Apache

maven

Introduction

- What is Maven?

"Maven is a project management tool which encompasses a project object model, a set of standards, a project lifecycle, a dependency management system, and logic for executing plugin goals at defined phases in a lifecycle"

- Maven provides superset of features found in a build tool
- Maven *manages project build, reporting, and documentation from a central piece of information*

Objectives and Characteristics of MAVEN

- Maven is more than just Build Tool
- Maven was built considering certain objectives
- Maven Provides:
  - Easy Build Process
  - Uniform Build System
  - Quality Project Information
  - Guidelines for Best Practices Development
- Achieved Characteristics:
  - Visibility
  - Reusability
  - Maintainability
  - Comprehensibility "Accumulator of Knowledge"

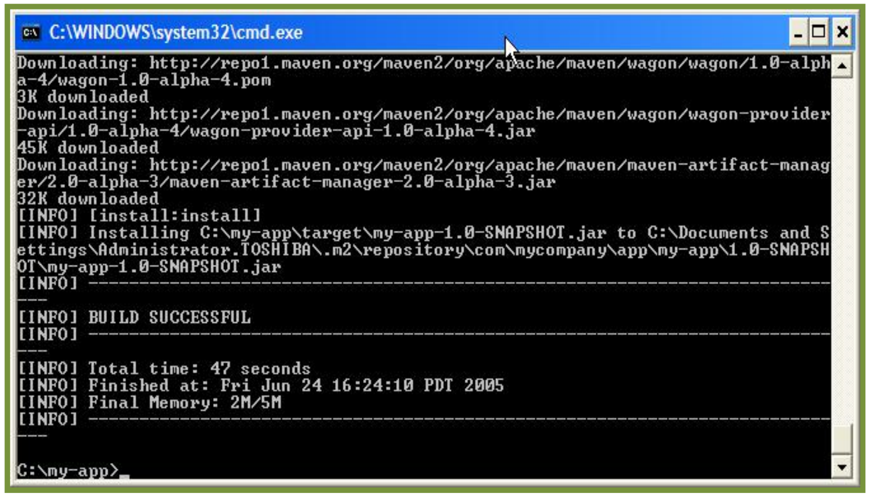
Comparison with ANT

ANT	MAVEN
Target build.xml	Goal pom.xml

1. One level above ANT
2. Higher level of reusability between builds
3. Faster turn around time to set up a powerful build
4. Project website generation
5. Less maintenance
6. Greater momentum
7. Repository management
8. Automatic downloads

