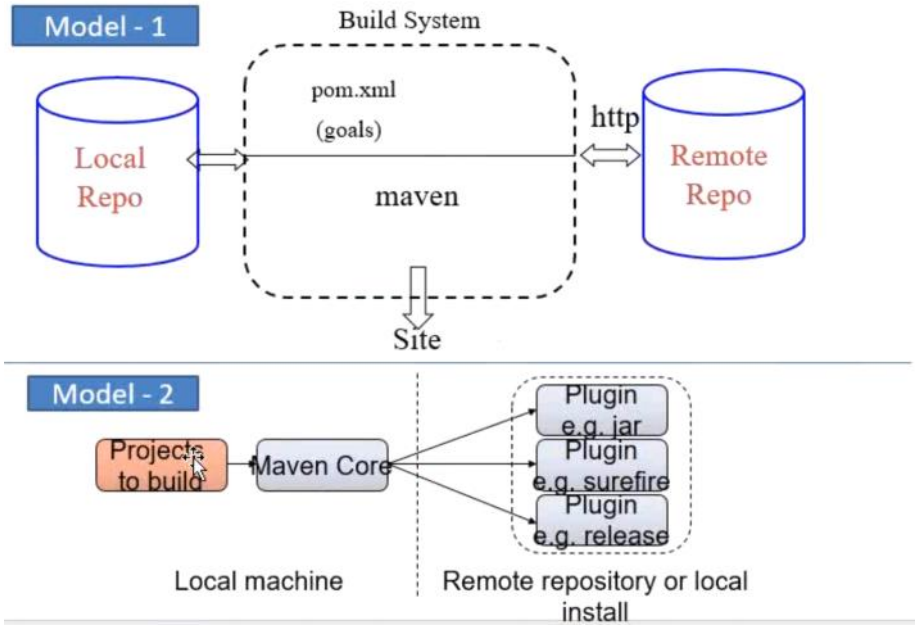
Architecture

Thursday, June 7, 2018 12:42 PM



pom - Build or Configuration file
- Details of the project

MAVEN BUILD LIFECYCLE

- Generate Sources / resources
  - List of required plugins
  - Libr
    - Looks for the resources in local repo
    - Not found
      - install the resources from remote repo
        - remote repos
          - configured to check from a specific site

- Compile
  - test - written by developer to run the

- compile
- test: - Written by developer to run the Unit Test
- package : Creating jar, war, ear files
- Integration - test (pre & post) ??? - Very specific to certain things - UNCLEAR
- Install - Dependancy of other projects
  - When required to run the project other project files are
  - Maven looks in the local and remote
  - These project files are installed in the local repo for maven to find.
- Deploy - Store it like a artifact (or) Put into some application server like Tomcat

→ Call any of these default it will run from the top step

→ Clean, Site - Not default life cycles.

→ Clean as namespace
- Delete (or) Clean the entire project directory
- All the runtime files Created will be deleted

Runtime files
- compilation - object/Class files

Runtime files
- compilation - object/Class files
- Unit Test - generate log/reports
- package - Creates jar/war/ear files

Site
- Create the documentation whenever it is needed

Example Maven Goals

- To invoke a Maven build you set a lifecycle "goal"
- mvn install
  - Invokes generate* and compile, test, package, integration-test, install
- mvn clean
  - Invokes just clean
- mvn clean compile
  - Clean old builds and execute generate*, compile
- mvn compile install - Doesn't run the already done tasks
  - Invokes generate*, compile, test, integration-test, package, install
- mvn test clean
  - Invokes generate*, compile, test then cleans

STANDARD DIRECTORY LAYOUT :

src/main/java	Application/Library sources
src/main/resources	Application/Library resources
src/main/filters	Resource filter files
src/main/assembly	Assembly descriptors
src/main/config	Configuration files
src/main/scripts	Application/Library scripts
src/main/webapp	Web application sources
src/test/java	Test sources
src/test/resources	Test resources
src/test/filters	Test resource filter files
src/site	Site
LICENSE.txt	Project's license
NOTICE.txt	Notices and attributions required by libraries that the project depends on
README.txt	Project's readme

Parent project dir:

- src
  - main
    - java
    - resources

- java
- resources
- assembly
- Config

- test
  - java
  - resources
  - filters

- pom.xml

Project Name (GAV)

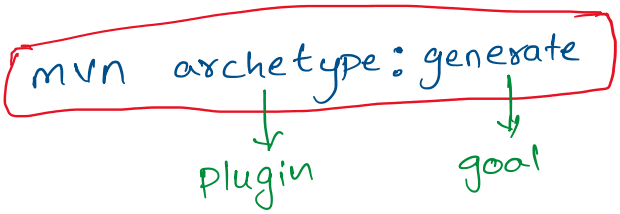
- Plugin is a jar file that maven runs internally
- GAV Syntax groupID:artifactID:version

- Maven uniquely identifies a project using:
  - groupID: Arbitrary project grouping identifier (no spaces or colons)
    - Usually loosely based on Java package
  - artfiactId: Arbitrary name of project (no spaces or colons)
  - version: Version of project
    - Format {Major}.{Minor}.{Maintenance}
    - Add '-SNAPSHOT ' to identify in development

- Each project will also have a unique GAV
- Default packaging type is jar
- Maven downloads all the plugins using the GAV reference

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
    <modelVersion>4.0.0</modelVersion>
    <artifactId>maven-training</artifactId>
    <groupId>org.lds.training</groupId>
    <version>1.0</version>
    <packaging>jar</packaging>
</project>
```

MAVEN_OPTS - Set to make sure adequate memory is available at Run time.



(quickstart)

7

groupId : - organisation name (or) business unit

artifactId : Actual product (or) project we are working

Version : 1.0-SNAPSHOT (default) - Since we are not working on specific release

Package : Name of the package/jar file.

default