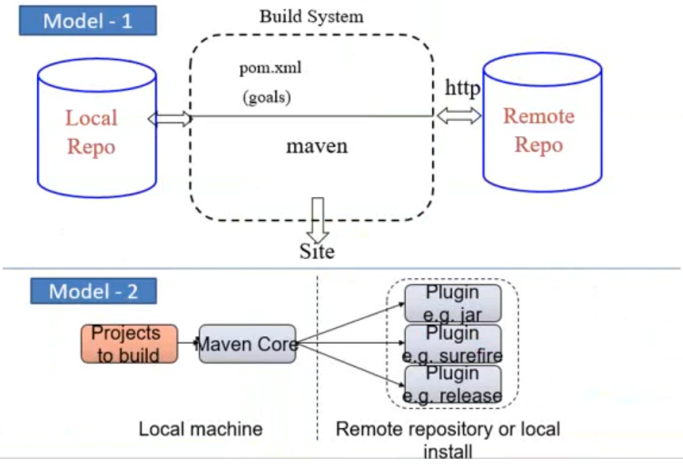
```
1.java : 1.class
2.java : 2.class  -> deliverable (jar, war, ear)
3.java : 3.class

compilation : raw source code -> object files
assembly : creating jar/war/ear

mvn -> goals -> plugins (jar) -> task

mvn <goals>
mvn compile

workspace
---------
src/main
        /java/1.java
        /resources
src/test
        /java/1_test.java
        /resources
pom.xml
```

Overview of Simple Architecture



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
  - Invokes generate*, compile, test, integration-test, package, install
- mvn test clean
  - Invokes generate*, compile, test then cleans

Project Name (GAV)

- 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
- GAV Syntax: groupId:artifactId:version
- Build type identified using the "packaging" element
- Tells Maven how to build the project
- Example packaging types:
  - pom, jar, war, ear, custom
  - Default is jar

```
<?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 Environment Setup

```
JAVA_HOME
M2_HOME
M2
MAVEN_OPTS = -Xms256m -Xmx512m
PATH=%PATH%;%M2%
```

```
mvn archetype:generate
```

Maven Repositories

- Dependencies are downloaded from repositories
  - Via http
- Downloaded dependencies are cached in a local repository
  - Usually found in ${user.home}/.m2/repository
- Repository follows a simple directory structure
  - {groupId}/{artifactId}/{version}/{artifactId}-{version}.jar
  - groupId '.' is replaced with '/'
- Maven Central is primary community repo
  - http://repo1.maven.org/maven2

POM

- What is POM?

  POM Stands for Project Object Model

  As a fundamental unit of work in Maven, POM is an XML file that contains information about project and configuration details used by Maven to build the project"
- Describes a project
  - Name and Version
  - Artifact Type
  - Source Code Locations
  - Dependencies
  - Plugins
  - Profiles (Alternate build configurations)
- Uses XML by Default
  - Not the way Ant uses XML

Maven Plugin management

- Maven is actually a plugin execution framework where every task is actually done by plugins
- A plugin generally provides a set of goals and which can be executed using following syntax:

  % mvn [plugin-name]:[goal-name]

  % mvn compiler:compiler

Plugin Types

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

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

- Plugins are specified in pom.xml using plugins element.
- Each plugin can have multiple goals.
- You can define phase from where plugin should starts its processing using its phase element. You can configure tasks to be executed by binding them to goals of plugin.
- That's it, Maven will handle the rest. It will download the plugin if not available in local repository

```
<project>
  <build>
        <plugins>
          <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-antrun-plugin</artifactId>
            <version>1.1</version>
            <executions>
              <execution>
                  <id>id.clean</id>
                  <phase>clean</phase>
                  <goals>
                  <goal>run</goal>
                  </goals>
                  <configuration>
                  <tasks>
                     <echo>clean phase</echo>
                  </tasks>
                  </configuration>
              </execution>
            </executions>
          </plugin>
        </plugins>
    </build>
</project>
```

```
<groupId>scmlearningcentre</groupId>
<artifactId>demo</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>jar</packaging>

<name>demo</name>
<url>http://maven.apache.org</url>

<build>
      <plugins>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-antrun-plugin</artifactId>
          <version>1.1</version>
          <executions>
            <execution>
                <id>id.clean</id>
                <phase>compile</phase>
                <goals>
                <goal>run</goal>
                </goals>
                <configuration>
                <tasks>
                   <echo>====================</echo>
                   <echo>  Calling ANT task   </echo>
                   <echo>====================</echo>
                </tasks>
                </configuration>
            </execution>
          </executions>
        </plugin>
      </plugins>
</build>
```

```
<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>exec-maven-plugin</artifactId>
  <version>1.2.1</version>
  <configuration>
   <executable>git</executable>
   <arguments>
    <argument>--version</argument>
   </arguments>
  </configuration>
</plugin>
```