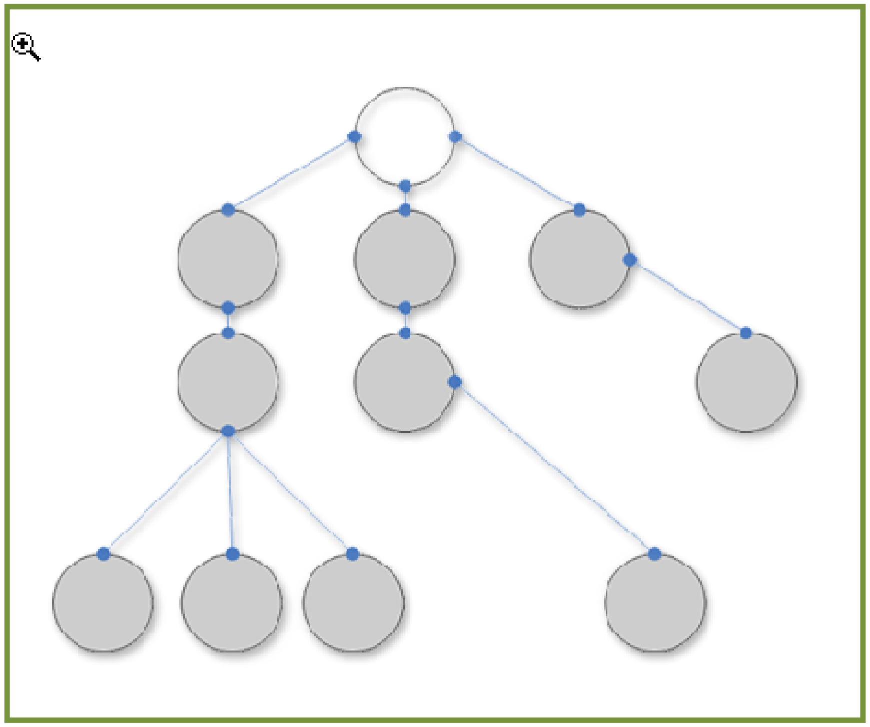
Main Features of MAVEN

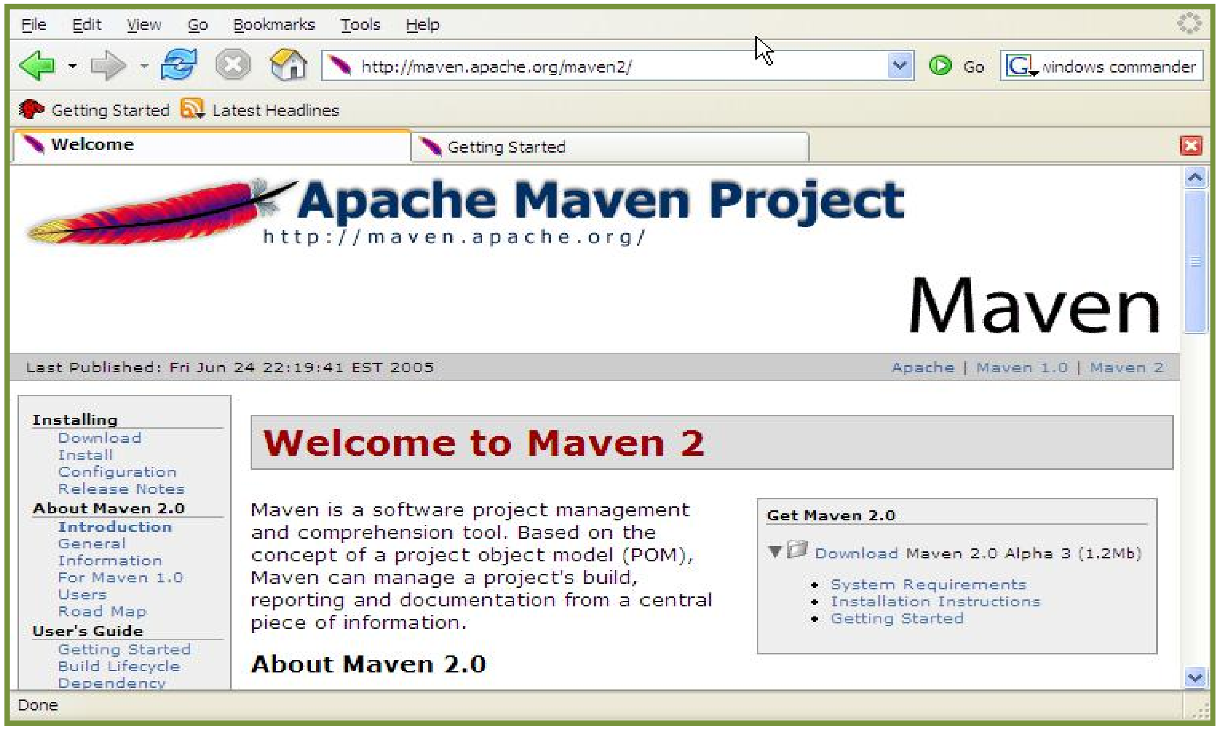
- ➢Build-Tool
- ➢Dependency Management Tool
- ➢Documentation Tool



```
Downloading: http://repo1.maven.org/maven2/org/apache/maven/wagon/wagon/1.0-alph
a-4/wagon-1.0-alpha-4.pom
3K downloaded
Downloading: http://repo1.maven.org/maven2/org/apache/maven/wagon/wagon-provider
-api/1.0-alpha-4/wagon-provider-api-1.0-alpha-4.jar
45K downloaded
Downloading: http://repo1.maven.org/maven2/org/apache/maven/maven-artifact-manag
er/2.0-alpha-3/maven-artifact-manager-2.0-alpha-3.jar
32K downloaded
[INFO] [install:install]
[INFO] Installing C:\my-app\target\my-app-1.0-SNAPSHOT.jar to C:\Documents and S
ettings\Administrator.TOSHIBA\.m2\repository\com\mycompany\app\my-app\1.0-SNAPSH
OT\my-app-1.0-SNAPSHOT.jar
[INFO] -------------------------------------------------------------------------
---
[INFO] BUILD SUCCESSFUL
[INFO] -------------------------------------------------------------------------
---
[INFO] Total time: 47 seconds
[INFO] Finished at: Fri Jun 24 16:24:10 PDT 2005
[INFO] Final Memory: 2M/5M
[INFO] -------------------------------------------------------------------------
---

C:\my-app>_
```





