FIS

BUILD AUTOMATION WITH MAVEN

By Satya

Agenda:

- Introduction
- Installation
- Content – pom.xml
- Examples

What is Maven ?

- Maven is a Java based build automation tool from Apache.
- Maven is a software project management and comprehension tool.
- It’s based on the concept of a project object model (POM)
- Maven can manage a project's build, reporting and documentation from a central piece of information.
- It handles java based projects.

Advantages of Using Maven over Ant

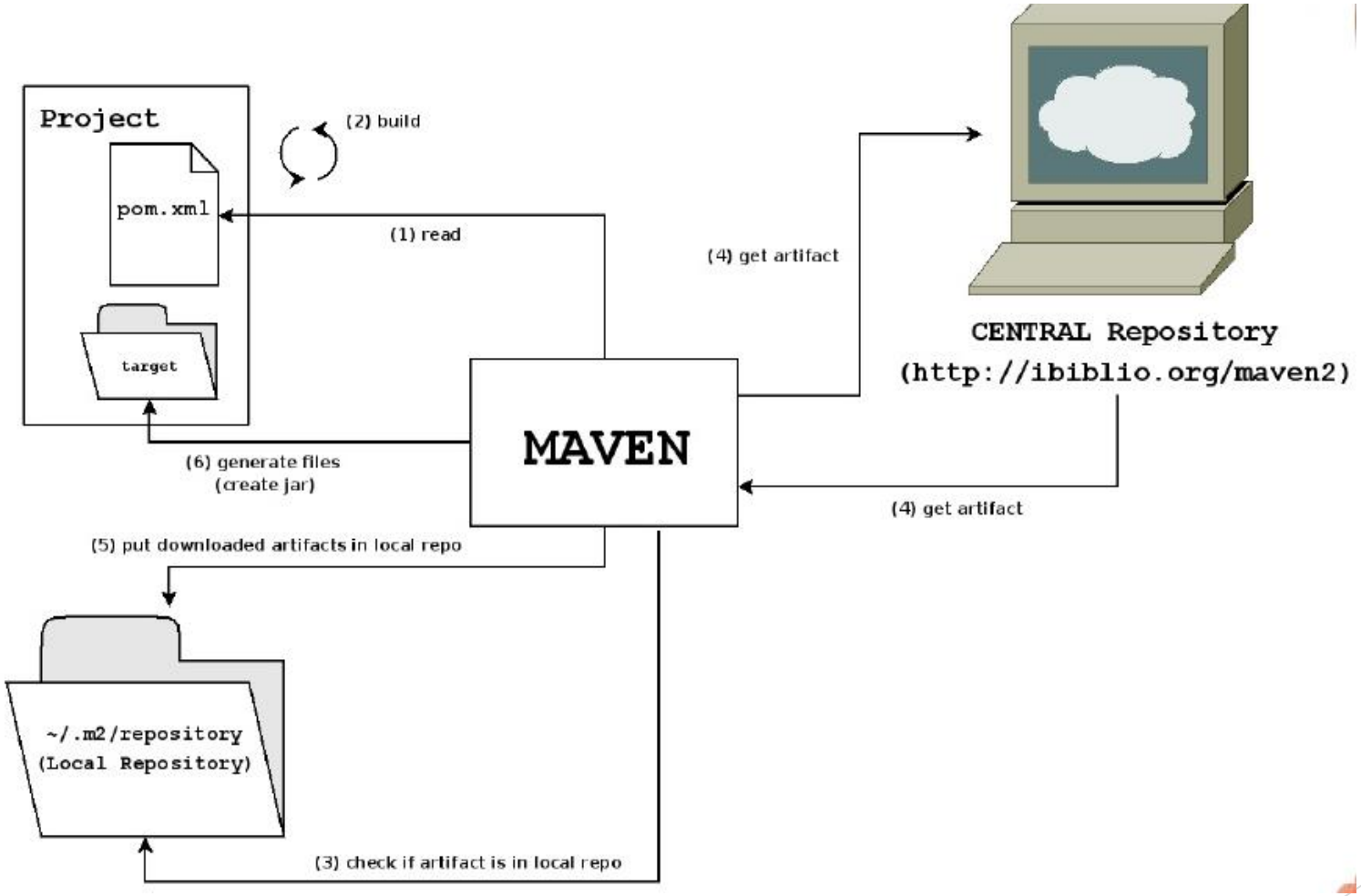
- Better dependency management
- More powerful builds
- Better debugging
- Better collaboration
- Reduced duplication
- More consistent project structure

Maven Installation

- **Pre-requisite** : Install JDK and set JAVA_HOME environment variable.
- Download Maven Archive from https://maven.apache.org/download.cgi
- Extract the Maven Archive and set the M2_HOME environment variable and Add Maven bin Directory Location to System Path
- Verify Maven Installation using below command.

  mvn --version

How maven works ?



What is pom.xml?

- A Project Object Model or **POM** is the fundamental unit of work in Maven.
- It is an **XML** file that contains information about the project and configuration details used by Maven to build the project. It contains default values for most projects.

