Build Automation with Maven

Khiem Quang Tran
Software Engineer
ktran47@csc.com

Warm up - Introductions

- Your role
- Your background, experiences in the subject
- What do you want from this course

Course Objectives

- At the end of the course, you will have acquired sufficient knowledge to:
  - Using Maven with advance knowledge of configurations.

Agenda

- Introduction to Maven
- Using Maven (Getting started)
- Maven principles detail

Course Audience and Prerequisite

- The course is for JAVA developers.
- The following are prerequisites:
  - JAVA fundamentals.
  - J2EE basic experience.
  - Software development process experience.
  - No other course is required.

Assessment Disciplines

- Exercise: <50%>
- Final Exam: <50%>
- Passing Scores: <70%>

Duration and Course Timetable

- Course Duration: <3 hrs>

Further References

- Maven Home: http://maven.apache.org/
- Guides: http://maven.apache.org/guides/
- Build lifecycle: http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html
- Better Builds with Maven: http://www.maestrodev.com/better-build-maven
- Articles: http://maven.apache.org/articles.html
- …

Set Up Environment

- To complete the course, your PC must install:
  - Eclipse
  - Maven
  - **This slide** with sample codes
  - Connect to Internet or local Maven repository.

Course Administration

- In order to complete the course you must:
  - Sign in the Class Attendance List
  - Participate in the course
  - Provide your feedback in the End of Course Evaluation
  - Complete exercises and mini-test

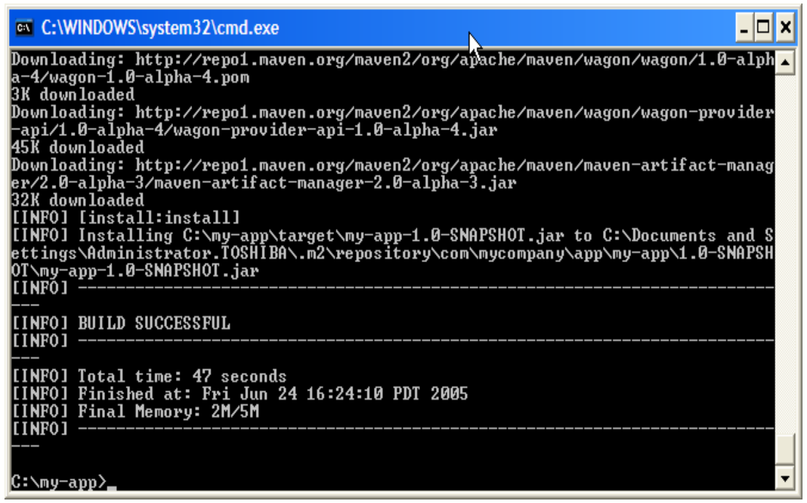
Introduction

General approach

- What is Maven?
- When we use Maven?
- Where is Maven in the development process?
- Why use Maven?
- How can Maven do that?

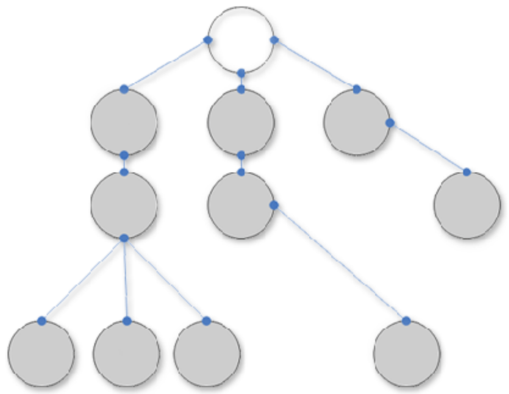
What is Maven?

A build tool

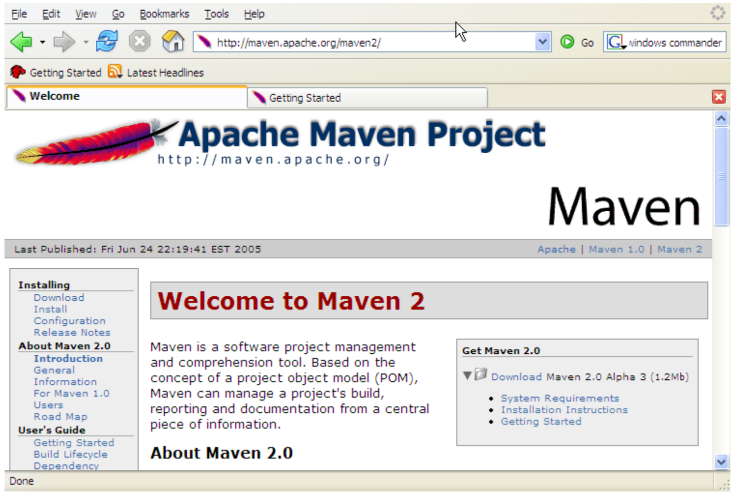


```
Downloading: http://repo1.maven.org/maven2/org/apache/maven/wagon/wagon/1.0-alph
a-4/wagon-1.0-alpha-4.pom
3K downloaded
Downloading: http://repo1.maven.org/maven2/org/apache/maven/wagon/wagon-provider
-api/1.0-alpha-4/wagon-provider-api-1.0-alpha-4.jar
45K downloaded
Downloading: http://repo1.maven.org/maven2/org/apache/maven/maven-artifact-manag
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>_
```

A dependency management tool



A documentation tool


File Edit View Go Bookmarks Tools Help
http://maven.apache.org/maven2/
Getting Started Latest Headlines
Welcome
Getting Started
Apache Maven Project
http://maven.apache.org/
Maven

Last Published: Fri Jun 24 22:19:41 EST 2005 | Apache | Maven 1.0 | Maven 2

Installing
- Download
- Install
- Configuration
- Release Notes

About Maven 2.0
- Introduction
- General Information
- For Maven 1.0 Users
- Road Map

User's Guide
- Getting Started
- Build Lifecycle
- Dependency

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.

Get Maven 2.0

Download Maven 2.0 Alpha 3 (1.2Mb)
- System Requirements
- Installation Instructions
- Getting Started

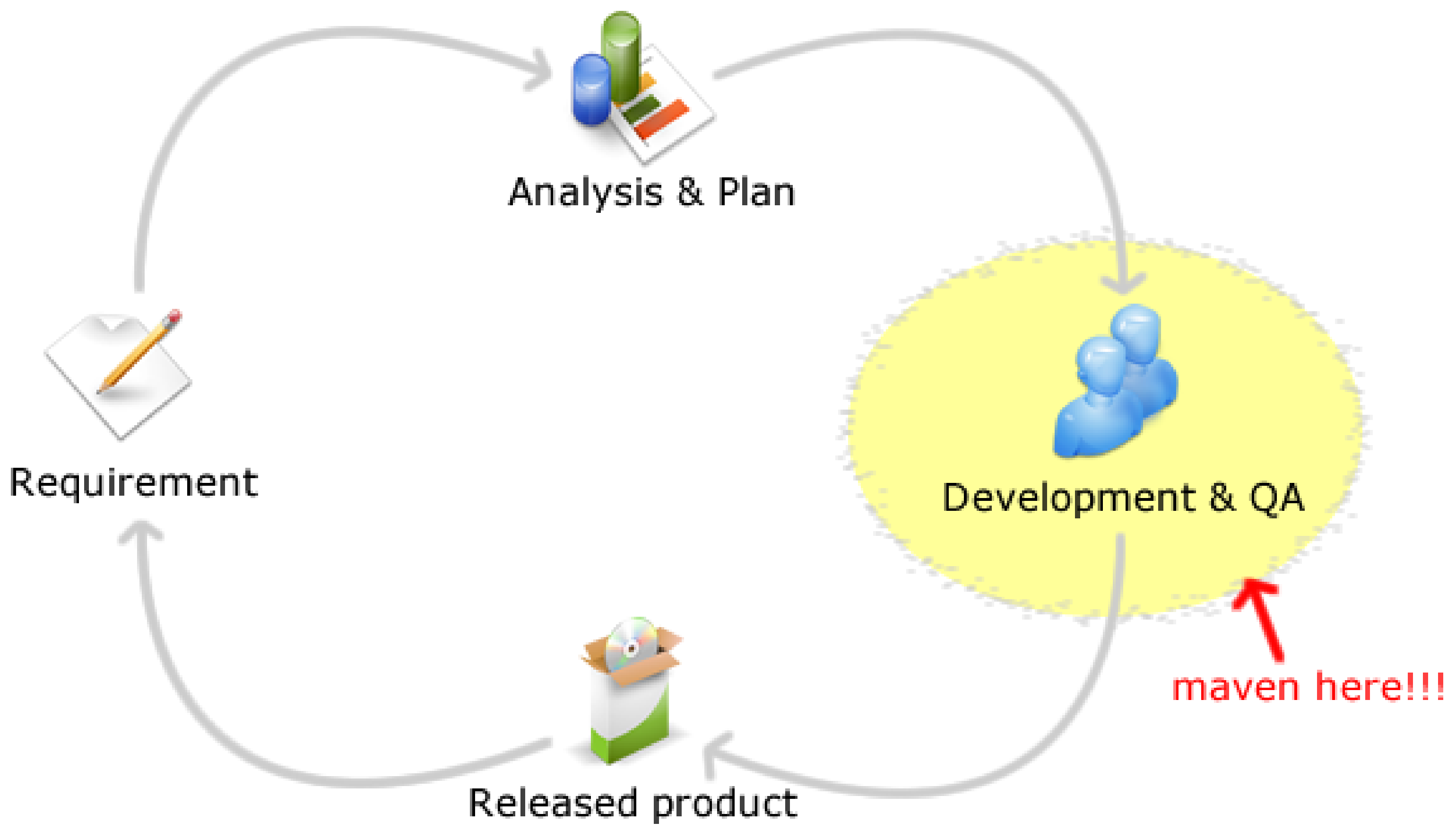
About Maven 2.0

Done

When use Maven

- Customer request
- Develop project from scratch
- Follow existing project structure

Where is Maven?