Last Published: Fri Jun 24 22:19:41 EST 2005

Welcome to Maven 2

Maven is a software project management and comprehension tool. 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.

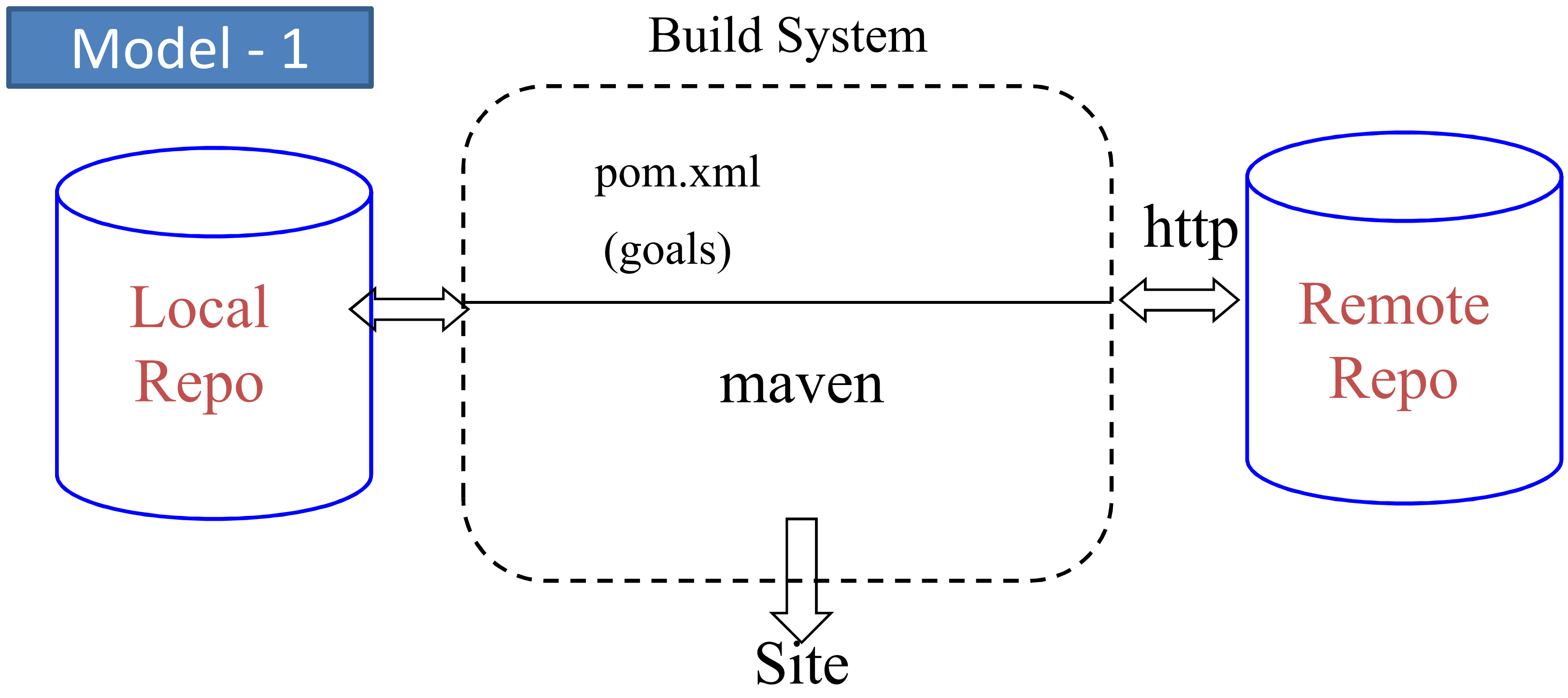
About Maven 2.0

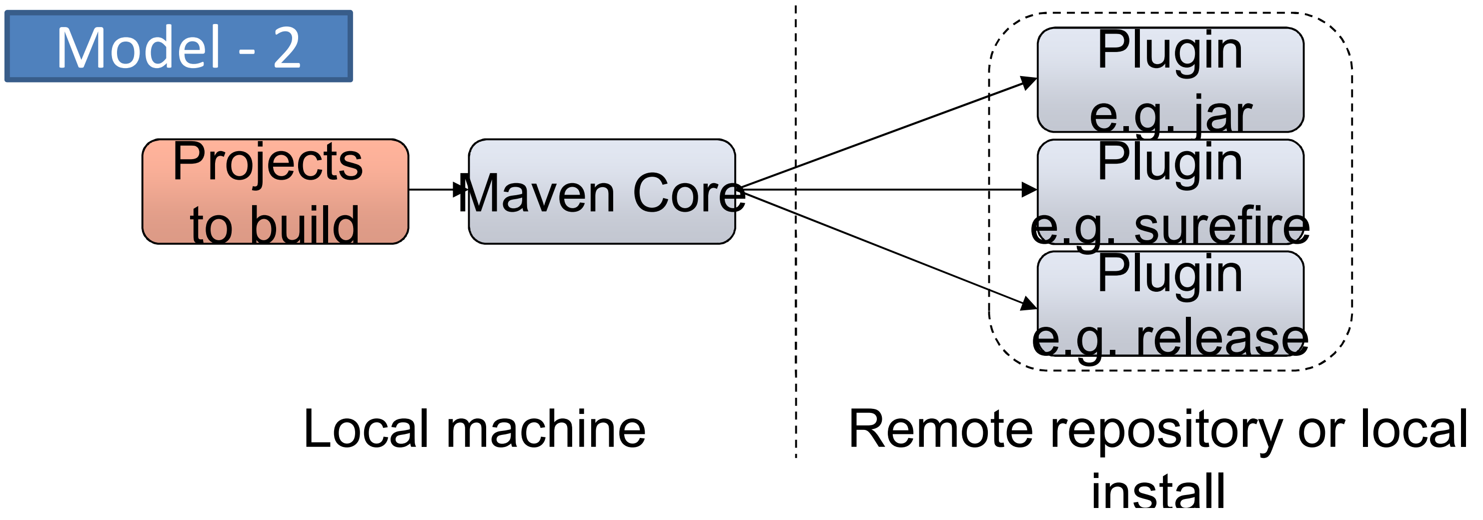
Get Maven 2.0

Download Maven 2.0 Alpha 3 (1.2Mb)

- System Requirements
- Installation Instructions
- Getting Started

Overview of Simple Architecture





Other Java Build Tools

- Ant (2000)
  - Granddaddy of Java Build Tools
  - Scripting in XML
  - Very flexible
- Ant+Ivy (2004)
  - Ant but with Dependency Management
- Gradle (2008)
  - Attempt to combine Maven structure with Groovy Scripting
  - Easily extensible
  - Immature

Maven Build Lifecycle

- A Maven build follow a lifecycle
- Default lifecycle
  - generate-sources/generate-resources
  - compile
  - test
  - package
  - integration-test (pre and post)
  - Install
  - deploy
- There is also a Clean, Site lifecycle

