#mvn -version

#mvn archetype:generate

'groupId': com.epam.brest.cource

'artifactId': sampleuserapp

**Fully qualified name (FQN):**

packagename.packagename2.packagename3.ClassName

JDK/JRE/ JAVAC(синтакис, лексика, семантика, оптимизация) -> ByteCode -> JVM

1. #java App - wrong. FQN is needed
2. #java com.epam.brest.cource.App - wrong, there is no `com.epam.brest.cource` folder in the current working directory
3. #java -classpath . com.epam.brest.cource.App - wrong
4. #java -cp ../../../../ com.epam.brest.cource.App
5. #java -cp absPath com.epam.brest.cource.App
6. echo Main-Class: com.epam.brest.cource.App > manifest.txt
7. jar cvfm App.jar manifest.txt com/epam/brest/cource/\*.class
8. javac com/epam/brest/cource/App.java
9. java -jar App.jar

Ant / Ivy / Maven / Gradle (Groovy)

**ANT (Example)**

<project name="MyProject" basedir=".">  
  
 <property name="version">1.2.3</property>  
 <property name="dir.src">src/main/java</property>  
 <property name="dir.build">build</property>  
 <property name="dir.build.classes">${dir.build}/classes</property>  
 <property name="dir.build.javadoc">${dir.build}/javadoc</property>  
 <property name="file.jar">${dir.build}/MyProject-${version}.jar</property>  
  
 <path id="projectClasspath">  
 <fileset dir="lib">  
 <include name="\*\*.jar"/>  
 </fileset>  
 </path>  
  
 <target name="clean">  
 <delete dir="${dir.build}"/>  
 </target>  
  
 <target name="init">  
 <mkdir dir="${dir.build}"/>  
 <mkdir dir="${dir.build.classes}"/>  
 </target>  
  
 <target name="compile" depends="init">  
 <echo>Compiling Java source</echo>  
 <javac classpathref="projectClasspath" srcdir="${dir.src}" destdir="${dir.build.classes}" />  
 </target>  
  
 <target name="jar" depends="compile">  
 <echo>Making JAR file</echo>  
 <jar basedir="${dir.build.classes}" file="${file.jar}" />  
 </target>  
  
 <target name="javadoc">  
 <echo>Making JavaDoc from source</echo>  
 <javadoc sourcepath="${dir.src}" destdir="${dir.build.javadoc}"></javadoc>  
 </target>  
  
</project>

**Gradle (Example)**

**apply plugin: "idea"  
apply plugin: "eclipse"  
  
subprojects {  
 apply plugin: "java"  
  
 tasks.withType(JavaCompile) {  
 sourceCompatibility = JavaVersion.VERSION\_1\_7  
 targetCompatibility = JavaVersion.VERSION\_1\_7  
 }  
  
 repositories {  
 mavenCentral()  
 }  
}  
  
task wrapper(type: Wrapper) {  
 gradleVersion = "1.12"  
}**

**MAVEN**

**POM** - Project Object Model

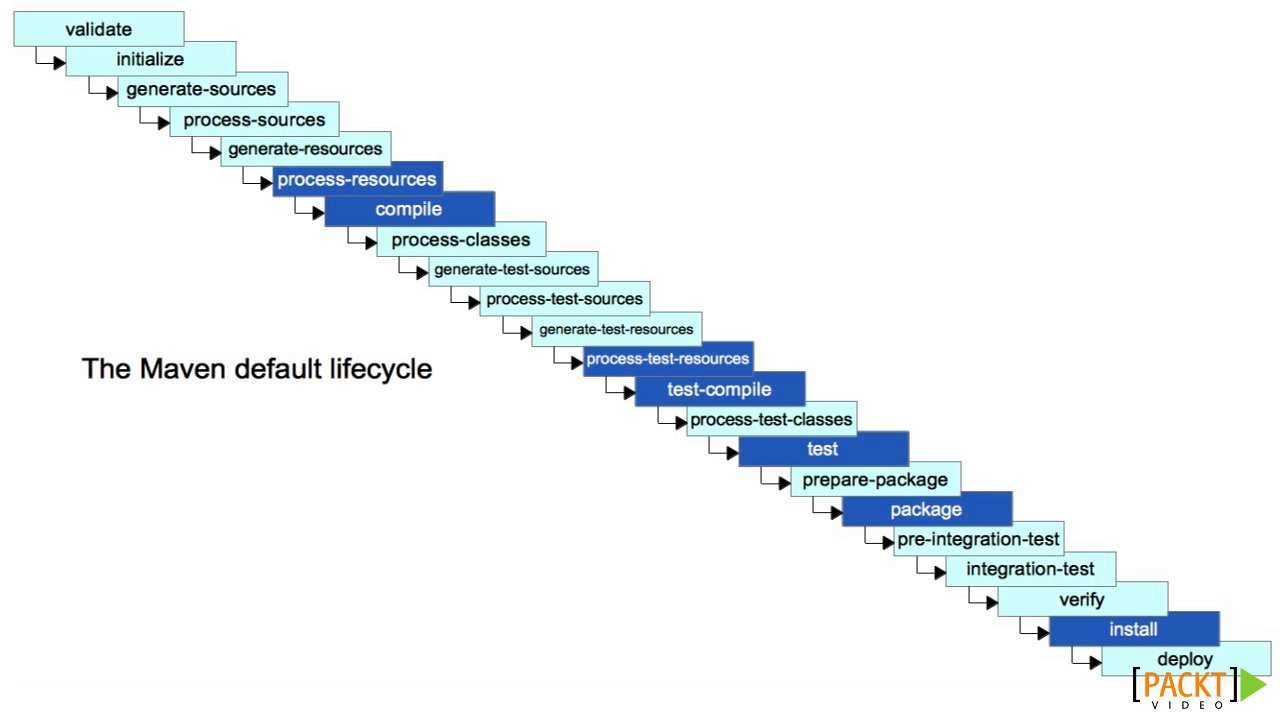
Концепции:

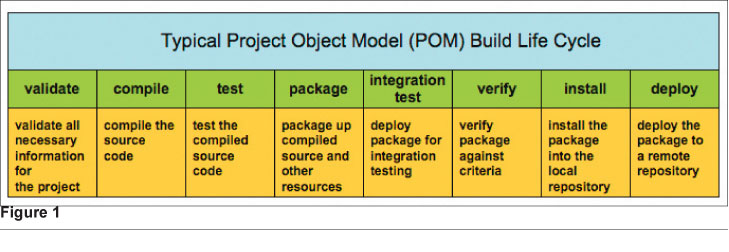
1. Inheritance (наследование):

* super pom
* org parent pom
* project root pom
* module pom

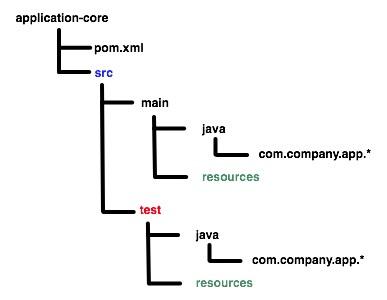
1. Plugins
2. Repositories (artefacts, Artefactory, Nexus, Maven Cental, hosting: jetty)
3. Dependencies
4. IDE Independence

**MAVEN Lifecycles:**

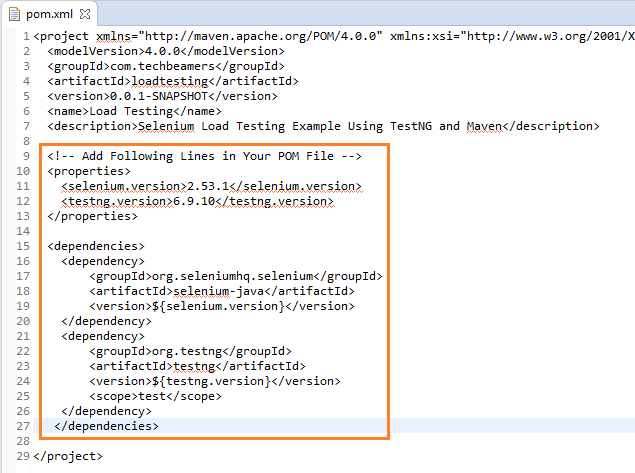




**MAVEN Project Structure:**



**MAVEN Versioning:**



**MAVEN Phases:**

#mvn clean

#mvn test

#mvn clean package

#mvn -help

#mvn -X -e clean package

#mvn dependency:tree

#mvn dependency:tree -Dverbose -Dincludes=commons-collections

<plugin>

<groupId>org.mortbay.jetty</groupId>

<artifactId>maven-jetty-plugin</artifactId>

<version>6.1.26</version>

<configuration>

<scanIntervalSeconds>10</scanIntervalSeconds>

<stopKey>stop</stopKey>

<stopPort>9999</stopPort>

</configuration>

<executions>

<execution>

<id>start-jetty</id>

<phase>pre-integration-test</phase>

<goals>

<goal>run</goal>

</goals>

<configuration>

<scanIntervalSeconds>0</scanIntervalSeconds>

<daemon>true</daemon>

</configuration>

</execution>

<execution>

<id>stop-jetty</id>

<phase>post-integration-test</phase>

<goals>

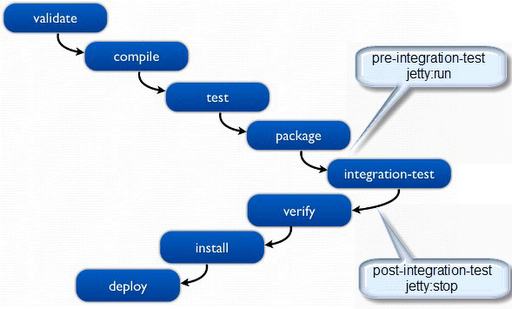
<goal>stop</goal>

</goals>

</execution>

</executions>

</plugin>



**MAVEN Profile:**



**MAVEN Plugin:**

<build>

<plugins>

<plugin> <!-- Build an executable JAR -->

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

<artifactId>maven-jar-plugin</artifactId>

<version>3.0.2</version>

<configuration>

<archive>

<manifest>

<addClasspath>true</addClasspath>

<classpathPrefix>lib/</classpathPrefix>

<mainClass>com.mypackage.MyClass</mainClass>

</manifest>

</archive>

</configuration>

</plugin>

</plugins>

</build>

**GIT:**

#git clone …

mv -> git\_folder

cd git\_folder

#git status

#git add --all

#git status

#git commit -am "<commit message>"

#git push