Mvn compile----it will create app.class in target folder

Mvn test-----in target it created surifie.reports----we can say log file---every devekloper has going to check this file for any test cases failure….it will create apptest.class

Mvn package----it will create pjar package

**Artifactid-snapshot version.jar---------**the jar file will create with this name

Eg: test-artifact-1.0-snapshot.jar

Mvn install------it will install dependcies in local

Mvn versions:set -Dnewversions=4.0------it will chande snapshot into new version

And the old snapshot version pom.xml it will automatically take back up like **pom.xml.versionsBackup**

**Mvn versions:revert ------then** the backup version (**pom.xml.versionsBackup ) is reverted and this file will go off and the changes in the pom will changes to previous version**

**Eg: Suppose u changed from 6.0 to 7.0**

Mvn versions:commit------then the backup version of 6.0 is removed but the pom.xml will remain same as 7.0 it will not change as 6.0

Removes the initial backup of the pom, thereby accepting the changes.

The main diff b/w commit and

revert is revert will remove backup and do changes in pom(**into older version which u have taken backup**)

commit is will remove backup and donot changes in pom(it will have new edited version only)

**Creating multi-projects:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| <https://www.baeldung.com/maven-multi-module>   |  |  | | --- | --- | |  | mvn archetype:generate -DgroupId=org.baeldung -DartifactId=parent-project |   Once the parent is generated, we have to open the pom.xml file located in the parent’s directory and change the packaging to pom.   |  |  | | --- | --- | | 1 | <packaging>pom</packaging> | |

### 6.2. Creating Submodules

As our parent POM was named parent-project, we need to make sure we’re in the parent’s directory and run generatecommands:

|  |  |
| --- | --- |
| 1  2  3 | cd parent-project  mvn archetype:generate -DgroupId=org.baeldung  -DartifactId=core  mvn archetype:generate -DgroupId=org.baeldung  -DartifactId=service  mvn archetype:generate -DgroupId=org.baeldung  -DartifactId=webapp |

After that, Maven will generate three submodules and modify for us the parent’s *pom.xml* file by adding some tags:

|  |  |
| --- | --- |
| 1  2  3  4 | <modules>      <module>core</module>      <module>service</module>      <module>webapp</module>  </modules> |

After all, if we wish to share all the configuration with our submodules, in heir *pom.xml* files, we’ll have to declare the parent:

|  |  |
| --- | --- |
| 1  2  3  4 | <parent>      <groupId>org.baeldung</groupId>      <artifactId>parent-project</artifactId>      <version>1.0-SNAPSHOT</version>  </parent> |

**Mvn package -pl modulename--------it will build specific project(module)**

**Mvn pakage -am -------------------------** If project list is specified, also build projects required by the list

clean:  *removes files generated at build-time in a project's directory* (target by default)

install: *installs the package into the****local****repository, for use as a dependency in other projects locally*

deploy: *copies the final package to the****remote****repository for sharing with other developers and projects*

mvn release: This is not a valid phase nor a goal so this won't do anything. But if refers to the [Maven Release Plugin](http://maven.apache.org/plugins/maven-release-plugin/) that is used to automate release management. Releasing a project is done in two steps: prepare and perform

mvn install -X------------it will debug output

mvn install -q------------it will show only errors

mvn install -f the path of pom----so that it will take that path and execute the build

unit testing------ suppose if developer commit the code so the developer will check whether that code is correct or not

integration testing------in unit testing he changes some code and he performs unit testing that the required changes is suitable for releated modules or not

smoke test-----it means critical functional their will test i.e before going to release they will check critical functionalites checking and at the last they will validate

**SYSTEM TESTING** is a level of software **testing** where a complete and integrated software is tested

Regression testing: check in my book

Perforanmce testing-----suppose if the application is developed with a performance ogf 10 people at a time…so we will check if 10 people will access then what is that performance…it u less than then what is the performance

Beta testing: check in my book