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

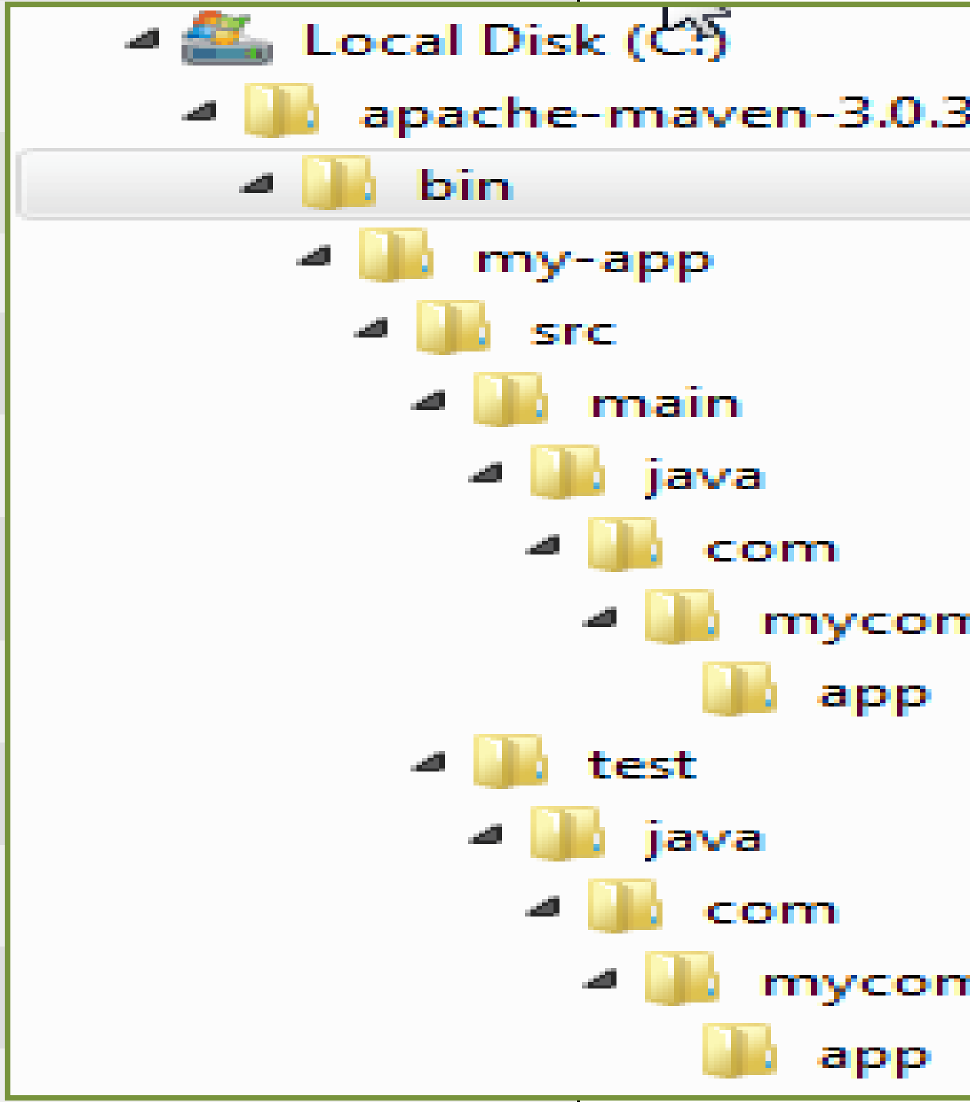
MAVEN_OPTS = *-Xms256m -Xmx512m*

PATH=%PATH%;%M2%

```
mvn archetype:generate
```

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



```
Local Disk (C:)
  apache-maven-3.0.3
    bin
      my-app
        src
          main
            java
              com
                mycom
                  app
          test
            java
              com
                mycom
                  app
```

