**Maven**

A Build Lifecycle is a well-defined sequence of phases, which define the order in which the goals are to be executed. Here phase represents a stage in life-cycle. As an example, a typical Maven Build Lifecycle consists of the following sequence of phases.

Phase Handles Description prepare resources resource copying Resource copying can be customized in this phase. validate Validating the information Validates if the project is correct and if all necessary information is available. compile compilation Source code compilation is done in this phase.

Testing Tests the compiled source code suitable for testing framework. package packaging This phase creates the JAR/WAR package as mentioned in the packaging in POM.xml. This phase installs the package in local/remote maven repository. Deploy Deploying Copies the final package to the remote repository. There are always pre and post phases to register goals, which must run prior to, or after a particular phase. When Maven starts building a project, it steps through a defined sequence of phases and executes goals, which are registered with each phase.

• Dependency management including automatic updating.

• A large and growing repository of libraries.

• Extensible, with the ability to easily write plugins in java or scripting languages.

• Instant access to new features with little or no extra configuration.

• Model-based builds: Maven is able to build any number of projects into predefined output types such as jar, war, metadata.

• Coherent site of project information: Using the same metadata as per the build process, maven is able to generate a website and a PDF including complete documentation.

• Release management and distribution publication: Without additional configuration, maven will integrate with your source control system such as CVS and manages the release of a project. A Build Lifecycle is a well-defined sequence of phases, which define the order in which the goals are to be executed. Here phase represents a stage in life cycle. As an example, a typical Maven Build Lifecycle consists of the following sequence of phases. Phase Handles Description prepare resources resource copying Resource copying can be customized in this phase. validate Validating the information Validates if the project is correct and if all necessary information is available. compile compilation Source code compilation is done in this phase. Deploy Deploying Copies the final package to the remote repository. There are always pre and post phases to register goals, which must run prior to, or after a particular phase. When Maven starts building a project, it steps through a defined sequence of phases and executes goals, which are registered with each phase.

APACHE MAVEN-POM

POM stands for Project Object Model. It is fundamental unit of work in Maven. It is an XML file that resides in the base directory of the project as pom.xml. The POM contains information about the project and various configuration detail used by Maven to build the project(s). POM also contains the goals and plugins. While executing a task or goal, Maven looks for the POM in the current directory. It reads the POM, gets the needed configuration information, and then executes the goal. Some of the configuration that can be specified in the POM are following:

• project dependencies

• plugin

* Adding functionality to Maven is done through the plugin mechanism.

• goals

• build profiles

• project version

• developers

• mailing list Before creating a POM, we should first decide the project group (groupId), its name (artifactId) and its version as these attributes help in uniquely identifying the project in repository.

• All POM files require the project element and three mandatory fields: groupId, artifactId, version.

• Projects notation in repository is groupId:artifactId:version.

APACHE MAVEN-BUILD LIFECYCLE

A Build Lifecycle is a well-defined sequence of phases, which define the order in which the goals are to be executed. Here phase represents a stage in life cycle. As an example, a typical Maven Build Lifecycle consists of the following sequence of phases.

prepare resources: Resource copying can be customized in this phase.

Validate: Validating the information. Validates if the project is correct and if all necessary information is available. compile :compilation Source code, compilation is done in this phase.

Test: Testing Tests the compiled source code suitable for testing framework. Package: packaging This phase creates the JAR/WAR package as mentioned in the packaging in POM.xml.

install: installation This phase installs the package in local/remote maven repository. Deploy: Deploying Copies the final package to the remote repository. There are always pre and post phases to register goals, which must run prior to, or after a particular phase. When Maven starts building a project, it steps through a defined sequence of phases and executes goals, which are registered with each phase.

Maven has the following three standard lifecycles:

• clean

• default(or build)

• site

A goal represents a specific task which contributes to the building and managing of a project. It may be bound to zero or more build phases. A goal not bound to any build phase could be executed outside of the build lifecycle by direct invocation. The order of execution depends on the order in which the goal(s) and the build phase(s) are invoked. For example, consider the command below. The clean and package arguments are build phases while the dependency: copy-dependencies is a goal. mvn clean dependency: copy-dependencies package Here the clean phase will be executed first, followed by the dependency: copy dependencies goal, and finally package phase will be executed. Clean Lifecycle When we execute mvn post-clean command, Maven invokes the clean lifecycle consisting of the following phases.

• pre-clean

• clean

• post-clean

Maven clean goal (clean: clean) is bound to the clean phase in the clean lifecycle.

Its clean: clean goal deletes the output of a build by deleting the build directory. Thus, when mvn clean command executes, Maven deletes the build directory. We can customize this behaviour by mentioning goals in any of the above phases of clean life cycle. In the following example, We'll attach maven-antrun-plugin:run goal to the pre-clean, clean, and post-clean phases. This will allow us to echo text messages displaying the phases of the clean lifecycle.

**UNIT TESTING:**

The Maven team believes that unit testing can be a very useful tool in improving the quality of software, in particular to enforce a loosely coupled design and to give the ability to regression test the code over time. For this reason, unit test support (provided by JUnit in Java) is included out of the box, and an integral part of the build process.

**Maven objectives**

Maven's primary goal is to allow a developer to comprehend the complete state of a development effort in the shortest period of time. In order to attain this goal there are several areas of concern that Maven attempts to deal with: • Making the build process easy

• Providing a uniform build system

• Providing quality project information

• Providing guidelines for best practices development

• Allowing transparent migration to new features.