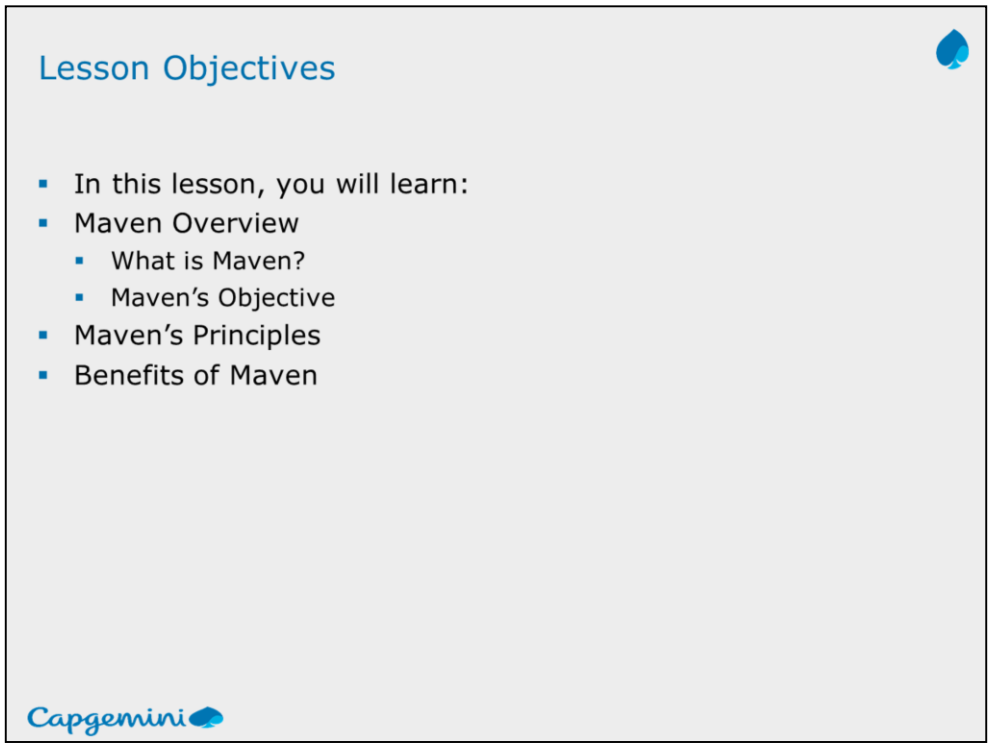
Instructor Notes:

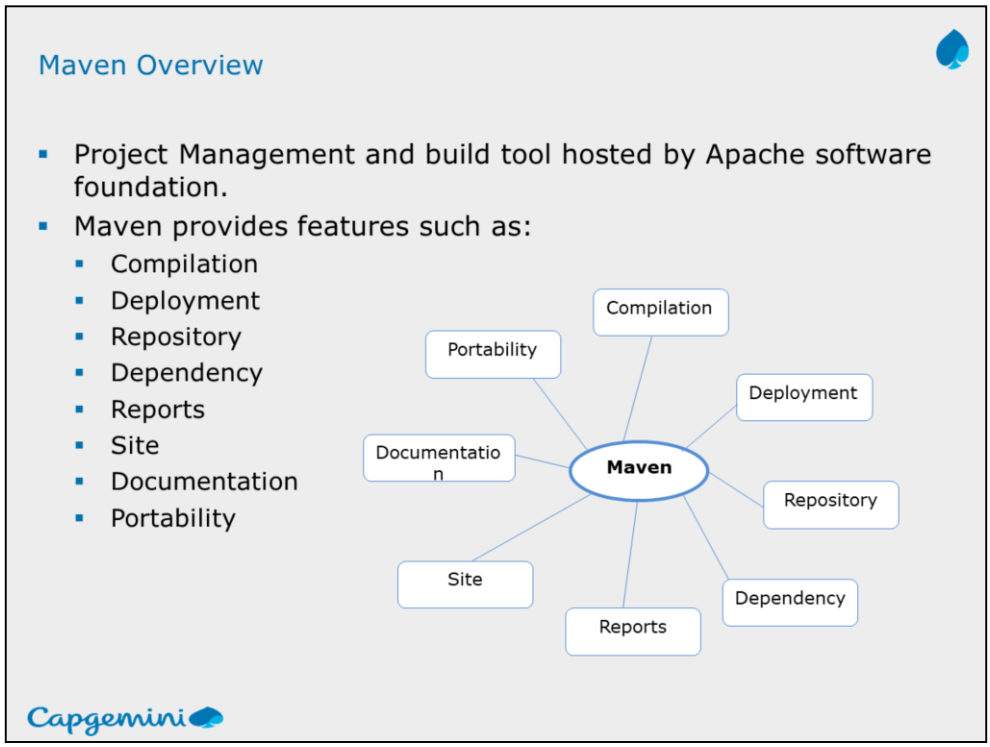
Instructor Notes:



Instructor Notes:

Ant is Task based build tool. Maven brings Object Oriented approach to build table.

A build tool such as Ant is focused solely on preprocessing, compilation, packaging, testing, and distribution. A project management tool such as Maven provides a superset of features found in a build tool. In addition to providing build capabilities, Maven can also run reports, generate a web site, and facilitate communication among members of a working team.



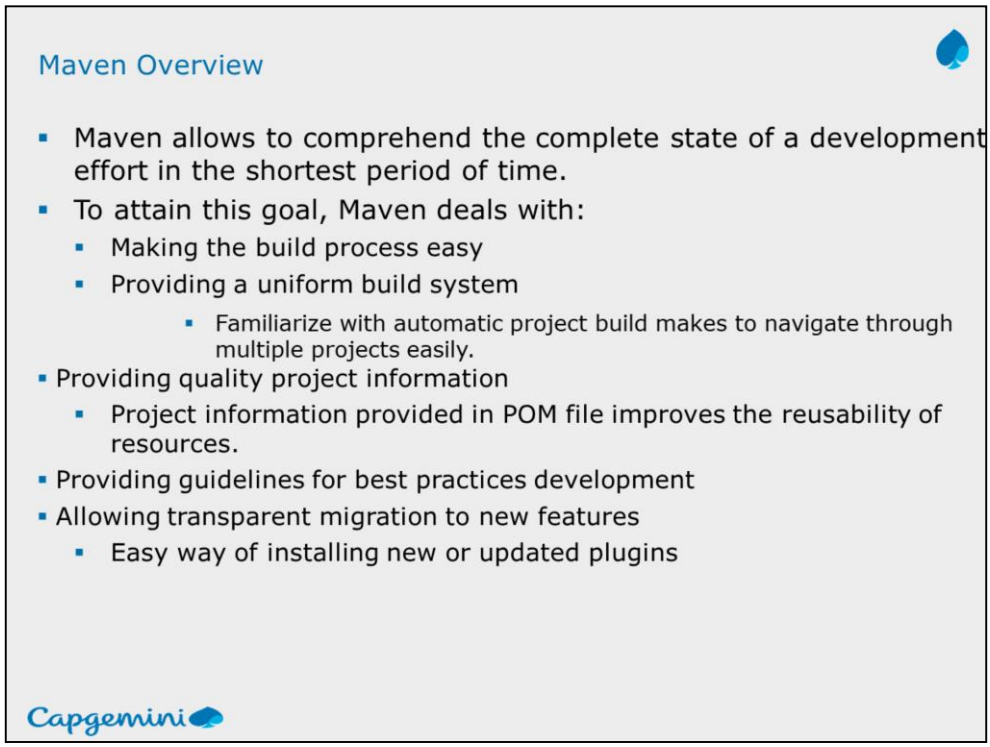
- **Maven** is a tool for project management and build automation.
- **Maven is not Ant++**
- Maven serves a similar purpose to the Apache Ant tool, but it is based on different concepts and works in a profoundly different manner.
- Maven is hosted by the Apache Software Foundation, where it was formerly part of the Jakarta Project**.**
- It is pronounced as “May-ven” and Maven is a Yiddish (Jewish Lang) word which means “**accumulator of knowledge**“.

Actually, there is a open source project named Jakarta Turbine which was facing complexity issues with module build processes and atlast the Apache team came up with a solution to simplify this build process which was than termed as Apache Maven.