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 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

Proxy Repositories

- Proxy Repositories are useful:
  - Organizationally cache artifacts
  - Allow organization some control over dependencies
  - Combines repositories
- Many uses the Nexus repository manager
- All artifacts in Nexus go through approval process
  - License verified
  - Improve organizational reuse

Project Creation in MAVEN

```
mvn archetype:generate

-DgroupId = com.mycompany.app

-DartifactId = my-app

-DarchetypeArtifactId = maven-archetype-quickstart

-DinteractiveMode = false
```

Project Object Model (POM)

- Metadata: Location of Directories, Developers/Contributors, Dependencies, Repositories
- Dependencies (Transitive Dependencies), Inheritance, and Aggregation
- Key Elements
  - ➢ Project
  - ➢ Model Version
  - ➢ Group ID
  - ➢ Packaging
  - ➢ Artifact ID
  - ➢ Version
  - ➢ Name
  - ➢ URL
  - ➢ Description

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

Example 1

```xml
<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>
```

Example 2

```
<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>
```

```
mvn exec:exec
```

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 everytime when they release an updated code
  - Demo2 have to update their pom.xml to get the latest Demo.jar

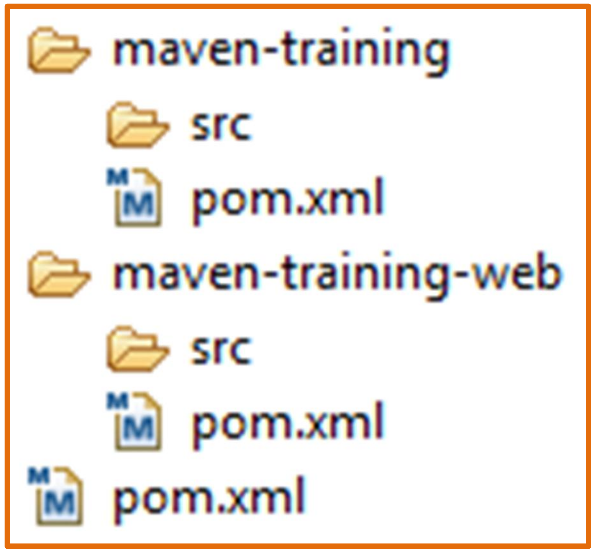
What is SNAPSHOT?

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

```xml
<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>
```

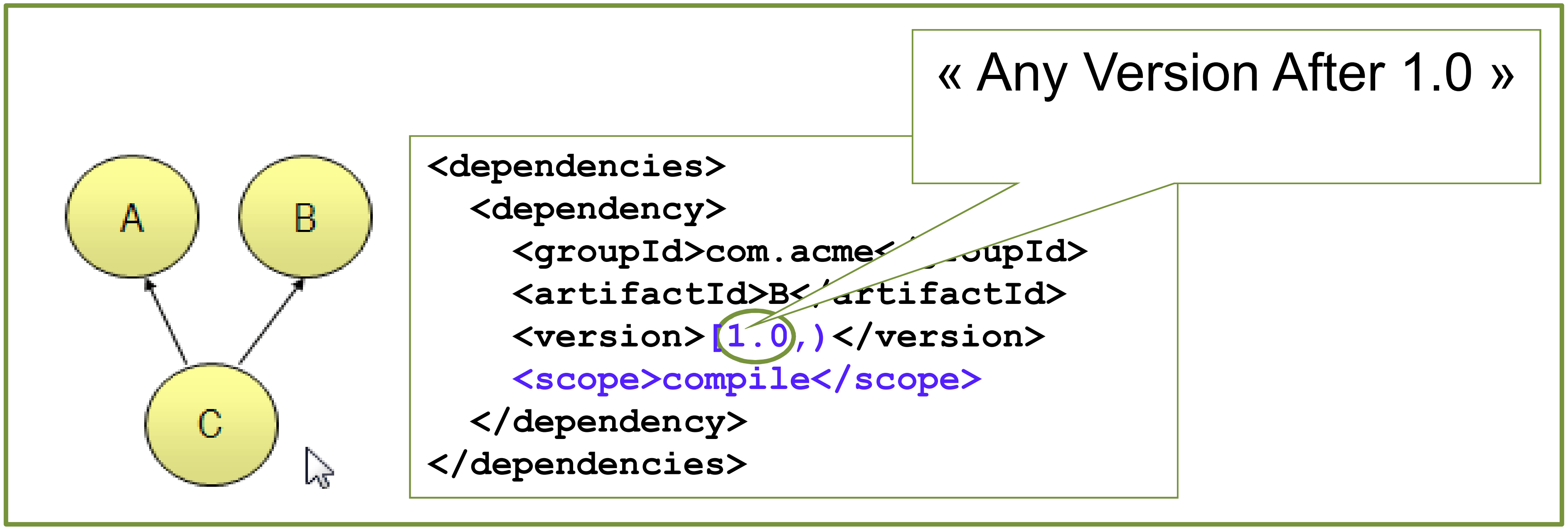


```
maven-training
    src
    pom.xml
maven-training-web
    src
    pom.xml
pom.xml
```

