



Introduction to Maven



Agenda

1

Introduction to Maven

2

Maven POM

3

Maven Fundamentals and Concepts

4

Maven Configuration and Dependencies

5

Maven Repository

Objectives

At the end of this module, you will be able to :

- Learn about Maven, its features and usage as a build tool
- Install MAVEN and set up the environment to run sample builds
- Configure & build projects using maven

Introduction to Maven



Introduction – History of Maven

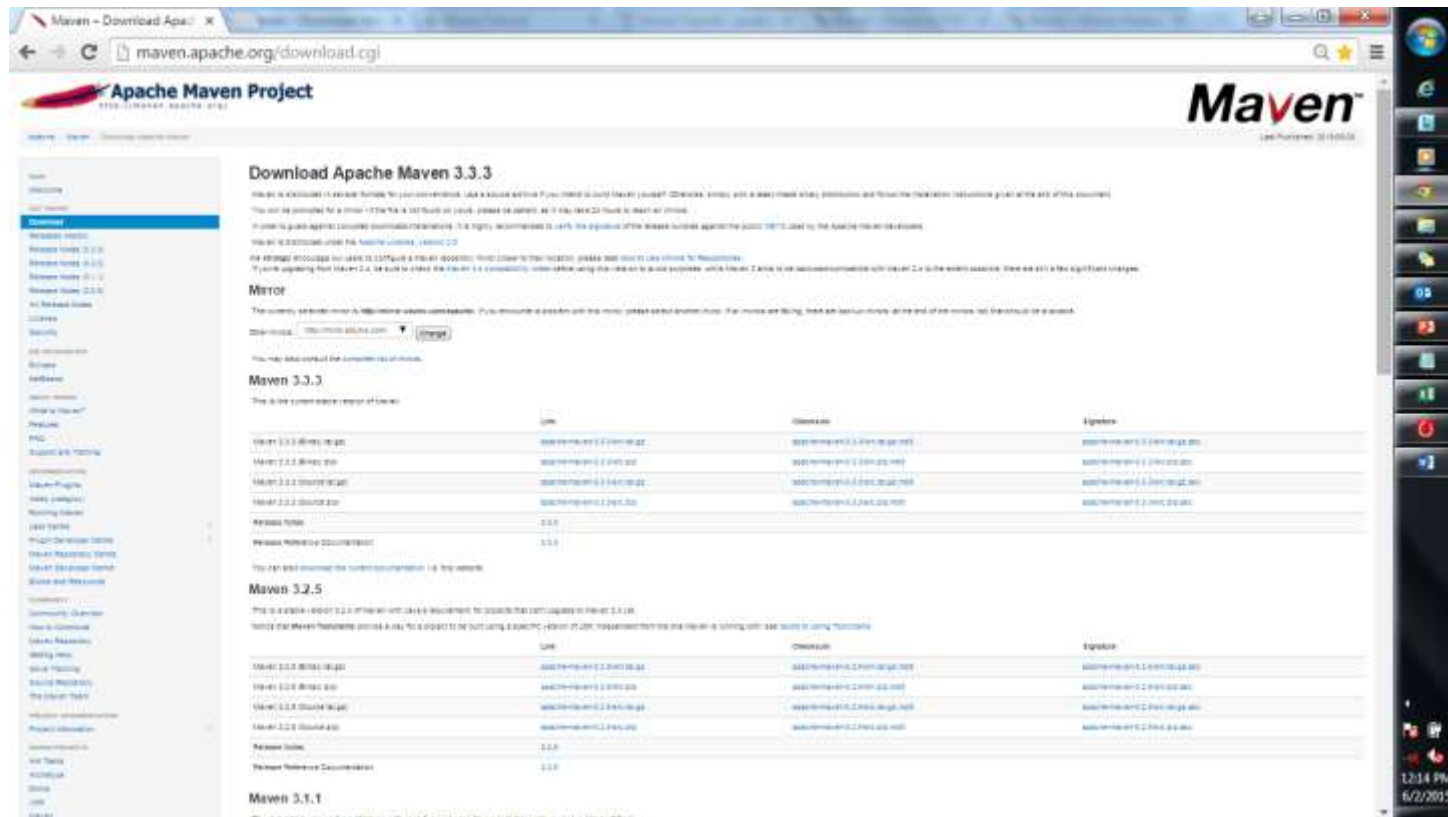
- Maven is a **build and software management tool**
- Maven was first used in Jakarta Alexandria project in August 2001
- Later it was designed for building project under Jakarta Turbine project by ***Jason van Zyl*** to overcome the multiple builds developed using ANT
- The first beta version launched, was in 2002 march