Maven benefits (Why use Maven?)

- Making build process much easier.
- Managing dependencies.
- Centralizing module dependency.
- Maintaining a standard project layout.
- Generating report on broken code, broken coding convention, unit test code coverage etc…

Maven’s Principles (How can Maven do that?)

- **Convention over configuration**
  - Standard directory layout for projects
  - The concept of a single Maven project producing a single output
  - Standard naming conventions
- **Declarative execution**:
  - using *Maven's Project Object Model (POM)*
  - The execution of Maven's plug-ins is coordinated by Maven's build life cycle in a declarative fashion with instructions from Maven's POM.
- **Reuse of build logic**
  - Separate building logic into coherent modules.
  - Everything accomplished in Maven is the result of a plugin executing. Plugins are the key building blocks for everything in Maven.
- **Coherent organization of dependencies**
  - Automatically locates dependencies from remote and local repositories.

Using Maven

Getting Started

Install Maven

- **Step 1** - verify Java installation on your machine
- **Step 2** - set JAVA environment - set the **JAVA_HOME** environment variable
- **Step 3** - download Maven2 from

  http://maven.apache.org/

  For example : apache-maven-3.3.3-bin.zip
- **Step 4**: Extract the Maven archive
- **Step 5**: Set Maven environment variables

  Add M2_HOME, M2, MAVEN_OPTS to environment variable

  For example: Set the environment variables using system properties.

M2_HOME=C:\Program Files\Apache Software Foundation\apache-maven-3.3.3

MAVEN_OPTS=-Xms256m -Xmx512m

Install Maven

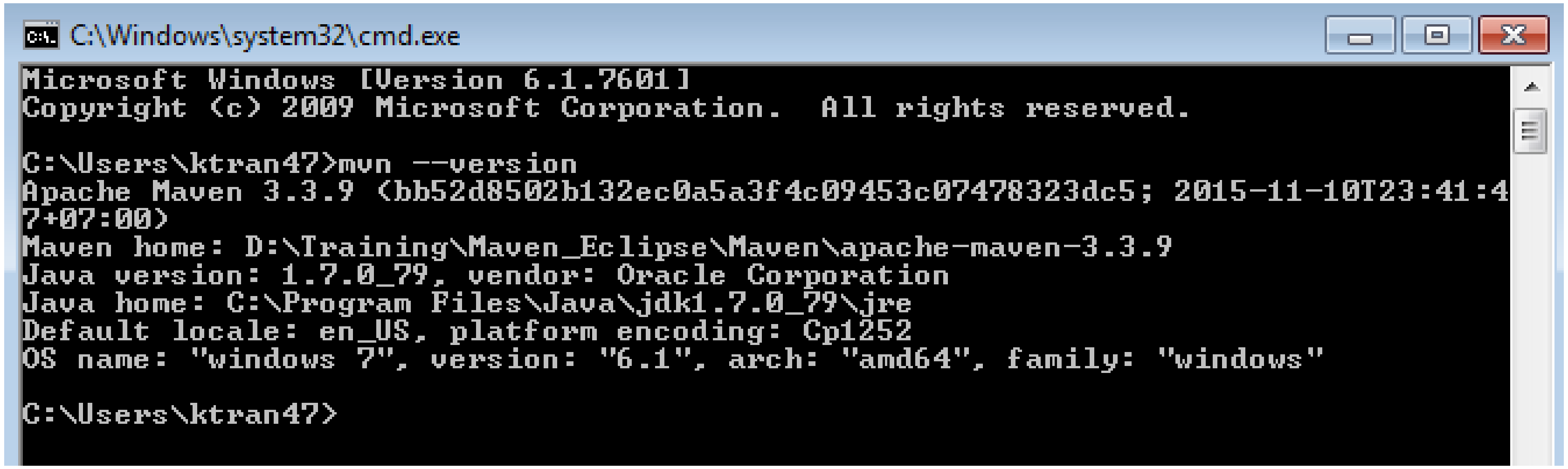
- **Step 6**: Add Maven bin directory location to system path

  %M2_HOME%\bin

- **Step 8:** Verify Maven installation

  execute the following **mvn** command.

  mvn –version



C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\ktran47>mvn --version
Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5; 2015-11-10T23:41:4
7+07:00)
Maven home: D:\Training\Maven_Eclipse\Maven\apache-maven-3.3.9
Java version: 1.7.0_79, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.7.0_79\jre
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 7", version: "6.1", arch: "amd64", family: "windows"

C:\Users\ktran47>
```

Maven - POM

- POM stands for *Project Object Model*
- It is fundamental Unit of Work in Maven.
- It is an XML file.
- It always resides in the base directory of the project as pom.xml.
- **Example POM**

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.companyname.insurance</groupId>
    <artifactId>cscv</artifactId>
    <packaging>jar</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>cscv</name>
    <url>http://maven.apache.org</url>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

Maven - POM

- There is a single POM file for each project.
- All POM files require the **project** element and three mandatory fields: **groupId, artifactId,version.**
- Projects notation in repository is **groupId:artifactId:version.**
- Root element of POM.xml is **project**

Maven - Build Life Cycle

A *Build Lifecycle* is a well defined sequence of phases which define the order in which the goals are to be executed

- **Clean Lifecycle**
- **Default (or Build) Lifecycle**

This is the primary life cycle of Maven and is used to build the application.

There are 23 phases. Example phrase.

validate - Validates whether project is correct and all necessary information is available to complete the build process.

generate-sources - Generate any source code to be included in compilation phase.

generate-resources - Generate resources to be included in the package.

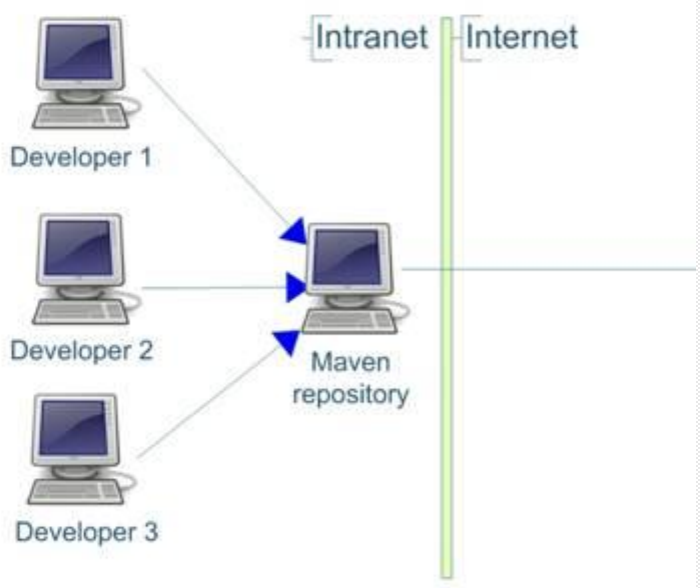
package Take the compiled code and package it in its distributable format, such as a JAR, WAR, or EAR file.

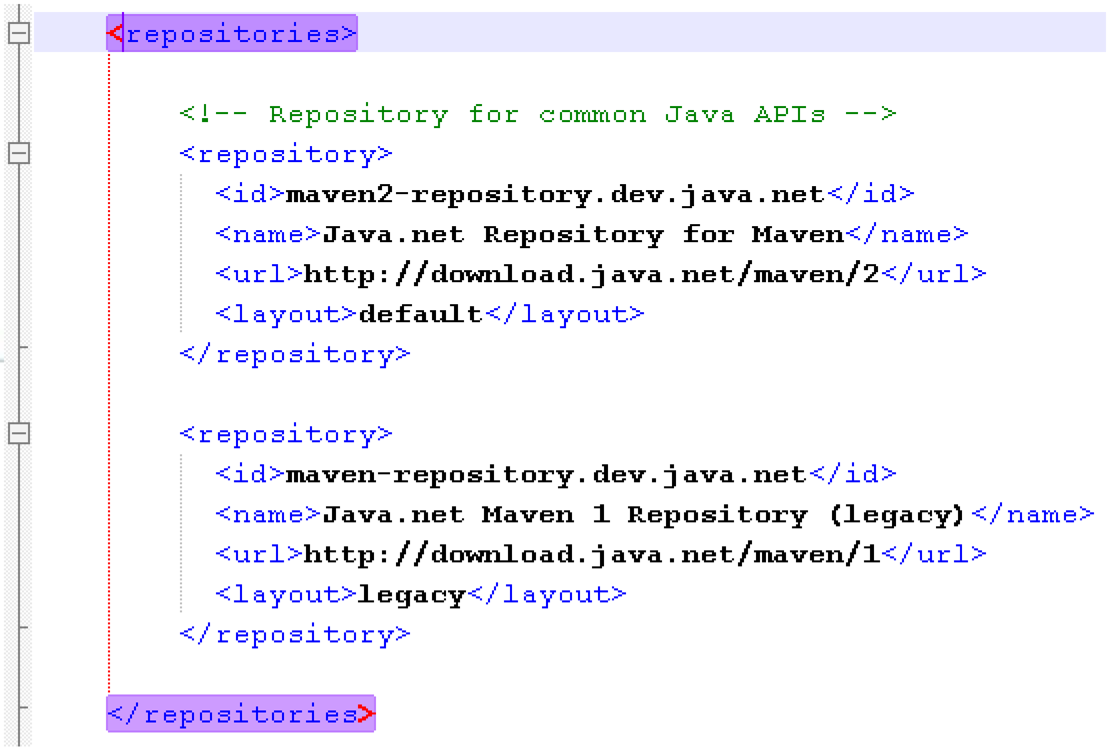
compile - Compile the source code of the project.

- **Site Lifecycle**

Maven Repositories

- A repository is a place i.e. directory where all the project jars, library jar, plugins or any other project specific artifacts are stored
- Can be used by Maven easily.