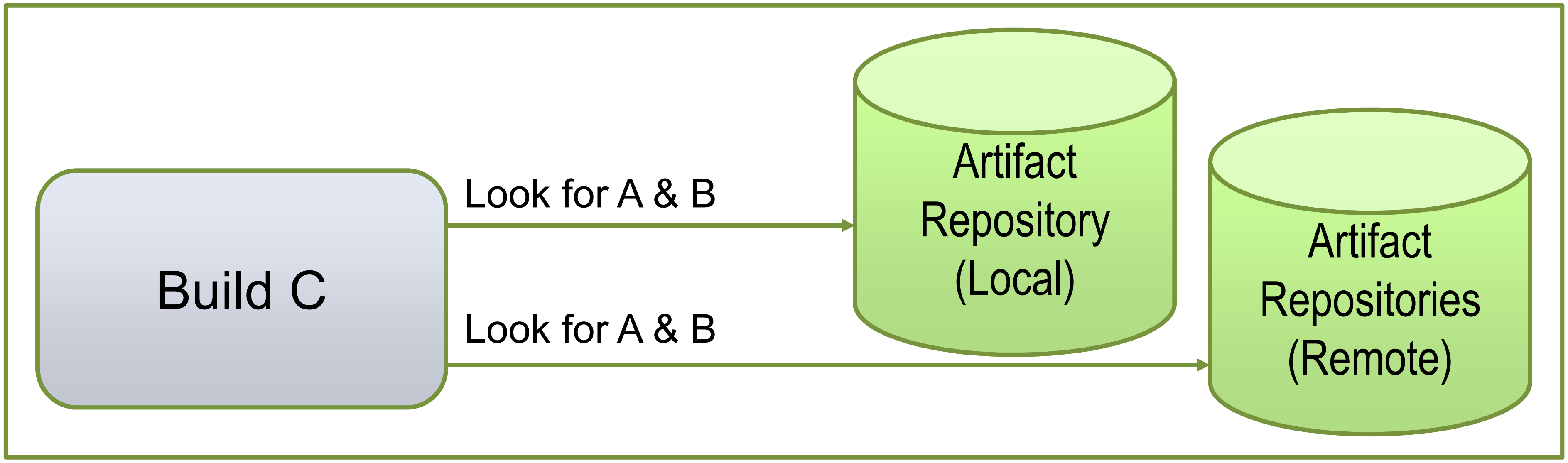
Multi Modules ..

```
<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>
```