Maven provides features such as:

- Build tool Capabilities
- Run Reports
- Generate a website
- Dependency Management
- Repositories (Reusable Plug-ins)
- Continuous Integration build systems
- Portable
- Building configuration using maven are portable to another machine without any effort

Instructor Notes:



Maven allows to comprehend the complete state of a development effort in the shortest period of time. To attain this goal, Maven deals with:

Making the build process easy:

While using Maven doesn't eliminate the need to know about the underlying mechanisms, Maven does provide a lot of shielding from the details.

Providing a uniform build system:

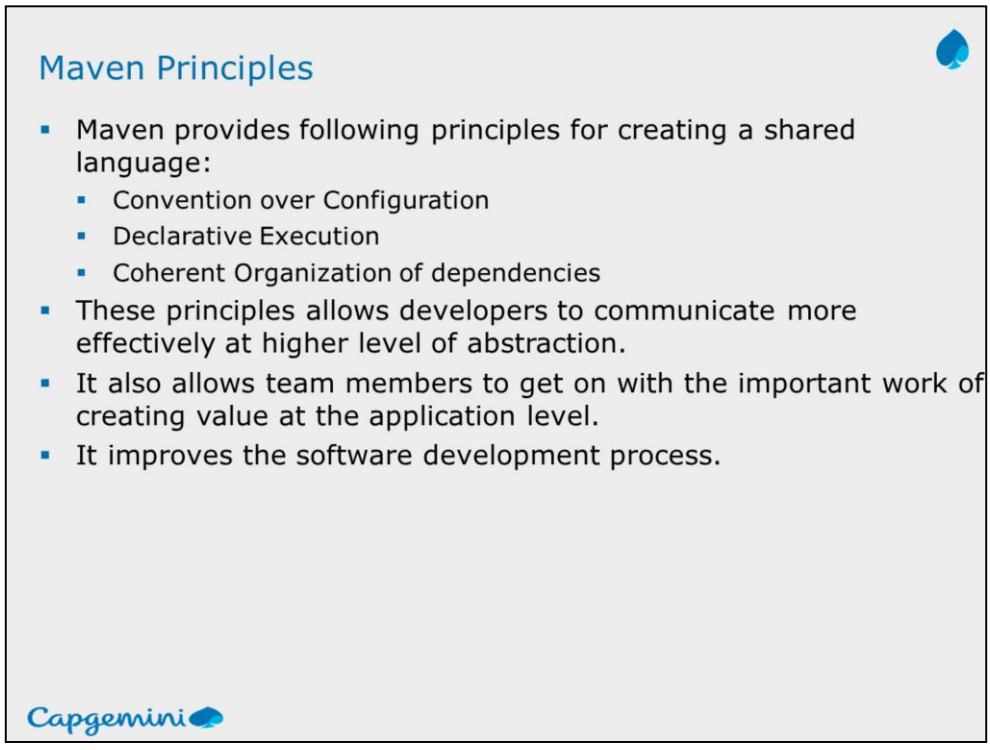
Maven allows a project to build using its project object model (POM) and a set of plugins that are shared by all projects using Maven, providing a uniform build system. Once you familiarize yourself with how one Maven project builds you automatically know how all Maven projects build saving you immense amounts of time when trying to navigate many projects.

Providing quality project information:

Maven provides plenty of useful project information that is in part taken from your POM and in part generated from your project's sources. For example, Maven can provide:

- Change log document created directly from source control
- Cross referenced sources
- Mailing lists
- Dependency list
- Unit test reports including coverage

Instructor Notes:

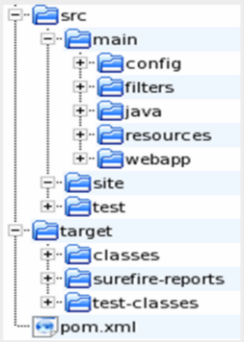


Instructor Notes:

Zero effort – No need to specify the location of source file where it is. Just place source file in the correct directory, maven will take care of the rest while building the project.

Maven Folder Structure

- Facilitate a uniform build system that speeds up the development cycle.
- There are three primary conventions that Maven employs to promote a familiar development environment:
  - Standard Project Layout
  - Standard Naming conventions
  - One Primary Output per Project
- Maven will require almost zero effort,
- if convention is followed.



Capgemini

Conventions over configuration used to facilitate a uniform build system that speeds up the development cycle.