```
<repositories>

    <!-- Repository for common Java APIs -->
    <repository>
      <id>maven2-repository.dev.java.net</id>
      <name>Java.net Repository for Maven</name>
      <url>http://download.java.net/maven/2</url>
      <layout>default</layout>
    </repository>

    <repository>
      <id>maven-repository.dev.java.net</id>
      <name>Java.net Maven 1 Repository (legacy)</name>
      <url>http://download.java.net/maven/1</url>
      <layout>legacy</layout>
    </repository>

</repositories>
```

Maven Repository Configuration in ‘pom.xml’

Maven Repositories

- Maven repository are of three types : local, central, remote.

- **Local Repository**

  Maven local repository is a folder location on your machine. It gets created when you run any maven command for the first time. Maven local repository keeps your project's all dependencies (library jars, plugin jars etc)

- **Central Repository**

  It is repository provided by Maven community

  It contains a large number of commonly used libraries

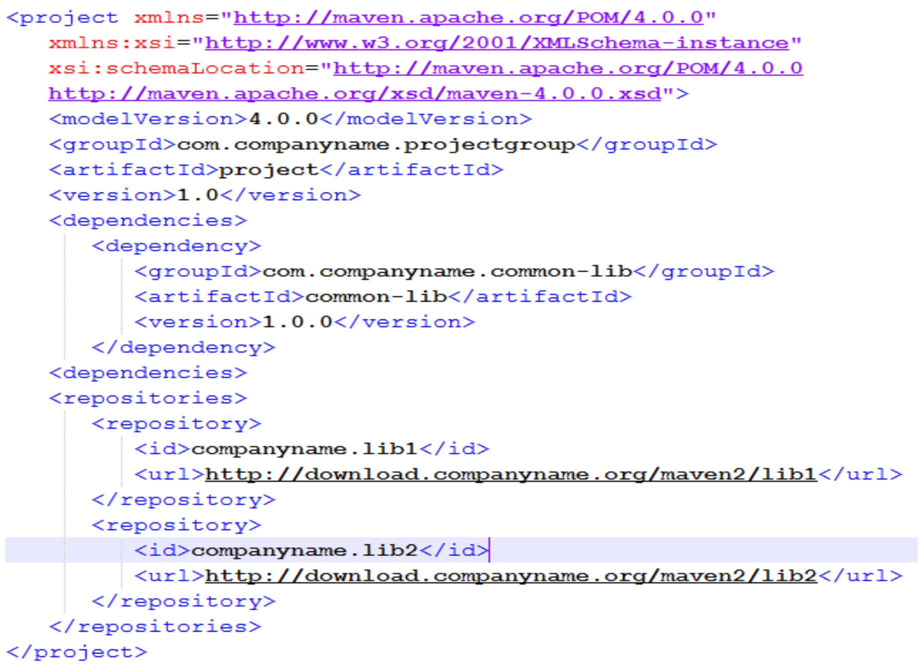
  When Maven does not find any dependency in local repository, it starts searching in central repository.

  When dependency in central repository is not found, it will search dependency in remote repository.

Maven Repositories

- **Remote Repository**

  Maven does not find a mentioned dependency in central repository as well then it stopped build process and output error message to console. To prevent such situation, Maven provides concept of Remote Repository which is developer's own custom repository containing required libraries or other project jars.