Introduction – Features of Maven

- Maven 's primary objective is to give a project model which is Reusable and Maintainable
- Maven provides the following solution to the developers
 - Builds including Compilation
 - Dependencies
 - Documentation and Reporting
 - Software Configuration Management
 - Releases
 - Distribution and
 - Mailing list
- Maven provides a simplified project build process by handling most of build phases

Maven Environment set up

- Maven can be set up using 3 simple steps
 - Download Maven from apache site
 - Use the below link to download maven
- <http://maven.apache.org/download.cgi>



Maven Environment set up (cont)

- **Step 1: Download Maven from apache site (cont...)**
 - Version can be decided based on the java version of project.

Recent List of Maven versions compatible with Java

Maven Version	Date of release	Java Compatible Version
2.1.0	March-09	Java 1.4
3.1.1	October-13	Java 5
3.2.5	December-14	Java 6
3.3.3	April-15	Java 7

- Extract the downloaded jar into the required directory
E.g: E:\Maven\apache-maven-3.3.1

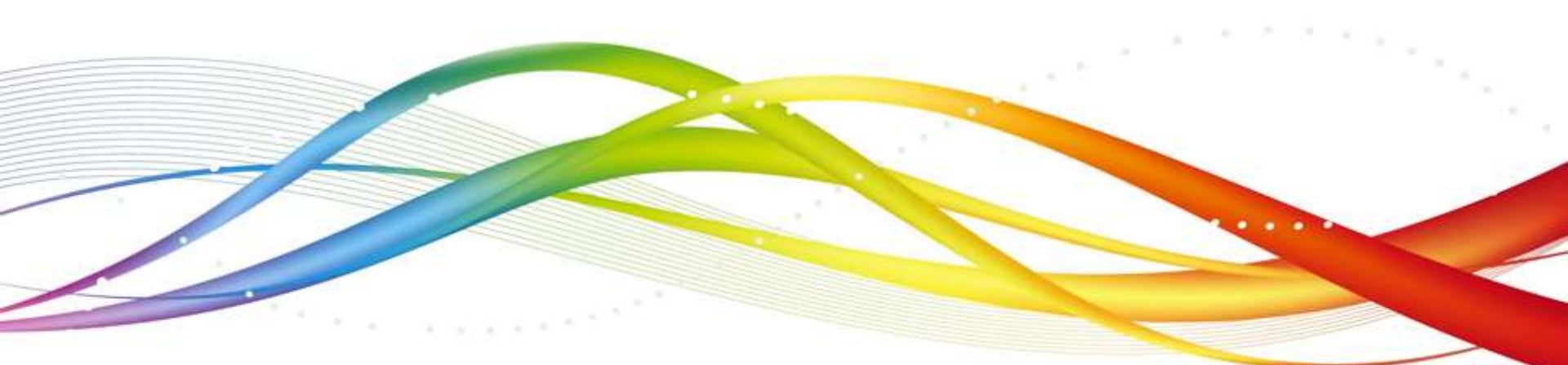
Maven Environment set up (cont)

- Pre-request: compatible jdk should be installed and path should have been set
- **Step 2: Set environment variables for Maven**
 - M2_HOME
 - Set the installed location of maven
 - Eg: M2_HOME = E:\Maven\apache-maven-3.3.1
 - M2
 - add M2(ie maven bin directory) to system PATH variable
 - Eg: M2 = E:\Maven\apache-maven-3.3.1\bin
- **Step 3: verify installation**
 - mvn - - version
 - command is used to check the installation.



```
C:\Windows\system32\cmd.exe
E:\meven-demo>mvn --version
Apache Maven 3.3.1 (cab6659f9874fa96462afef40fcf6bc033d58c1c; 2015-03-14T01:40:27+05:30)
Maven home: E:\Maven\apache-maven-3.3.1
Java version: 1.7.0_55, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.7.0_55\jre
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 7", version: "6.1", arch: "x86", family: "windows"
E:\meven-demo>
```