There are three primary conventions that Maven employs to promote a standardized development environment:

Standard Project Layout:

Having a common directory layout would allow for users familiar with one Maven project to immediately feel at home in another Maven project. The advantages are analogous to adopting a site-wide look-and-feel.

Standard Naming conventions

If multiple projects are involved, standard naming convention for directories provide clarity and immediate comprehension.

One Primary Output per Project:

Instead of producing a single JAR file for entire project, Maven would encourage you to have three, separate projects: a project for the client portion of the application, a project for the server portion of the application, and a project for the shared utility code portion.

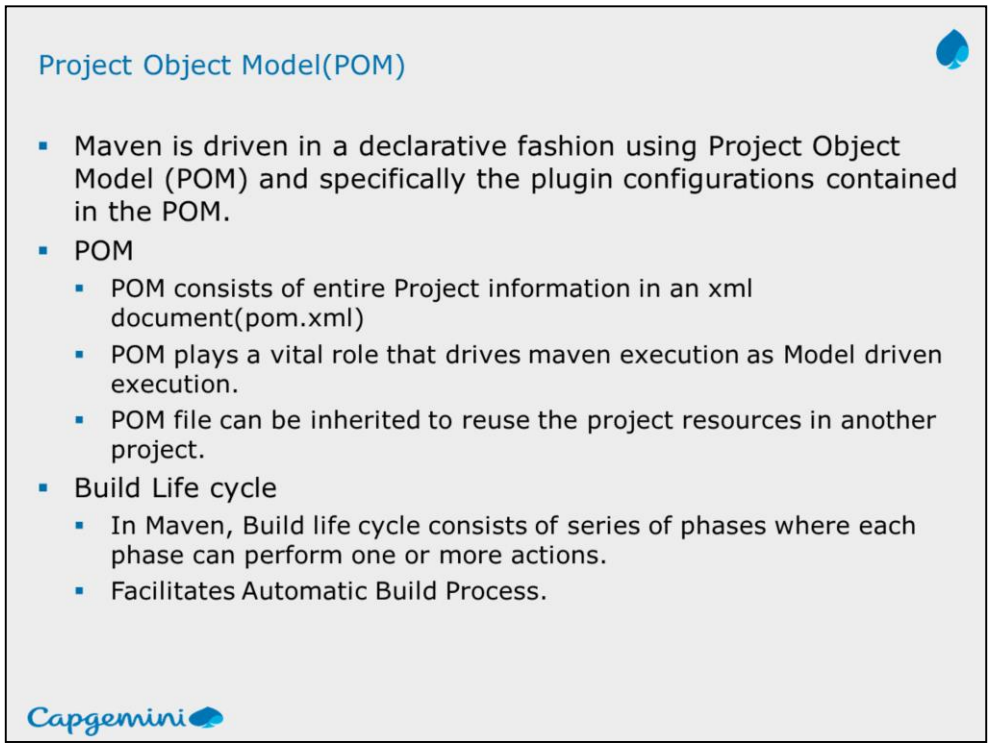
This separation of concerns (SoC) principle used to achieve the required engineering quality factors such as adaptability, maintainability, extendibility and reusability.

If you follow the convention, Maven will require almost zero effort - just put your

source in the correct directory and Maven will take care of the rest.