Simple pom.xml :

```
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mycompany.app</groupId>
    <artifactId>my-app</artifactId>
    <version>1.0.0</version>
</project>
```

Maven settings.xml

- There are two locations where a settings.xml file may live:
  - ✓ The Maven install: ${maven.home}/conf/settings.xml
  - ✓ A user's install: ${user.home}/.m2/settings.xml

```
<settings>
    <servers/>
    <mirrors/>
    <localRepository/>
    <pluginGroups/>
    <proxies/>
    <profiles/>
    <activeProfiles/>
</settings>
```

Maven project creation

- We can create maven project in 2 ways.
  1. Archetypes
  2. Manual creation
- **Archetypes:**
  - Archetype is a Maven plugin whose task is to create a project structure as per its template.
    - Archetype is a Maven project templating toolkit.
    - An archetype is defined as an original pattern or model from which all other things of the same kind are made.
    - To create a new project based on an Archetype, we need to run below command

      mvn archetype:generate

Build Lifecycle and Phases

- A Build Lifecycle is a well-defined sequence of phases, which define the order of goals to be executed. Here phase represents a stage in life cycle.
- A Maven phase is nothing but a stage in the Maven build life cycle. Each phase executes a specific task.
- Here are a few important phases in the default build life cycle

  - **validate** – This phase checks if all information necessary for the build is available
  - **compile** – This phase compiles the source code
  - **test-compile** – This phase compiles the test source code
  - **test** – This phase runs unit tests
  - **package** – This phase packages compiled source code into the distributable format (jar, war)

- **integration-test** – This phase processes and deploys the package if needed to run integration tests
- **install** – This phase installs the package to a local repository
- **deploy** – This phase copies the package to the remote repository

- Maven executes phases in a specific order.
- **Maven Goals**

  A sequence of goals constitutes a phase and each goal executes a specific task. When you run a phase, then Maven executes all the goals in an order that are associated with that phase.

  compiler:compile – compile phase

  compiler:test – test-compile phase

  surefire:test – test phase

  install:install – install phase

  jar and war:war – package phase

Clean Lifecycle

- Maven Clean Lifecycle have the following phases.

  pre-clean

  clean

  post-clean

ex: mvn clean

This goal 'cleans' the project's build (usually 'target') directory, which typically involves deleting old files.

Default Lifecycle

- Maven Default Lifecycle have the following phases.
  - validate
  - compile
  - test
  - package
  - verify
  - install
  - deploy
- Example commands
  - mvn package
  - mvn clean install
  - mvn clean install –Dmaven.test.skip=true

Site Lifecycle

- Maven Site is generally used for the documentation to create reports.
- It has the following phases −

  pre-site

  site

  post-site

  site-deploy

ex: mvn site

Dependency management

- Dependency management is a core feature of Maven.
- The dependency management is a mechanism for centralizing dependency information.
- When you have a set of projects that inherit from a common parent, it's possible to put all information about the dependency in the common POM and have simpler references to the artifacts in the child POMs.

```
<dependencyManagement>
  <dependencies>
   <dependency>
    <groupId>test</groupId>
    <artifactId>a</artifactId>
    <version>1.0</version>
   </dependency>
  </dependencies>
</dependencyManagement>
```

Repositories in Maven

- In Maven terminology, a repository is a directory where all the project jars, plugins or any other project specific artifacts are stored and can be used by Maven easily.
- Maven repository are of three types.

  Local (.m2 folder in local)

  Central (Repository provided by Maven community)

  Remote (Company's repository)

Artifact Versioning

- There are 2 types of versions.
- **SNAPSHOTS** --- Used by projects during project development as it implies that development is still occurring and that project may change
- **RELEASE** --- A version that is assumed never to change. Only to be used for a single state of the project when it released and then updated to the next snapshot version.

- Ex: 1.0.0-SNAPSHOT
- 1.0.0
- 1.0.1-SNAPSHOT
-

Distribution management

- It manages the distribution of the artifact and supporting files generated throughout the build process

```
<distributionManagement>
    <repository>
        <id>releases</id>
        <name>project-release-local</name>
        <url>https://release repo url</url>
    </repository>
    <snapshotRepository>
        <id>snapshots</id>
        <name>project-snapshot-local</name>
        <url>https://snapshot repo url</url>
    </snapshotRepository>
</distributionManagement>
```

Maven plugins

- Maven is actually a plugin execution framework where every task is actually done by plugins.
- We have 2 different types of plugins.

  - **Build plugins -** They execute during the build process and should be configured in the <build/> element of pom.xml.

  - **Reporting plugins -** They execute during the site generation process and they should be configured in the <reporting/> element of the pom.xml.

  - **List of few common plugins**:
  - Clean, Compiler, Surefire
  - Jar, War, Javadoc
  - Antrun, Release

War Plugin snippet:

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-war-plugin</artifactId>
            <version>2.3</version>
            <configuration>
                <failOnMissingWebXml>false</failOnMissingWebXml>
            </configuration>
        </plugin>
          ...
    </plugins>
</build>
```

Maven Release plugin

- This plugin is used to release a project with Maven, saving a lot of repetitive, manual work.
- Releasing a project is made in two steps: prepare and perform.
  - mvn release:prepare
  - mvn release:perform
  - mvn release:rollback
- Usage : there are 2 things you should include in our pom:
  - the scm-section with a developerConnection
  - the maven-release-plugin with a locked version

```
<scm>
<developerConnection>scm:svn:https://svn.mycompany.com/repos/myapplication/trunk/mycomponent/</developerConnection>
</scm>


<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-release-plugin</artifactId>
<version>3.0.0-M1</version>
</plugin>
```

Q&A