```
<project xmlns="http://maven.apache.org/POM/4.0.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
   http://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <groupId>com.companyname.projectgroup</groupId>
   <artifactId>project</artifactId>
   <version>1.0</version>
   <dependencies>
      <dependency>
         <groupId>com.companyname.common-lib</groupId>
         <artifactId>common-lib</artifactId>
         <version>1.0.0</version>
      </dependency>
   <dependencies>
   <repositories>
      <repository>
         <id>companyname.lib1</id>
         <url>http://download.companyname.org/maven2/lib1</url>
      </repository>
      <repository>
         <id>companyname.lib2</id>
         <url>http://download.companyname.org/maven2/lib2</url>
      </repository>
   </repositories>
</project>
```

Maven - Plug-ins

- 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]

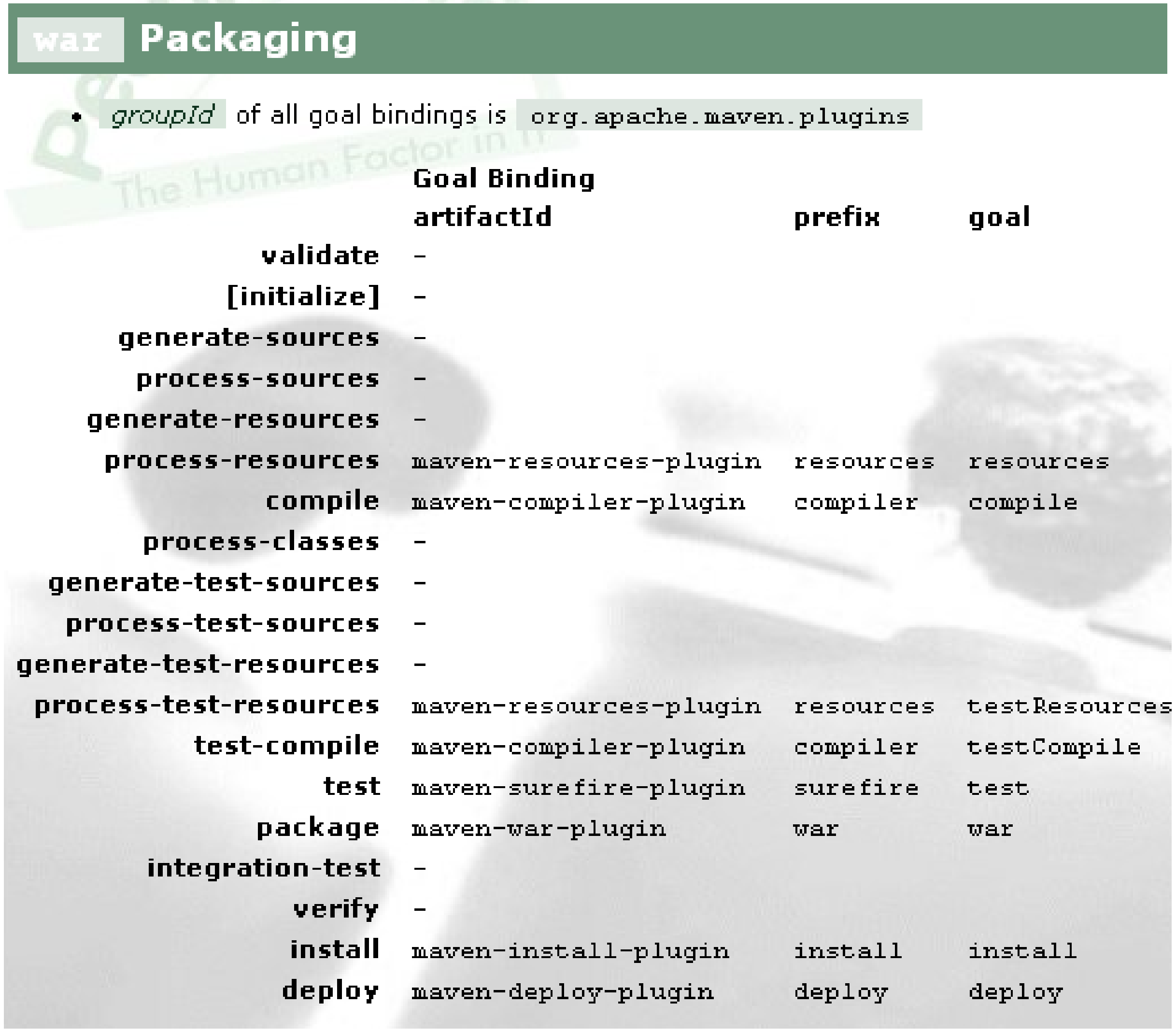
  Example: mvn compiler:compile

- Maven provided following two types of Plugins:

  Build plugins - configured in the <build/> element of pom.xml

  Reporting plugins - configured in the <reporting/> element of the pom.xml

Maven - Plug-ins



war Packaging

- *groupId* of all goal bindings is `org.apache.maven.plugins`

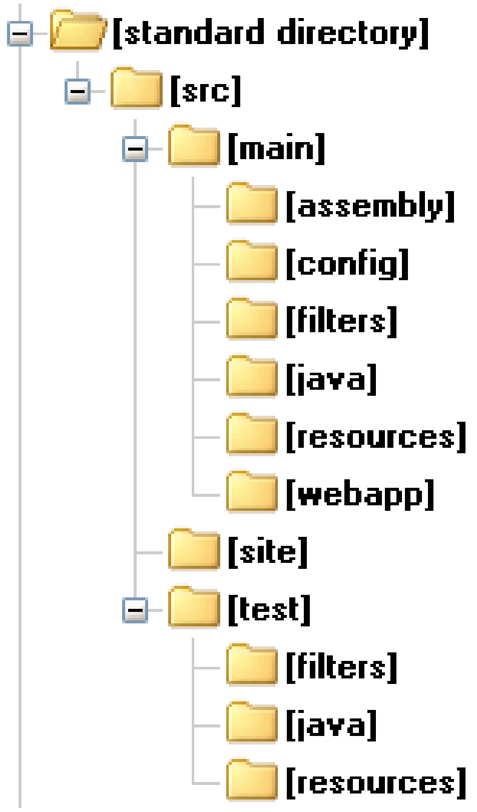
	Goal Binding		
	artifactId	prefix	goal
validate	-		
[initialize]	-		
generate-sources	-		
process-sources	-		
generate-resources	-		
process-resources	maven-resources-plugin	resources	resources
compile	maven-compiler-plugin	compiler	compile
process-classes	-		
generate-test-sources	-		
process-test-sources	-		
generate-test-resources	-		
process-test-resources	maven-resources-plugin	resources	testResources
test-compile	maven-compiler-plugin	compiler	testCompile
test	maven-surefire-plugin	surefire	test
package	maven-war-plugin	war	war
integration-test	-		
verify	-		
install	maven-install-plugin	install	install
deploy	maven-deploy-plugin	deploy	deploy

Create a new maven project

Maven uses **archetype** plugins to create projects. To create a simple java application, we'll use maven-archetype-quickstart plugin

```
mvn archetype:generate
-DgroupId=com.companyname.insurance
-DartifactId=cscv
-DarchetypeArtifactId=maven-archetype-quickstart
-DinteractiveMode=false
```

Maven Standard Directory


```
[standard directory]
  [src]
    [main]
      [assembly]
      [config]
      [filters]
      [java]
      [resources]
      [webapp]
    [site]
    [test]
      [filters]
      [java]
      [resources]
```

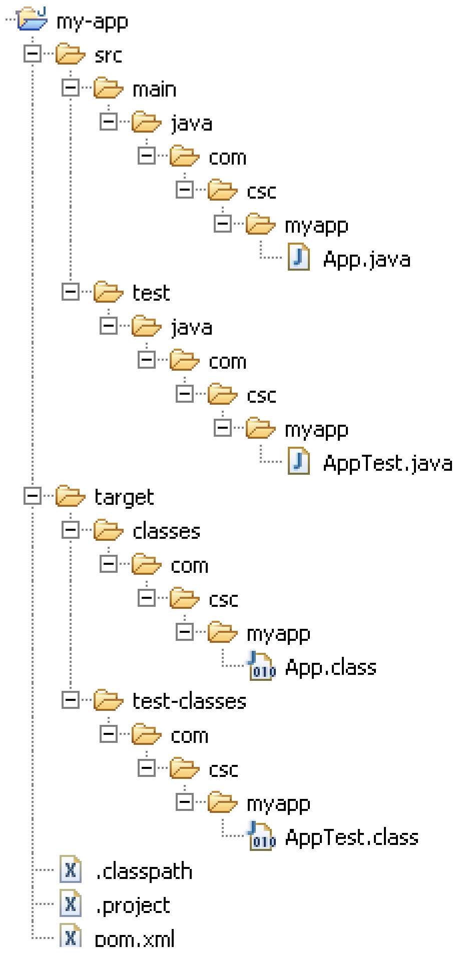
src/main/java	Application/Library sources
src/main/resources	Application/Library resources
src/main/filters	Resource filter files
src/main/assembly	Assembly descriptors
src/main/config	Configuration files
src/main/webapp	Web application sources
src/test/java	Test sources
src/test/resources	Test resources
src/test/filters	Test resource filter files
src/site	Site
LICENSE.txt	Project's license
README.txt	Project's readme

Maven - Build & Test Project

Go to directory where you've created your java application

```
mvn clean package
```

Generated classes and resources after compile



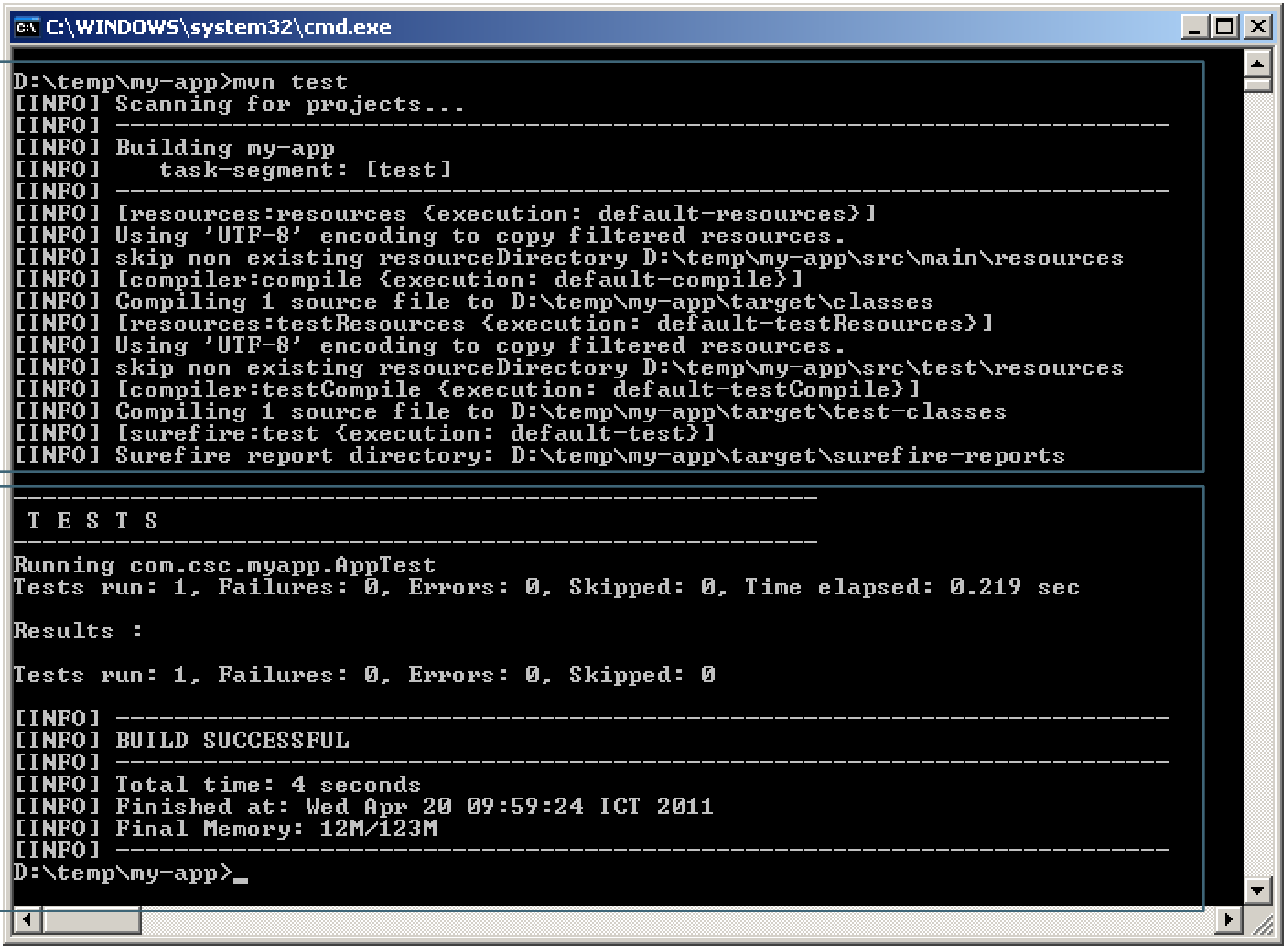
```
my-app
├─ src
│  ├─ main
│  │  └─ java
│  │     └─ com
│  │        └─ csc
│  │           └─ myapp
│  │              └─ App.java
│  └─ test
│     └─ java
│        └─ com
│           └─ csc
│              └─ myapp
│                 └─ AppTest.java
├─ target
│  ├─ classes
│  │  └─ com
│  │     └─ csc
│  │        └─ myapp
│  │           └─ App.class
│  └─ test-classes
│     └─ com
│        └─ csc
│           └─ myapp
│              └─ AppTest.class
├─ .classpath
├─ .project
└─ pom.xml
```

Running unit test on maven project