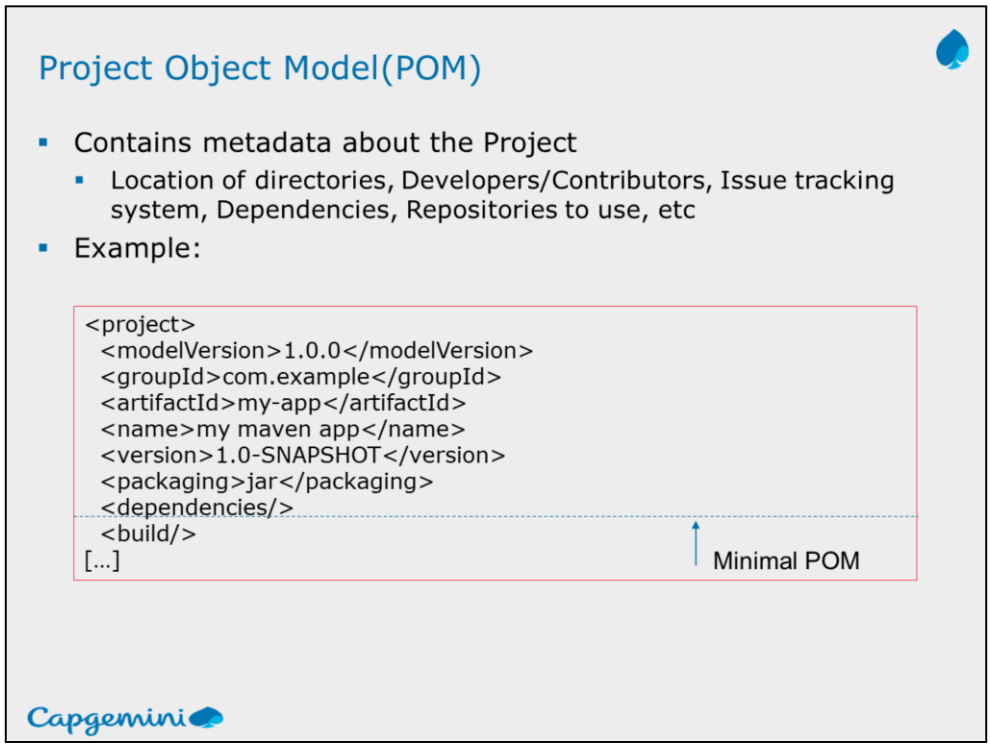
Instru

Instructor Notes:

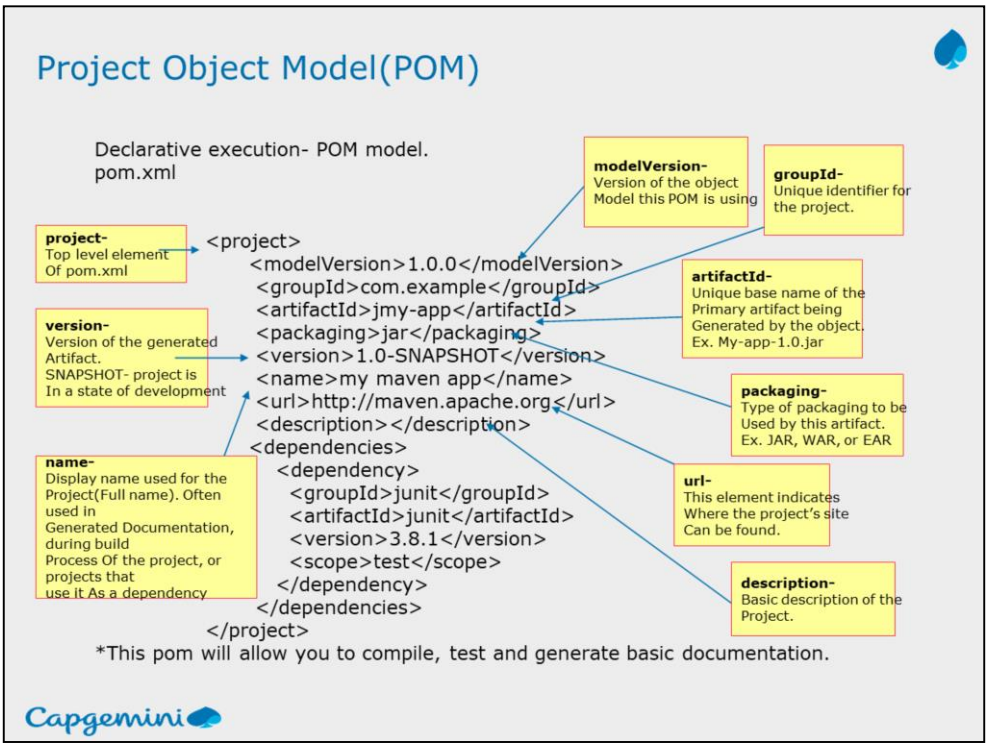


Maven is driven in a declarative fashion using Maven's Project Object Model (POM) and specifically the plug-in configurations contained in the POM.

Instructor Notes:



Instructor Notes:



The **<project>** element is the root of the project descriptor.

<modelVersion> : Declares to which version of project descriptor this POM conforms.