```
<project xmlns="http://maven.apache.org/P
xsi:schemaLocation="http://maven.apache.c
   <modelVersion>4.0.0</modelVersion>
   <groupId>com.companyname.projectgroup<
   <artifactId>project</artifactId>
   <version>1.0</version>
   <profiles>
     <profile>
         <id>test</id>
         <build>
         <plugins>
       <plugin>
             <groupId>org.apache.maven.p
                 <artifactId>maven-antru
                 <version>1.1</version>
                 <executions>
                   <execution>
                        <phase>clean</ph
                        <goals>
                          <goal>run</goa
                        </goals>
                        <configuration>
```

Maven SNAPSHOTS

- A large software application generally consists of multiple modules and it is common scenario where multiple

teams are working on different modules of same application

- For ex Demo2 team uses Demo.jar
- Now if Demo team builds a new jar
  - Demo should inform every time when they release an updated code
  - Demo2 have to update their pom.xml to get the latest Demo.jar

What is SNAPSHOT?

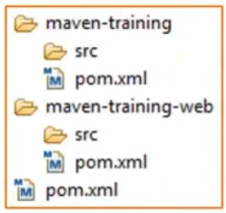
SNAPSHOT is a special version that indicates a current development copy. Unlike regular versions, Maven checks for a new SNAPSHOT version in a remote repository for every build.

Multi Module Projects

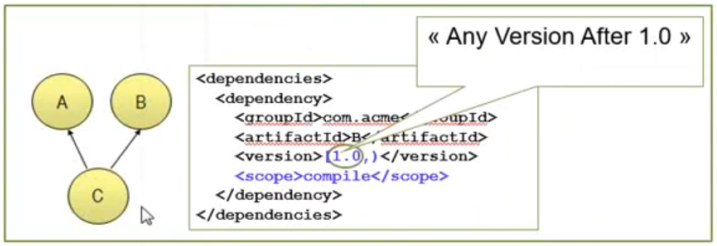
- Maven has 1st class multi-module support
- Each maven project creates 1 primary artifact
- A parent pom is used to group modules
- To run a particular module alone

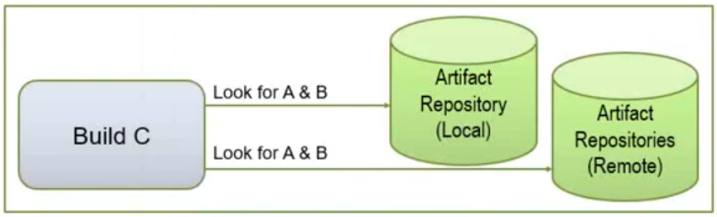
$ mvn clean –pl <modulename>

```
<project>
  <groupId>EBU</groupId>
  <artifactId>Parent-module</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>pom</packaging>
    <modules>
        <module>Child-jar</module>
        <module>child-war</module>
    </modules>
</project>
```



Dependency Management





- Lets try with adding dependency of child-jar for the child-war from the previous example

```
<dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>EBU</groupId>
      <artifactId>child-jar</artifactId>
      <version>1.0-SNAPSHOT</version>
    </dependency>
  </dependencies>
```

```
<groupId>scmlearningcentre</groupId>
<artifactId>demo</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>pom</packaging>

<name>demo</name>
<url>http://maven.apache.org</url>


<properties>
  <project.build.sourceEncoding>UTF-8</p
</properties>

<modules>
 <module>add</module>
 </module>sub</module>
</modules>

<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
</dependencies>
</project>
```

Multi Modules ..

```
<project>
…
<parent>
   <groupId>EBU</groupId>
   <artifactId>Parent-module</artifactId>
   <version>1.0-SNAPSHOT</version>
</parent>
<groupId>EBU</groupId>
<artifactId>child-jar</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>jar</packaging>
…
</project>
```

```
<project>
…
<parent>
   <groupId>EBU</groupId>
   <artifactId>Parent-module</artifactId>
   <version>1.0-SNAPSHOT</version>
</parent>
<groupId>EBU</groupId>
<artifactId>child-war</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>jar</packaging>
…
</project>
```

```
<groupId>scmlearningcentre</groupId>
<artifactId>demo</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>pom</packaging>

<name>demo</name>
<url>http://maven.apache.org</url>


<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>

<modules>
 <module>sub</module>
 <module>add</module>
 <module>div</module>
</modules>

<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
</dependencies>
</project>
```

```
Maven properties
- just as ant properties ${property_name}
- case sensitive
-- upper case for environment variables
--
-DOt(.) notated path
- built in properties
 --${basedir} - dir with pom
 -- ${version} - artifact version
-project properties
 --${project.build.directory}
 --${project.build.outputDirectory} (target/classes)
 --${project.name}
 --${project.version}

  <build>
    <sourceDirectory>${basedir}/src/main/java</sourceDirectory>
    <scriptSourceDirectory>${basedir}/src/main/scripts</scriptSourceDirectory>
    <testSourceDirectory>${basedir}/src/test/java</testSourceDirectory>
    <outputDirectory>${basedir}/target/classes</outputDirectory>
    <testOutputDirectory>${basedir}/target/test-classes</testOutputDirectory>
  </build>
```

```
maven reactor
- collects all the available modules to build
- sorts the projects into the current build order
-- a project dependecny on another module in the build
-- diff roles with plugins dependencies
-- thr order declared in the <modules> element (if no other rule applies)
-builds the selected projects in order
- be aware of cycles & same modules on diff parents
```

Deployment Automation

```
<build>
<plugins>
 <plugin>
<groupId>org.jboss.as.plugins</groupId>
<artifactId>jboss-as-maven-plugin</artifactId>
<version>7.3.Final</version>
 <configuration>
  <jbossHome>C:\Users\anmuruga\JBOSS\jboss-7.1.1.Final</jbossHome>
  <serverName>default</serverName>
  <groupId>classroom</groupId>
  <artifactId>sample</artifactId>
  <name>helloworld.war</name>
 </configuration>
</plugin>
</plugins>
</build>
```

- mvn jboss-as:deploy
- mvn jboss-as:undeploy

https://localhost:9990/console http://127.0.0.1:9990

Maven update version

- mvn versions:set -DnewVersion=your_new_version

  ex : mvn versions:set -DnewVersion=2.1.1
- mvn versions:commit
- mvn versions:revert

Documentation – Building Own Site

- mvn **site**
- pom.xml


```
<project> ...
<distributionManagement>
<site>
<id>website</id>
<url>scp://www.mycompany.com/www/docs/project/</url>
</site>
</distributionManagement> ...
</project>
```