filtered=*"true"* encoding=*"UTF-8"*>

<directory></directory>

<includes>

<include>.classpath</include>

<include>.project</include>

</includes>

</fileSet>

<fileSet encoding=*"UTF-8"*>

<directory></directory>

<includes>

<include>.gitignore</include>

</includes>

</fileSet>

</fileSets>

</module>

<module id=*"${rootArtifactId}-ejb"* dir=*"\_\_rootArtifactId\_\_-ejb"* name=*"${rootArtifactId}-ejb"*>

<fileSets>

<fileSet filtered=*"true"* packaged=*"true"* encoding=*"UTF-8"*>

<directory>src/main/java</directory>

<includes>

<include>\*\*/\*.java</include>

</includes>

</fileSet>

<fileSet filtered=*"true"* encoding=*"UTF-8"*>

<directory>src/main/resources</directory>

<includes>

<include>\*\*/\*.xml</include>

<include>\*\*/\*.properties</include>

</includes>

</fileSet>

<fileSet encoding=*"UTF-8"*>

<directory>.settings</directory>

<includes>

<include>\*\*/\*.prefs</include>

</includes>

</fileSet>

<fileSet filtered=*"true"* encoding=*"UTF-8"*>

<directory></directory>

<includes>

<include>.classpath</include>

<include>.project</include>

</includes>

</fileSet>

<fileSet encoding=*"UTF-8"*>

<directory></directory>

<includes>

<include>.gitignore</include>

</includes>

</fileSet>

</fileSets>

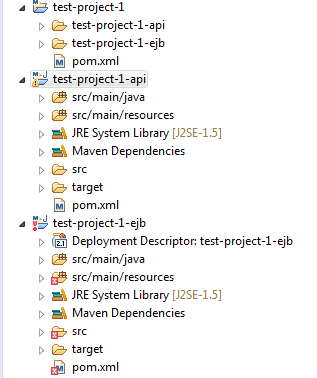
</module>

</modules>

</archetype-descriptor>

1. See the content of the xml file above-

Modules are the list of module that we want them to generates in our final project, like -



1. Module parameter



1: This variable denotes the name artefact id of the parent project multimodule-archetype\src\main\resources\archetype-resources\pom.xml

<groupId>${groupId}</groupId>

<artifactId>${artifactId}</artifactId>

<version>${version}</version>

<packaging>pom</packaging>

2: This name is the same name as the project we create into multimodule-archetype\src\main\resources\archetype-resources

1. **Create a pom.xml file inside multimodule-archetype\src\main\resources\archetype-resources**
2. This pom will use project main artifact and group id and will also define all dependnency management common to all the sub modules –

<groupId>${groupId}</groupId>

<artifactId>${artifactId}</artifactId>

<version>${version}</version>

<packaging>pom</packaging>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.jboss.spec</groupId>

<artifactId>jboss-javaee-6.0</artifactId>

<version>3.0.3.Final</version>

<type>pom</type>

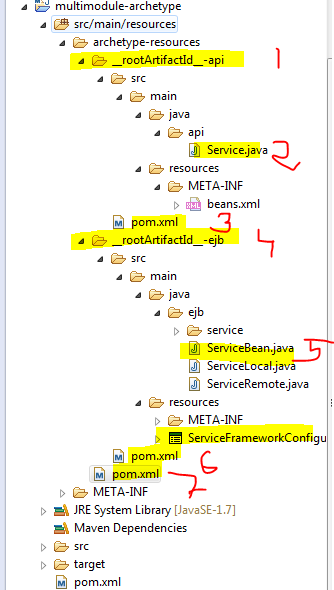
</dependency>

</dependencies>

</dependencyManagement>

</project>

1. Create all the template modules, modules’ classes, modules’ dependencies (in pom) and modules’ resources inside multimodule-archetype\src\main\resources\archetype-resources. Modules names should be the same name we have mentioned in 5.2



8.1: create a maven qucikstart project and name it the same name as defined in archetype-metadata.xml

8.2: Same java class that we want to put it in out generated project

8.3: in this POM.xml we can define all the dependency we require for the corresponding module in the generated project.

8.4: same as 8.1, this is the implementation

8.5: Same as 8.2

8.6: same as 8.3- all the dependency we need for this particular module

8.7. pom file that is belongs to archetype-resources directory, we have mentioned in step 6

1. This was we can create as many module as we want, also the way we want our generated folder needs to looks like. Like ejb and api, we can use it for war, ear, rar etc
2. Mvn clean install will publish the archetype in the local repo and we can create any project from that archetype. This project will have the same pattern we have defined in the archetype.

mvn archetype:generate -DarchetypeGroupId=com.maksud -DarchetypeArtifactId=multimodule-archetype -DarchetypeVersion=0.0.1-SNAPSHOT -DgroupId=org.test -DartifactId=test-project-1