<groupId> :A universally unique identifier for a project. It is normal to use a fully-qualified package name to distinguish it from other projects with a similar name (eg. org.apache.maven).

<artifactId > : The identifier for this artifact that is unique within the group given by the group ID. An artifact is something that is either produced or used by a project. Examples of artifacts produced by Maven for a project include: JARs, source and binary distributions, and WARs.

<packaging> : The type of artifact this project produces, for example jar war ear pom. Plugins can create their own packaging, and therefore their own packaging types, so this list does not contain all possible types.
Default value is: jar.

<Name> : The full name of the project.

<url> : The URL to the project's homepage.

<Version> : The current version of the artifact produced by this project.

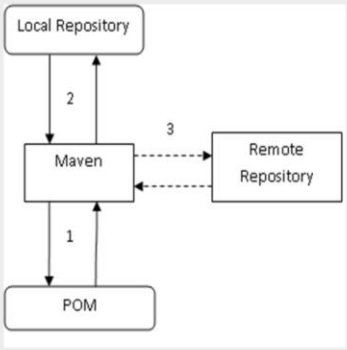
<description> : A detailed description of the project, used by Maven whenever it needs to describe the project, such as on the web site.

<dependencies>: This element describes all of the dependencies associated with a project. These dependencies are used to construct a classpath for your project during the build process. They are automatically downloaded from the repositories defined in this project.

Instructor Notes:

Maven Dependencies

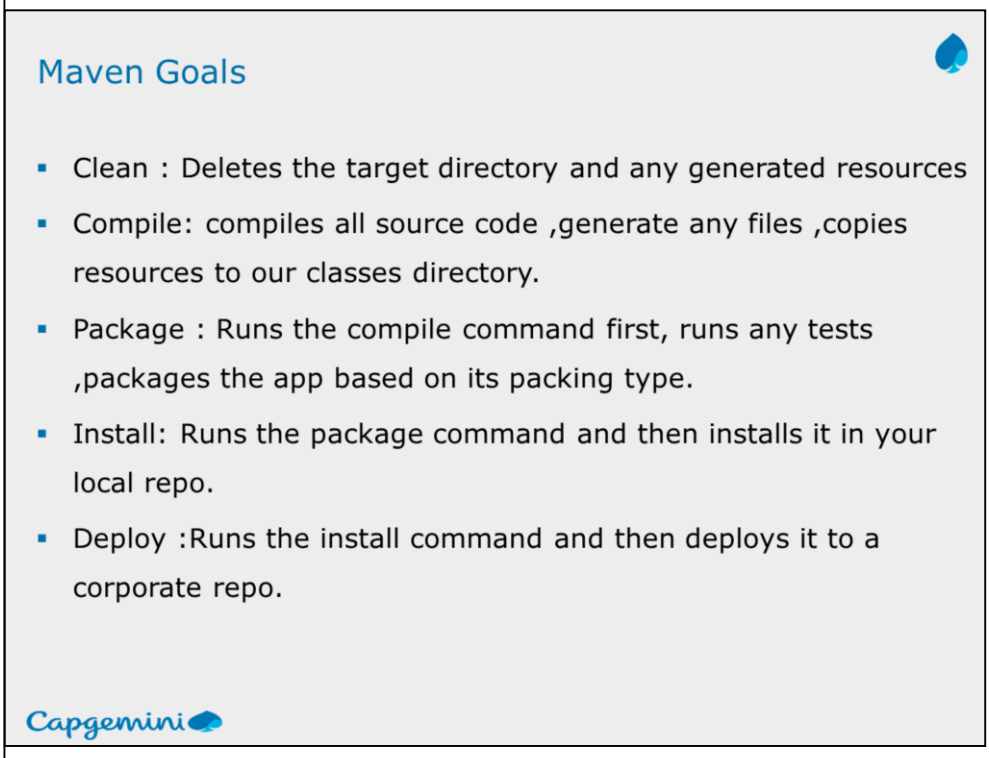
- A dependency is a reference to an specific artifact that resides in a repository.
- Dependencies are requested in a declarative fashion with dependency's coordinates by looking up in repositories.
- There are two repositories available:
  - Local Repository
    - By default, Maven creates your local repository in ~/.m2/repository
  - Remote Repository
    - Default central Maven repository: http://repo1.maven.org/maven2/