```
mvn test
```

Maven auto build before test

Test and Report

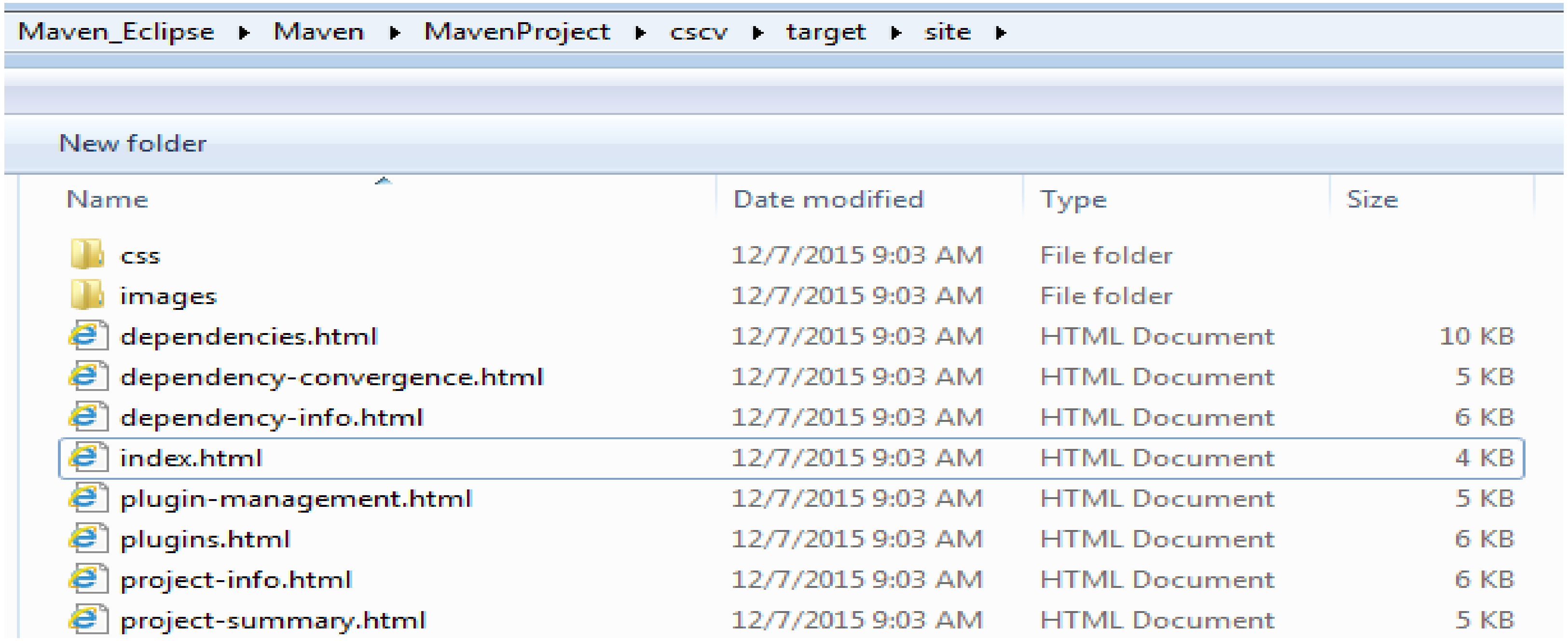