Maven POM



Maven POM

- Maven POM (Project Object Model) is the base model of complete Maven System
- It is an xml file named ***pom.xml***
- Few important tasks *pom* file declares are
 - the structure and the contents of the Maven Project
 - the goals and plugins
 - the configuration details to build the project
- *pom.xml* is placed under the base directory of the project
- POM gets classified as
 - POM – created by user for the current build
 - Super POM – this the base model on which user POM is created

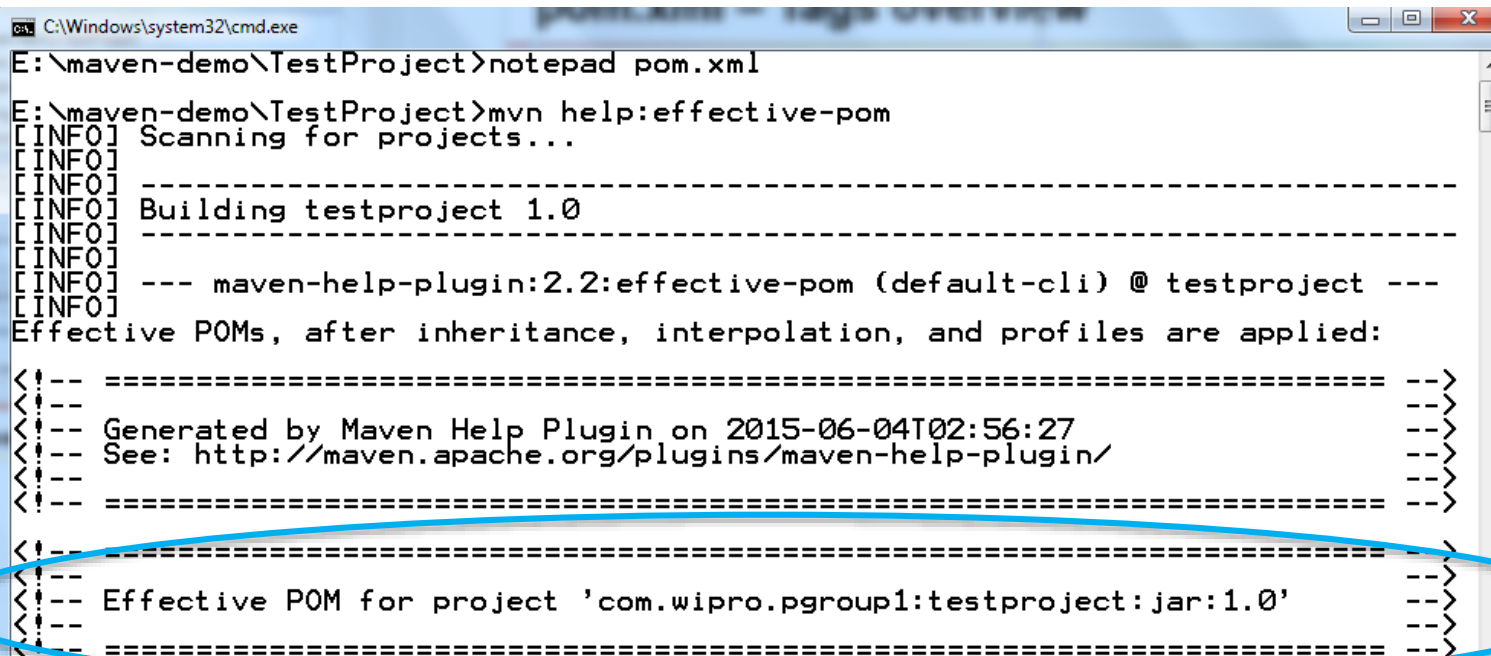
pom.xml – Tags an overview

- The root element of pom.xml is `<project> </project>`, this predominantly defines the namespace and maven schema.
- The three mandate sub elements are
 - `groupId` – used to represent a unique project category id in the organization
 - `artifactId` – used to represent a unique project name under the organization project category
 - `version` – used to give the current build version

```
<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.wipro.pgroup1</groupId>
<artifactId>testproject</artifactId>
<version>1.0</version>
</project>
```