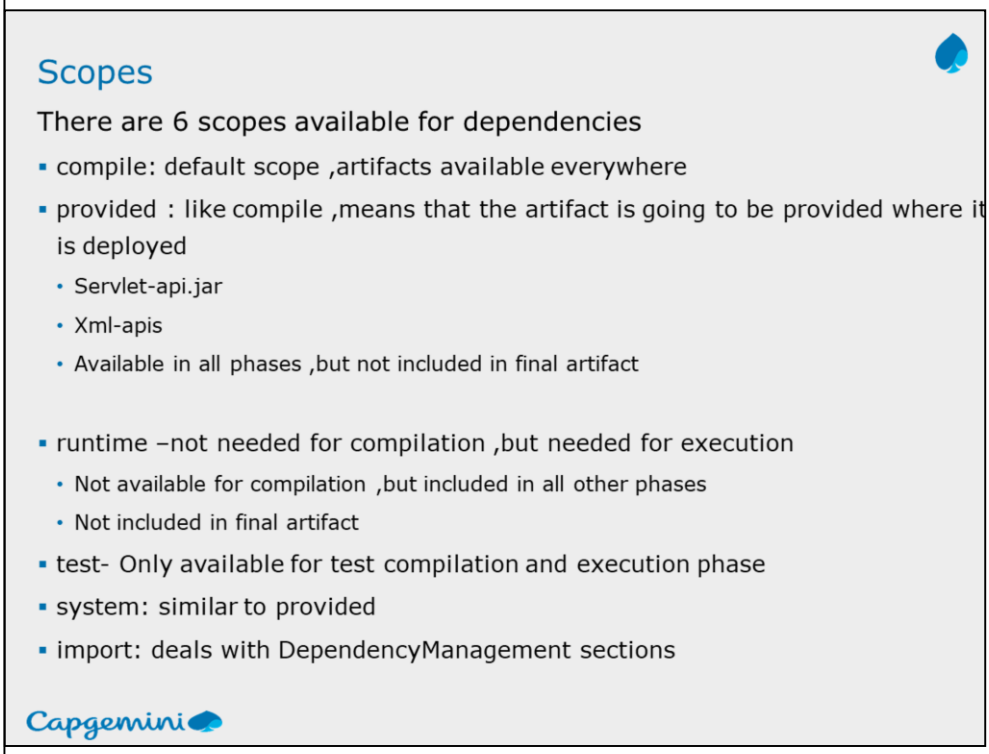
Capgemini

When a dependency is declared within the context of your project, Maven tries to satisfy that dependency by looking in repositories. Maven uses a local repository to resolve its dependencies. If not found one or more remote repositories are consulted to find a dependency. If found, the dependency is downloaded to the local repository and used from the local repository.

Instructor Notes:



Instructor Notes:



Instructor Notes:

The Compiler Plug-in

Used to compile code and test code .

Invokes Javac, but with the classpath set from the dependencies

Defaults to Java 1.5 regardless of what JDK is installed

Configuration Section allows customization

- Includes/Excludes
- Fork
- Memory
- Source/Target

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>2.5.1</version>
    <configuration>
        <fork>true</fork>
        <meminitial>128m</meminitial>
        <maxmem>512m</maxmem>
        <source>1.7</source>
        <target>1.7</target>
    </configuration>
</plugin>
```

Capgemini

Instructor Notes:

Source Plug-in

- The Source Plugin creates a jar archive of the source files of the current project. The jar file is, by default, created in the project's target directory.
- Tied to the package phase
  - Often overridden to later phase

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-source-plugin</artifactId>
    <version>2.2.1</version>
    <executions>
        <execution>
            <id>attach-sources</id>
            <phase>verify</phase>
            <goals>
                <goal>jar</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

Capgemini

Instructor Notes:

Jar Plug-in

- Used to package code into jar
- Tied to the package phase
- Configuration section allows customization
  - Includes/Excludes
  - Manifest

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-jar-plugin</artifactId>
            <version>2.4</version>
            <configuration>
                <useDefaultManifestFile>true</useDefaultManifestFile>
            </configuration>
        </plugin>
    </plugins>
</build>
```

Capgemini

Instructor Notes:

Answers for the Review Questions:
Question 1: Option 1 and 2
Question 2: option 2