```
D:\temp\my-app>mvn test
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Building my-app
[INFO]    task-segment: [test]
[INFO] ------------------------------------------------------------------------
[INFO] [resources:resources {execution: default-resources}]
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory D:\temp\my-app\src\main\resources
[INFO] [compiler:compile {execution: default-compile}]
[INFO] Compiling 1 source file to D:\temp\my-app\target\classes
[INFO] [resources:testResources {execution: default-testResources}]
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory D:\temp\my-app\src\test\resources
[INFO] [compiler:testCompile {execution: default-testCompile}]
[INFO] Compiling 1 source file to D:\temp\my-app\target\test-classes
[INFO] [surefire:test {execution: default-test}]
[INFO] Surefire report directory: D:\temp\my-app\target\surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.csc.myapp.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.219 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 4 seconds
[INFO] Finished at: Wed Apr 20 09:59:24 ICT 2011
[INFO] Final Memory: 12M/123M
[INFO] ------------------------------------------------------------------------
D:\temp\my-app>_
```

Maven - Project Documents

- Create documentation of the application in one go
- Go to the directory where you had created your java project, execute the following **mvn** command.

  mvn site

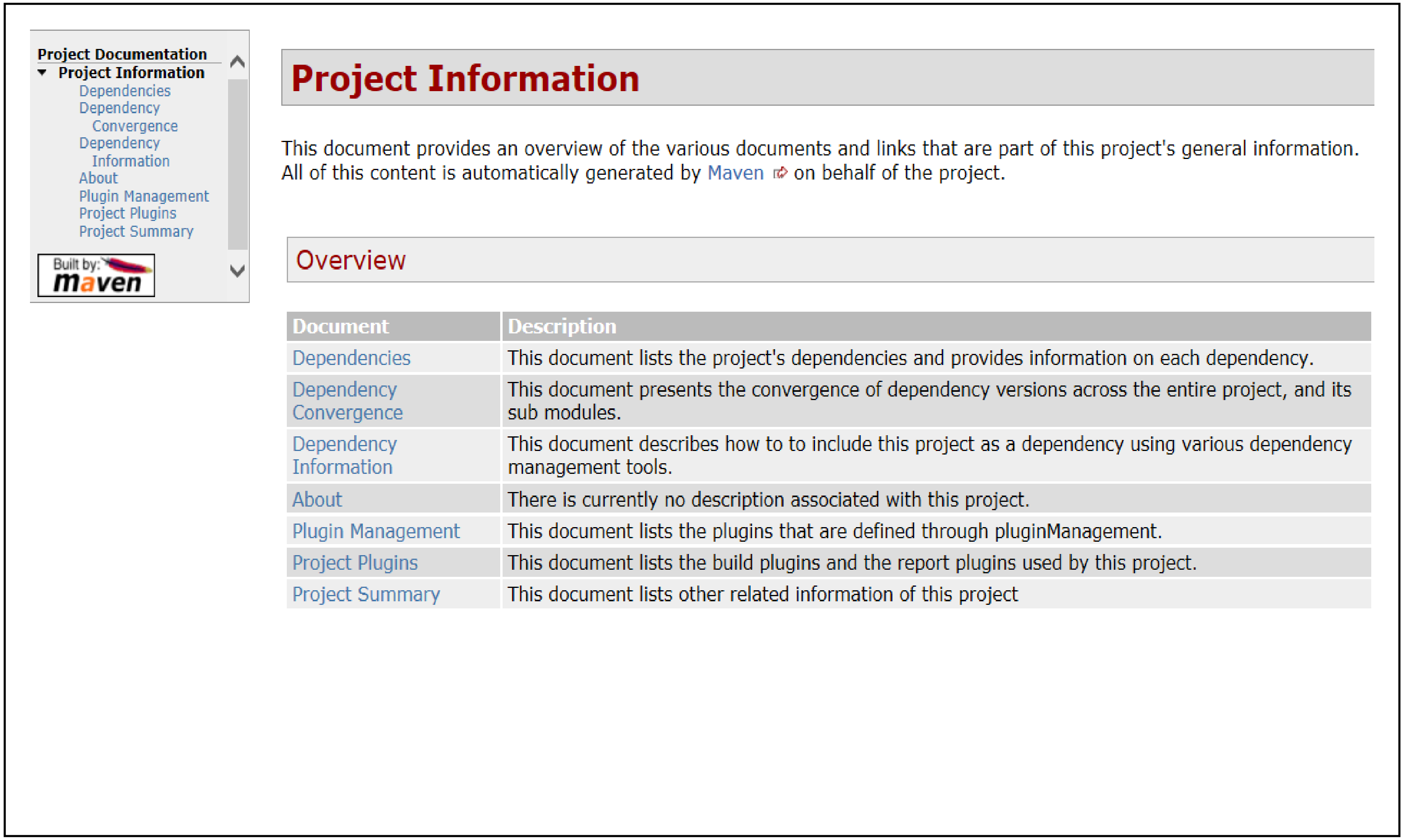


Maven_Eclipse ▸ Maven ▸ MavenProject ▸ cscv ▸ target ▸ site ▸

New folder

Name	Date modified	Type	Size
css	12/7/2015 9:03 AM	File folder	
images	12/7/2015 9:03 AM	File folder	
dependencies.html	12/7/2015 9:03 AM	HTML Document	10 KB
dependency-convergence.html	12/7/2015 9:03 AM	HTML Document	5 KB
dependency-info.html	12/7/2015 9:03 AM	HTML Document	6 KB
index.html	12/7/2015 9:03 AM	HTML Document	4 KB
plugin-management.html	12/7/2015 9:03 AM	HTML Document	5 KB
plugins.html	12/7/2015 9:03 AM	HTML Document	6 KB
project-info.html	12/7/2015 9:03 AM	HTML Document	6 KB
project-summary.html	12/7/2015 9:03 AM	HTML Document	5 KB

Maven - Project Documents



Project Information

This document provides an overview of the various documents and links that are part of this project's general information. All of this content is automatically generated by Maven on behalf of the project.

Overview

Document	Description
Dependencies	This document lists the project's dependencies and provides information on each dependency.
Dependency Convergence	This document presents the convergence of dependency versions across the entire project, and its sub modules.
Dependency Information	This document describes how to to include this project as a dependency using various dependency management tools.
About	There is currently no description associated with this project.
Plugin Management	This document lists the plugins that are defined through pluginManagement.
Project Plugins	This document lists the build plugins and the report plugins used by this project.
Project Summary	This document lists other related information of this project

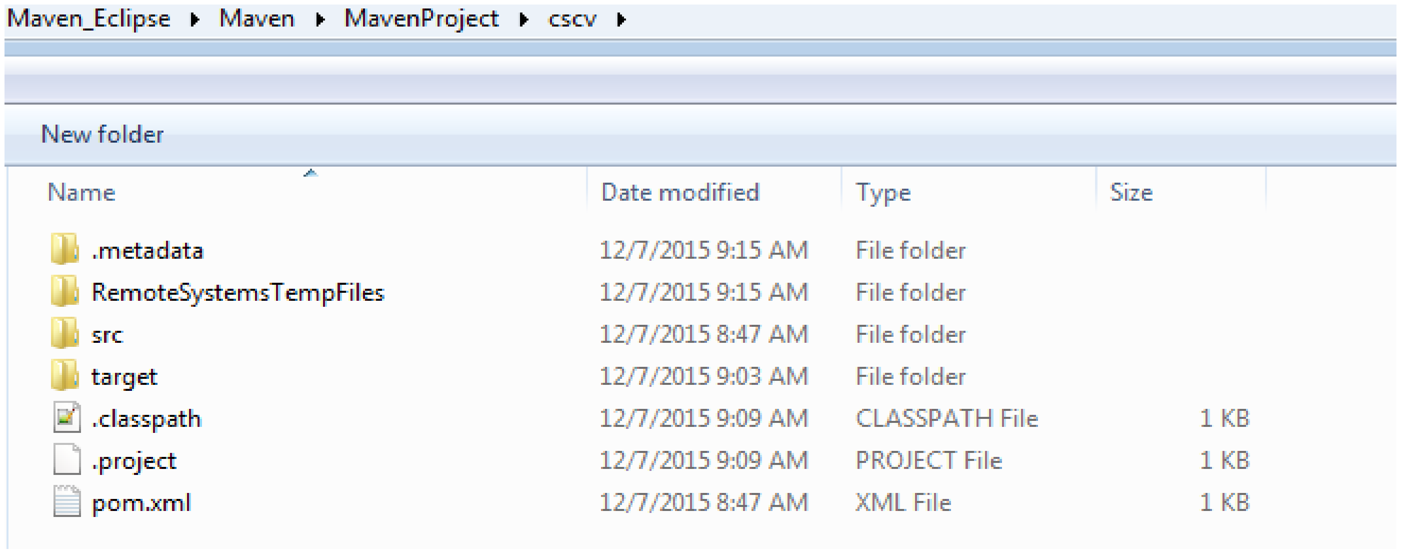
Convert Maven based Java Project to support Eclipse IDE

- Go to the directory where you had created your java project, execute the following **mvn** command.

mvn eclipse:eclipse

- Verify Java Project

You will see two new files are created – ".classpath" and ".project". Both files are created for Eclipse IDE.



Maven_Eclipse ▸ Maven ▸ MavenProject ▸ cscv ▸

New folder

Name	Date modified	Type	Size
.metadata	12/7/2015 9:15 AM	File folder	
RemoteSystemsTempFiles	12/7/2015 9:15 AM	File folder	
src	12/7/2015 8:47 AM	File folder	
target	12/7/2015 9:03 AM	File folder	
.classpath	12/7/2015 9:09 AM	CLASSPATH File	1 KB
.project	12/7/2015 9:09 AM	PROJECT File	1 KB
pom.xml	12/7/2015 8:47 AM	XML File	1 KB

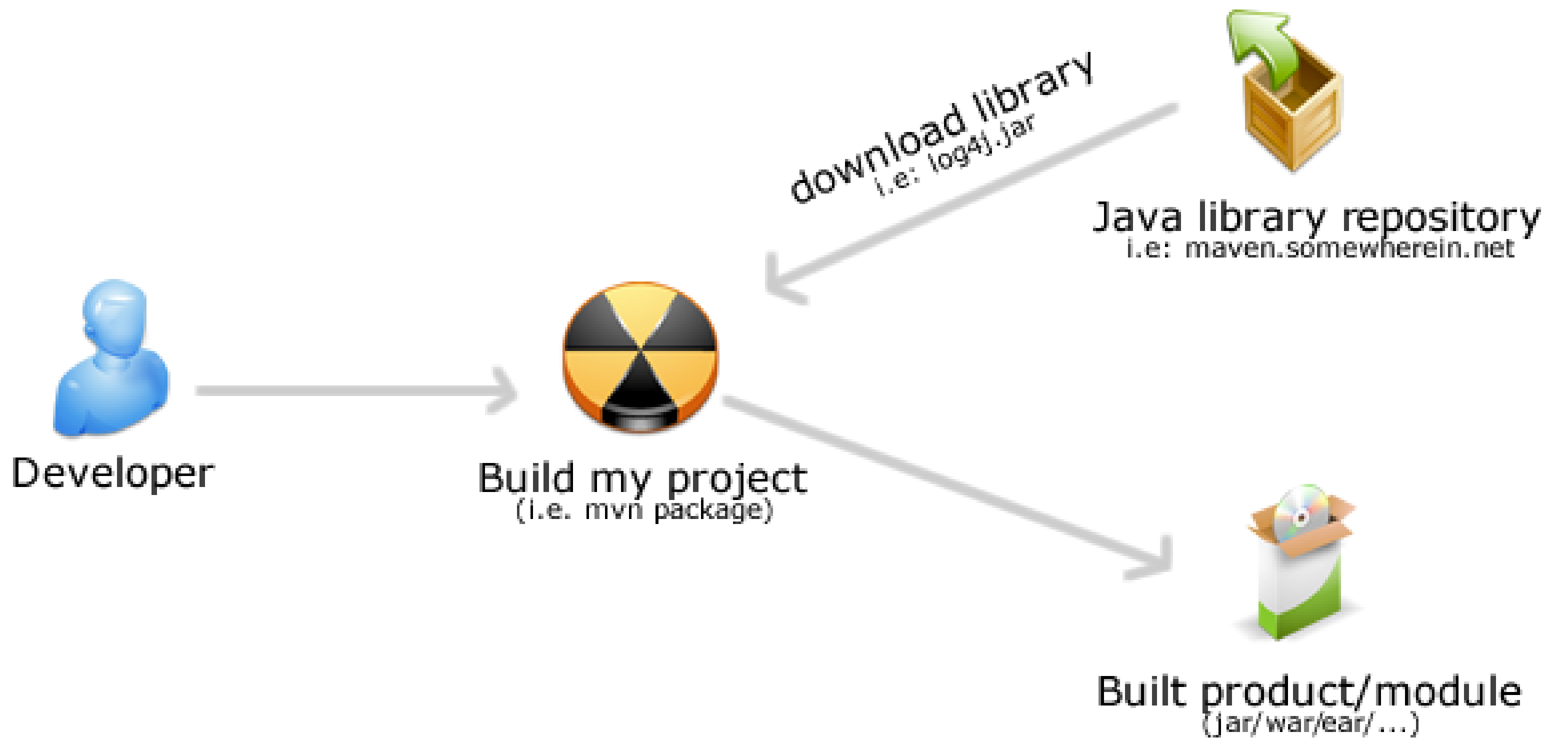
Exercise

- 10 minutes
- Create a new project with name {trainee-name}
- Write simple “HelloWorld” class which print out {trainee-name}
- Write simple unit test “HelloWorldTest” using JUnit4
- Use Maven to generate Eclipse project
- Submit project to trainer
- Print out dependency-tree, capture it and submit

Maven Principles Detail

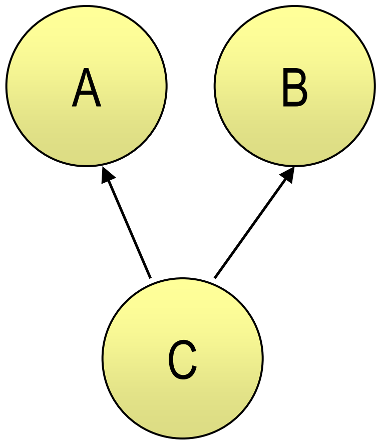
Maven - Manage Dependencies

One of the core features of Maven is Dependency Management

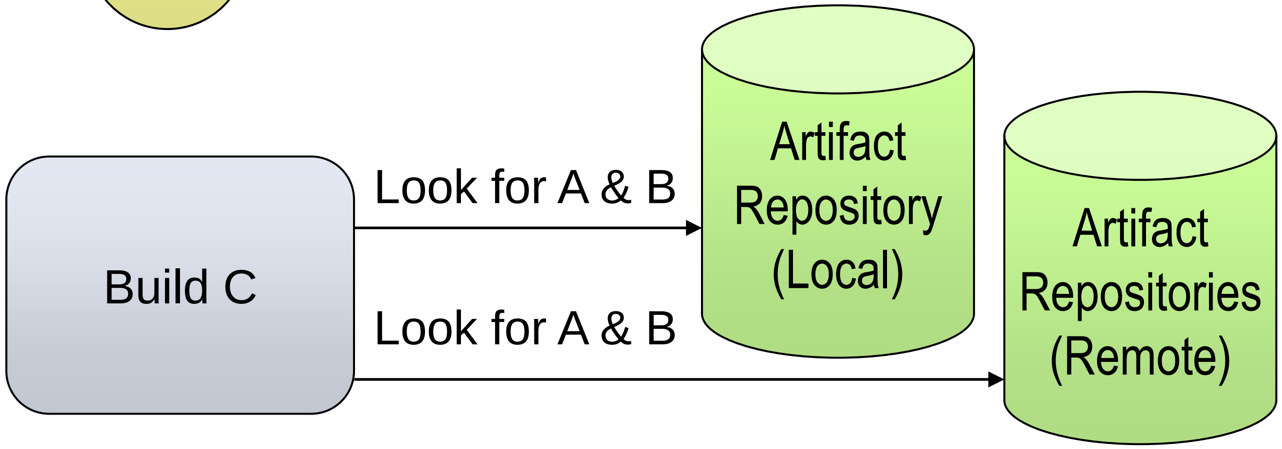


Maven - Manage Dependencies

Maven uses binary dependencies from local and remote repositories transparently from user's view.



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



Dependency scopes

- **compile** - *This scope indicates that dependency is available in classpath of project. It is default scope.*

- **provided** - *This scope indicates that dependency is to be provided by JDK or web-Server/Container at runtime*

  The Servlet API JAR is in this scope

- **test** - *Dependency is made available only at test time*

  This scope is appropriate for JUnit.

- **runtime** - *This scope indicates that dependency is not required for compilation, but is required during execution.*

- **system** - *Like "provided" but you have to provide the system path*

Maven - Snapshots

