Summary

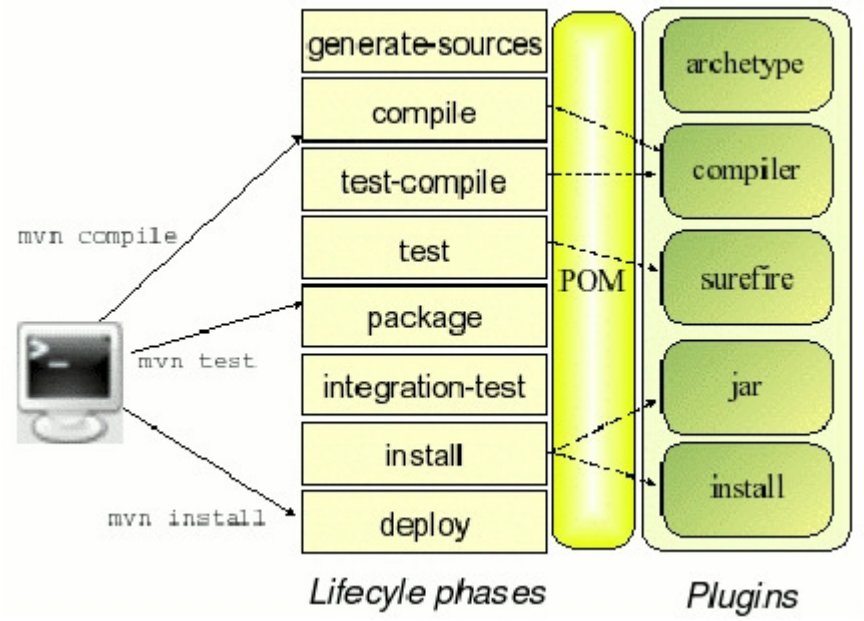
The Maven build follows the core build life cycle concept of the software project and defines the life cycle from build initialization to artifact distribution.

Description

The Maven life cycle phase is bound with plug-ins and the plug-ins execute commands. According to the life cycle, builds are carried out sequentially to carry out the command executed by the developer.

Example)When themvn install command is executed, the install command is executed by using compile and testcommands from the generate-sources phase.

생명주기 단계	설명
validate	현재 설정과 POM의 내용이 유효한지 확인
generate-sources	코드 생성기가 이 다음의 단계들에서 컴파일 되고 처리할 소스 코드를 생성하기 시작하는 순간
compile	소스 코드를 컴파일 한다. 컴파일 된 클래스들은 타깃 디렉터리 트리 구조에 저장된다.
test	컴파일 된 단위 테스트를 실행하고 그 결과를 표시한다.
package	실행 가능한 바이너리 파일들을 WAR나 JAR 같은 배포용 압축 파일로 묶는다.
install	압축 파일을 로컬 메이븐 저장소에 추가한다.
deploy	압축 파일을 원격 메이븐 저장소에 추가한다.



Example) Execute the Java compile: $mvncompilecommand.

Maven 2Basic Life Cycle Phase

life cycle phase	Description
validate	Check if the current settings and POM are valid.The process verifiesPOM.XMLfile tree structure.
initialize	The moment when initialization is allowed before main tasks to be carried out in the build cycle
generate-sources	The moment when the code generator is complied in the next phases to generate source codes
process-sources	Provide the parsing, modification and change of sources.Both normal and generated codes are handled here.
generate-resources	The phase where resources are generated.Usually meta data files and config files are included.
process-resources	The resource files are handled, instead of the source codes in the previous phase.Resource files are modified, changed and rearranged.
compile	The source files are compiled.The compiled classes are stored in the target directory tree structure.
process-classes	Class file conversion and revision phase are handled.Byte code weaver and instrumenttool are operated
generate-test-sources	The moment Mojo operates, which creates the unit test codes.
process-test-sources	Carry out the tasks required for test source code before compiling. In this phase, the source code can be modified, converted and copied.
generate-test-resources	The creation of rest related resource is allowed.
process-test-resources	The handling, conversion and rearrangement related with test are allowed.
test-compile	The unit test source code is compiled.
test	Execute the compiled unit test and display the results.
package	Use a compression file such as JAR and WAR to bundle executable binary files.
pre-integration-test	Prepare the integrated test which is to test the code in the actual deployment environment.In this phase, the above bundled compressed files can be deployed in the server
integration-test	Carry out the actual integrated test.
post-integration-test	Disable the preparation status of integrated test.This can include the reset or re-initialization of the test environment.
verify	Verify the integrity and validity of deployable compressed files. After this process, compressed files are installed
install	Add compressed files to the local maven directory ,which makes the user to be able to use other modules that depends on it.
deploy	Add compressed files to the remote maven directory, which allow more users to be able to use this artifact.

Environment configuration

Basically, the Maven plug-in is included in the Maven distribution file, requiring no additional Maven plug-ins .
If you have to use a plug-in which is not provided, check the Maven plug-in list [http://maven.apache.org/plugins/]and add it.

Manual

1.To carry out all build operations defined in the life cycle phase, execute the deploycommand, the last phase.

The deploy command is executed after validatephase and then installphase are carried out.

```
mvn deploy
```

2. Two or more maven commands can be executed sequentially. The following command is usually used to delete the contentbelow the Target directory and carry out a new build in order to distribute the package into the local repository.

```
mvn clean install
```

References

Maven Build Lifecycle [http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html]