Dependency Management

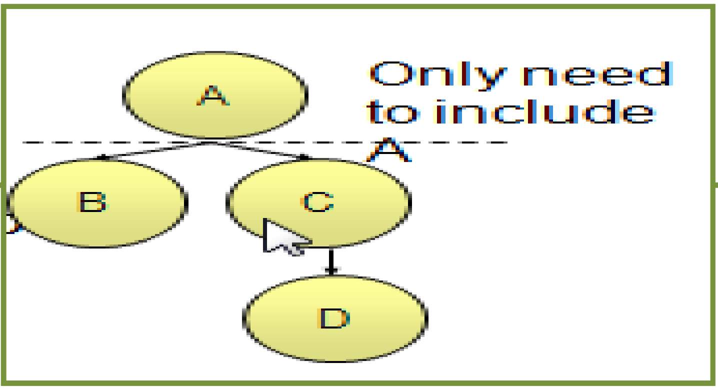


```
<dependencies>
  <dependency>
    <groupId>com.acme</groupId>
    <artifactId>B</artifactId>
    <version>[1.0,)</version>
    <scope>compile</scope>
  </dependency>
</dependencies>
```



Transitive Dependencies

- Transitive Dependency Definition:
  - A dependency that should be included when declaring project itself is a dependency
- ProjectA depends on ProjectB
- If ProjectC depends on ProjectA then ProjectB is automatically included
- Only compile and runtime scopes are transitive



- 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>
```

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

Maven SCM

```
 <scm>
    <url>https://github.com/scmlearningcentre/demo</url>
    <connection>scm:git:git://github.com/scmlearningcentre/demo.git</connection>
<developerConnection>scm:git:git@github.com:scmlearningcentre/demo.git</developerCon
nection>
 </scm>
```

- mvn scm:checkout
- mvn scm:checkin
- mvn scm:update

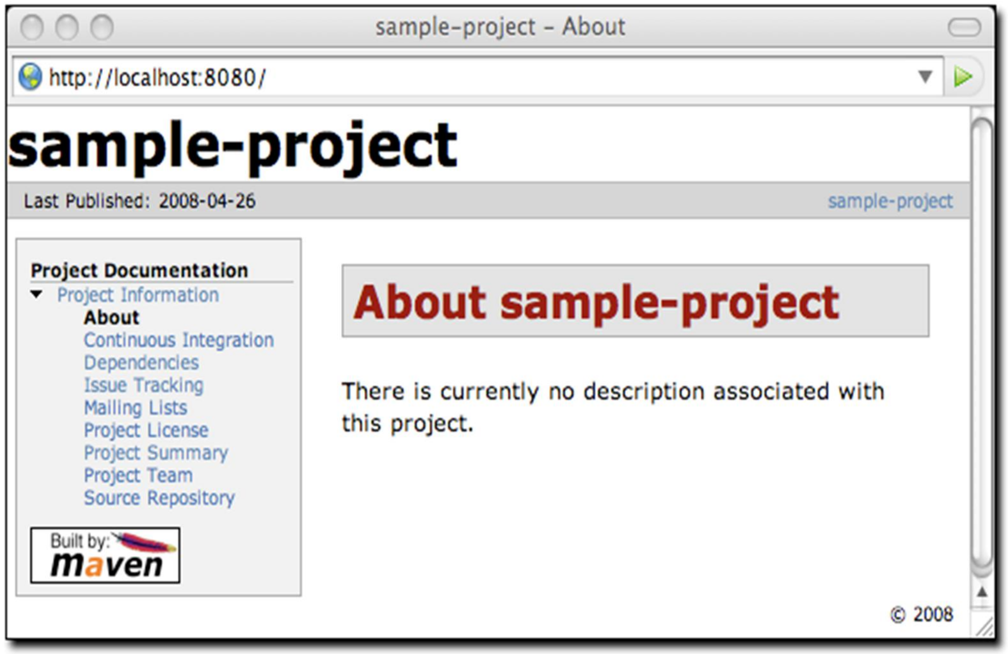
```
<distributionManagement>
    <repository>
     <id>Core-API-Java-Release</id>
     <name>Release repository</name>
     <url>http://localhost:8081/nexus/content/repositories//Core-Api-Release</url>
    </repository>
</distributionManagement>
```

- mvn deploy:deploy

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>
```



There is currently no description associated with this project.