```
<project>
       [...]
       <version>1.0-SNAPSHOT</version>
       [...]
</project>
```

- While developing multiple modules, modules are in flux (usually change).
- A snapshot in Maven is an artifact that has been prepared using **the most recent sources** available.
- Maven checks for a new SNAPSHOT version in a remote repository and will automatically fetch the latest SNAPSHOT for every build (transparent to user).

Deploying your Application

```
mvn deploy
```

Deploying to the File System

```
<project>
```

```
</
```

Deploying with SSH2

```
<project>
```

```
</p
```

Deploying with FTP

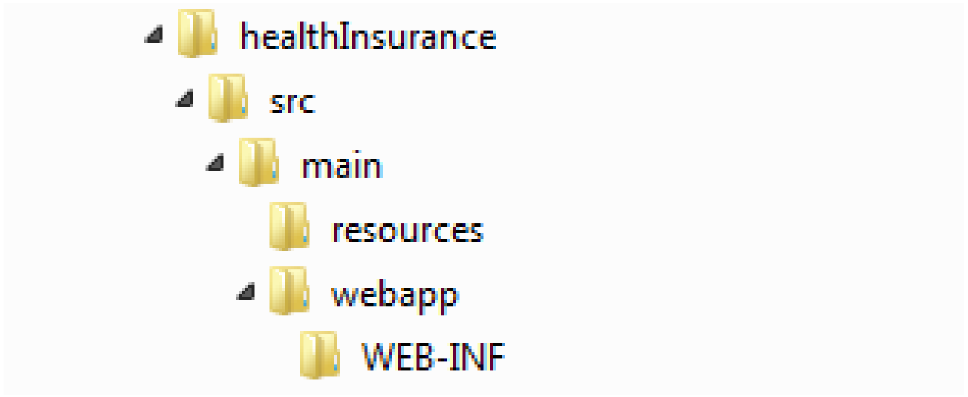
```
<project>
    [...]
    <distributionManagement>
        <repository>
            <id>proficio-repository</id>
            <name>Proficio Repository</name>
            <url>ftp://ftpserver.yourcompany.com/deploy</url>
        </repository>
    </distributionManagement>
    <build>
        <extensions>
            <extension>
                <groupId>org.apache.maven.wagon</groupId>
                <artifactId>wagon-ftp</artifactId>
                <version>1.0-alpha-6</version>
            </extension>
        </extensions>
    </build>
    [...]
</project>
```

Maven - Web Application

- To create a simple java web application, we'll use maven-archetype-webapp plugin. Execute the following **mvn** command.

```
  mvn archetype:generate
 -DgroupId=com.companyname.insurance
 -DartifactId=healthInsurance
 -DarchetypeArtifactId=maven-archetype-webapp
 -DinteractiveMode=false
```

You'll see a java application project created named **healthInsurance**



