Maven Fundamentals

What is Maven?

It’s a simple Build tool.

* It produces a artifact as output(i.e. jar/war/ear)
* Manage Dependencies

Project management tool

* Handles versioning/Releases
* Describes Project
* Produce Javadocs/Site Information

Who owns Maven?

* Apache Software foundation
* Maven is built with Maven
* Its open source

Why to use maven?

* Repeatable builds
* Transitive dependencies
* It contains everything it needs for an environment
* Local repo
* IDE and standalone
* Preferred method

Ant vs Maven

Ant

* Replacement for Make
* Cross platform
* Java and XML based

With ant you have to explicitly code everything, from targets, goals and everything.

Ant leads to lot of variations that might be problematic.

Maven

* Maven is full featured
* Implicit
* Consistency
* Inheritance
* Transitive dependencies
* Versioned

Pros and cons of Maven and Ant

**Maven Ant**

Black box Easily trace

Steeper learning curve Quick to learn

Convention over configuration Copy and paste

IDE project size(too big)

Less Overhead

Mindset

For java10+ applications, add a maven compiler plugin along with release in its configuration for its proper functioning.

Maven Goals

* Clean – cleans the target folder
* Compile – fetches all the properties file and .class files within target folder
* Package – packages the application as jar, war or ear
* Install – this internally first executes package phase and then install the artifact to local repository (i.e., .m2)
* Deploy – this phase first runs install command and then executes the deploy command, by deploying the artifact in company’s configured repository or remote repositories.

How to override the defaults?

* We need to use the build section to override things of a specific features.

Transitive Dependencies

* If we add a dependency, then automatically all the dependent dependencies and libraries needed for the main dependency will be downloaded automatically.

Maven Scopes

* Compile (default scope)
* Provided – means the artifact will be available through the build cycle.
* Runtime – dynamically loaded dependencies like jdbc, should be available throughout
* Test – it is used during test compilation phase
* System – (recommended not to use) as it is brittle and can break easily. Basically it hardcodes a path to a jar on your file system.
* Import – used in dependency management

Repository types

* Local repository
* Central Repository
* Remote Repository

Repositories

* http locations
* Super pom.xml
  + Default Maven Installation
* Default location
  + Repo.maven.apache.org
* Multiple repositories allowed
* Corporate Repository
  + Nexus (this is what central is)
  + Artifactory
* Dependencies repository
  + Snapshots
  + Releases
* Plugin repository

Usually, the snapshots and releases are not kept under the same repository.

Dependency and plugin repositories can be kept same.

Plugins

* Goals
  + Default goals
    - Clean, compile, test, package, install, deploy
  + Super pom has goals defined
    - Inherently added to the effective pom
* Phases
  + Validate- validate project and structure
  + Compile – its like compiling any source in the project
  + Test – Tests the compiled code
  + Package – packages the code in the specified package type
  + Integration-test – Deploy and run integration tests(newly added, not widely used yet)
  + Verify – Run checks to verify integrity
  + Install – install package in local repo
  + Deploy – Copy package to remote repo
* Compiler Plugin
  + Plugin which is used to compile our source code
  + Invokes javac
  + Defaults older
  + This plugin can be customized by defining the configuration section
  + Fork – to spin off on its own thread
  + Memory – by setting min and max memory
  + Source/target
* Jar plugin
  + Package - Its used to package our code into a jar file, its output result will make a JAR
  + Usually ties to package phase
  + Can change includes/excludes
  + It can also make an manifest file
* Source Plugin
  + Packages the source code
  + Used in package phase
    - Overridden to later phase
* Javadoc Plugin
  + This plugin is almost identical to source plugin, that we’re going to take our javadocs and patch them into a JAR file.
  + Package javadocs
  + Package phase
    - Overridden to later phase
  + Defaults
    - Many customization options