```
healthInsurance
  src
    main
      resources
      webapp
        WEB-INF
```

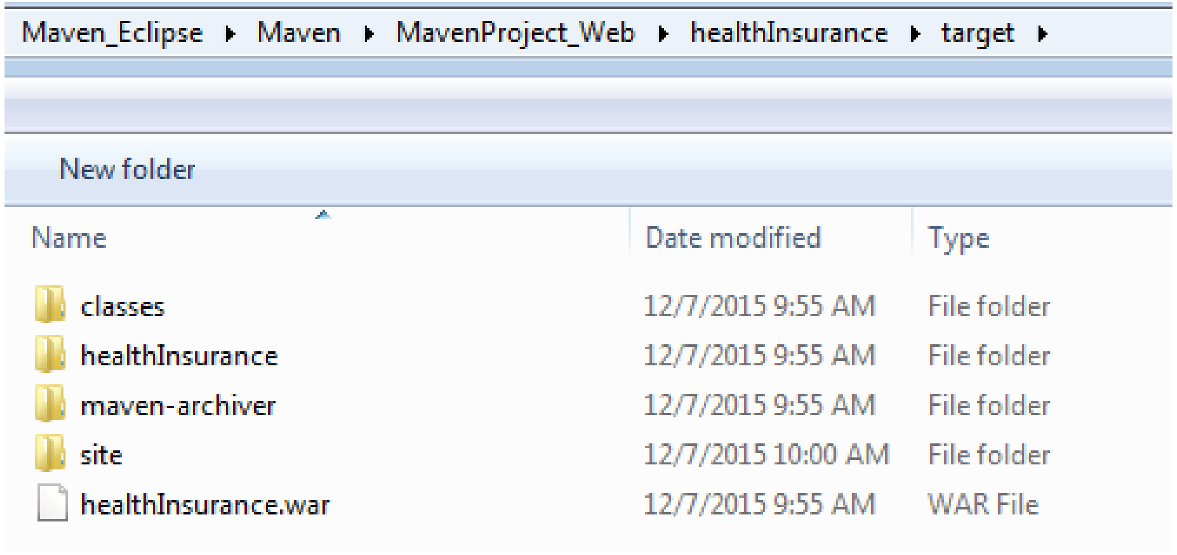
Maven - Web Application

- Build Web Application

  Execute the following **mvn** command.

  mvn clean package

A war file is created in target folder.



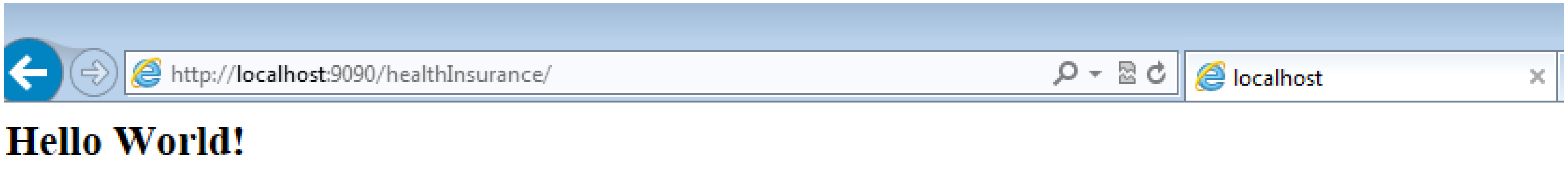
```
Maven_Eclipse ▸ Maven ▸ MavenProject_Web ▸ healthInsurance ▸ target ▸
```

New folder

Name	Date modified	Type
classes	12/7/2015 9:55 AM	File folder
healthInsurance	12/7/2015 9:55 AM	File folder
maven-archiver	12/7/2015 9:55 AM	File folder
site	12/7/2015 10:00 AM	File folder
healthInsurance.war	12/7/2015 9:55 AM	WAR File

Maven - Web Application

- Deploy Web Application by using Tomcat. Start webserver and verify output.



Hello World!

Maven - How to deploy Maven Base War file to Tomcat

Tomcat 7

Deploy URL = http://localhost:8080/manager/text

Command = mvn tomcat7:deploy

Tomcat 6

Deploy URL = http://localhost:8080/manager/

Command = mvn tomcat6:deploy

Maven - How to deploy Maven Base War file to Tomcat

Configure Tomcat

%TOMCAT7_PATH%/conf/tomcat-users.xml

```
<tomcat-users>
        <role rolename="manager-gui"/>
        <role rolename="manager-script"/>
        <user username="admin" password="password" roles="manager-gui,manager-script" />
</tomcat-users>
```

Maven - How to deploy Maven Base War file to Tomcat

- **Configure Maven settings.xml**

  %MAVEN_PATH%/conf/settings.xml

```
 <settings ...>
           <servers>
                      <server>
                                 <id>TomcatServer</id>
                                 <username>admin</username>
                                 <password>password</password>
                      </server>

           </servers>
</settings>
```

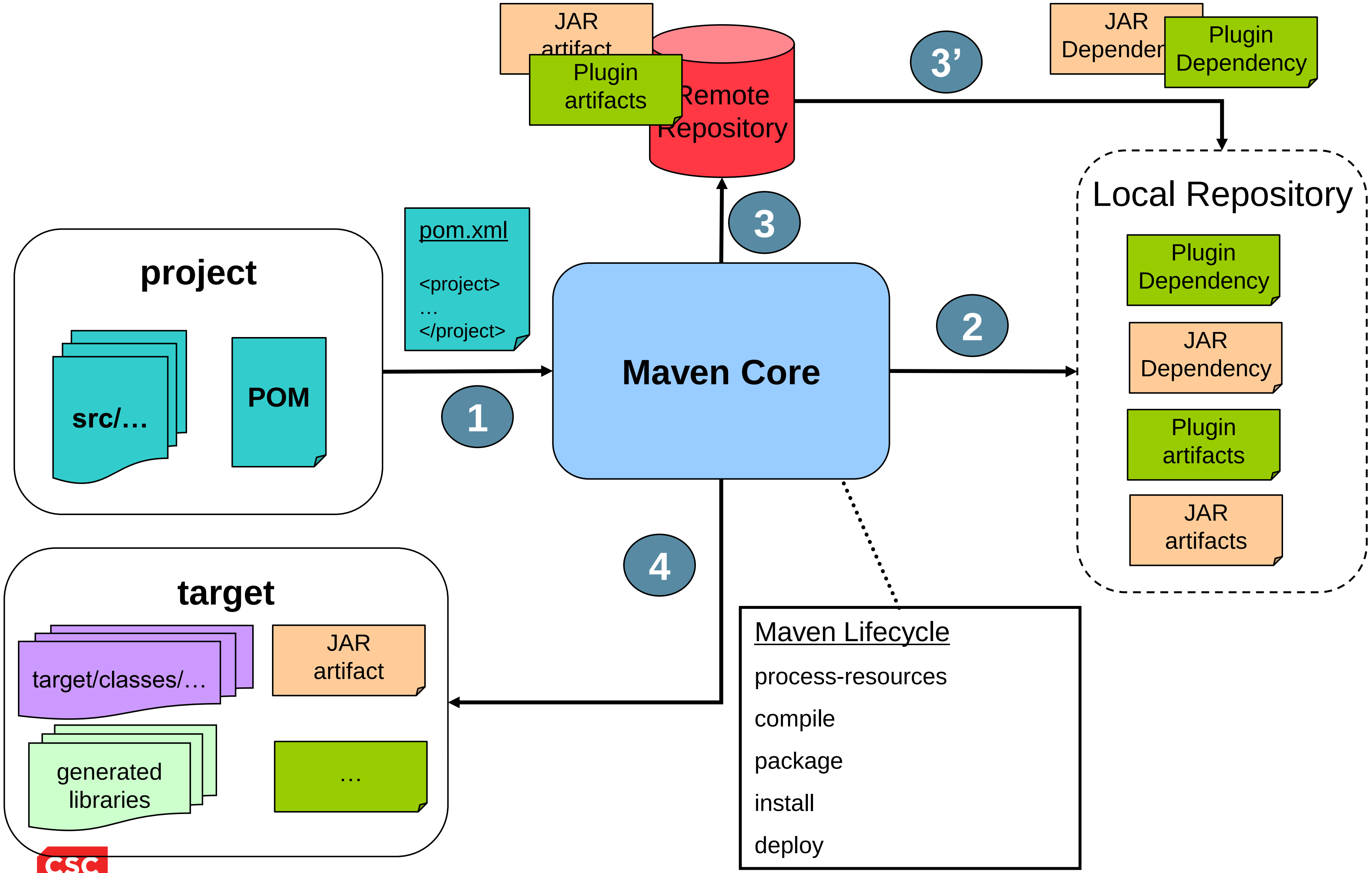
Maven - How to deploy Maven Base War file to Tomcat

- Add Maven plugin to POM file

```
<plugin>
        <groupId>org.apache.tomcat.maven</groupId>
        <artifactId>tomcat7-maven-plugin</artifactId>
        <version>2.2</version>
        <configuration>
                <url>http://localhost:8080/manager/text</url>
                <server>TomcatServer</server>
                <path>/mkyongWebApp</path>
        </configuration>
</plugin>
```

Points to Remember

Review Maven flow



Exercise

- Create a web-app project with name {trainee-name}-webapp
- Webapp has a build parameters which is the hostname of target server, this will be displayed on the index file (home page) of webapp as following: “You are running on ${hostname} server”
- Deploy project to Tomcat 6 or Tomcat 7 then submit screenshot

Q&A

Thank You.

Revision History

Date	Version	Description	Updated by	Reviewed and Approved By
18–Mar–2008	1	Release	Khanh Nguyen	
May 2011	2	RC2	Viet Nguyen	Reviewed by J2EE Group
August 2011	3	Update after pilot training: Overview of each section, Exercises and Assignments	Viet Nguyen	
September 2011	4	Update course structure, Flows, Diagrams, assignments and Final Exam tests	Viet Nguyen	First Release