pom.xml – Tags overview

- Create a folder called maven-demo\TestProject
- Create a file called pom.xml under maven-demo\TestProject
- Write the example script from previous slide
- To get the pom build running. Issue the below command
`mvn help:effective-pom`



```
C:\Windows\system32\cmd.exe
E:\maven-demo\TestProject>notepad pom.xml
E:\maven-demo\TestProject>mvn help:effective-pom
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building testproject 1.0
[INFO] -----
[INFO]
[INFO] --- maven-help-plugin:2.2:effective-pom (default-cli) @ testproject ---
[INFO]
Effective POMs, after inheritance, interpolation, and profiles are applied:
<!-- ===== -->
<!-- Generated by Maven Help Plugin on 2015-06-04T02:56:27 -->
<!-- See: http://maven.apache.org/plugins/maven-help-plugin/ -->
<!-- ===== -->
<!-- ===== -->
<!-- Effective POM for project 'com.wipro.pgroup1:testproject:jar:1.0' -->
<!-- ===== -->
```

pom.xml – Tags overview

- Maven uses repository for its intelligence
- The moment you build a project for the first time Maven downloads all supported intelligence files and jars
- So the first time run may take few minutes, and the same in next run would be in secs

```
</reporting>
</project>

[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 05:48 min
[INFO] Finished at: 2015-06-03T15:57:32+05:30
[INFO] Final Memory: 9M/21M
[INFO] -----

E:\maven-demo\TestProject>
```

```
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 15.270 s
[INFO] Finished at: 2015-06-04T14:56:28+05:30
[INFO] Final Memory: 7M/17M
[INFO] -----

E:\maven-demo\TestProject>
```

pom.xml – Tags overview

- Observe the generated pom file, you would see many tags apart from the original pom.xml, what we had written

```
pomres - Notepad
File Edit Format View Help
<!-- ===== -->
<!-- -->
<!-- Effective POM for project 'com.wipro.pgroup1:testproject:jar:1.0' -->
<!-- -->
<!-- ===== -->

<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.wipro.pgroup1</groupId>
  <artifactId>testproject</artifactId>
  <version>1.0</version>
  <repositories>
    <repository>
      <snapshots>
        <enabled>false</enabled>
      </snapshots>
      <id>central</id>
      <name>Central Repository</name>
      <url>https://repo.maven.apache.org/maven2</url>
    </repository>
  </repositories>
  <pluginRepositories>
    <pluginRepository>
      <releases>
        <updatePolicy>never</updatePolicy>
      </releases>
      <snapshots>
        <enabled>false</enabled>
      </snapshots>
      <id>central</id>
      <name>Central Repository</name>
      <url>https://repo.maven.apache.org/maven2</url>
    </pluginRepository>
  </pluginRepositories>
  <build>
    <sourceDirectory>E:\maven-demo\TestProject\src\main\java</sourceDirectory>
```

Detailed output of mvn *help:effective-pom* is listed in the notes page

pom.xml – Tags overview

- The tags on the previous slide are inherited from the super pom
- just like toString, clone or other methods from Object class that is by default inherited into every class we create
- *mvn help:effective-pom* command helps in identifying the super pom
- It also helps us in checking the default settings of the build
- To change the default values, override the specific tags in your created pom.xml file

Maven Fundamentals and Concepts



Maven Fundamentals

- Maven creates and manages projects in form of Maven Lifecycle
- Lifecycle is the process of building and distribution that an artifact goes through
- The Process is broken down as phases
- A lifecycle completes only when all the phases execute successfully
- Maven's major lifecycles are
 - Build (default)
 - Clean
 - Site
- The default in maven is the build lifecycle which handles the deployment of the project
- Clean lifecycle handles cleaning of project
- Site lifecycle handles the site documentation

Maven Fundamentals

- Maven's build lifecycle phases are
 - validate – checks project correctness and availability of necessary information
 - compile – compiles the project's source code
 - test – using suitable test framework, it tests the project
 - package – takes care of packaging and distribution, such as a JAR.
 - integration-test – checks the package in integration test environment
 - verify – checks the validity and quality of the project package
 - install - installs the package into the local repository
 - deploy - copies the package to the remote repository for sharing with other developers and projects, generally in integration or release environment

Maven Concepts

- To run maven, its important to understand some of the core concepts
 - Pom.xml
 - Maven Plugins and Goals
 - Dependency
 - Repository
-
- Pom.xml is the fundamental module and it contains information about the project details
 - We have already visited this in detail

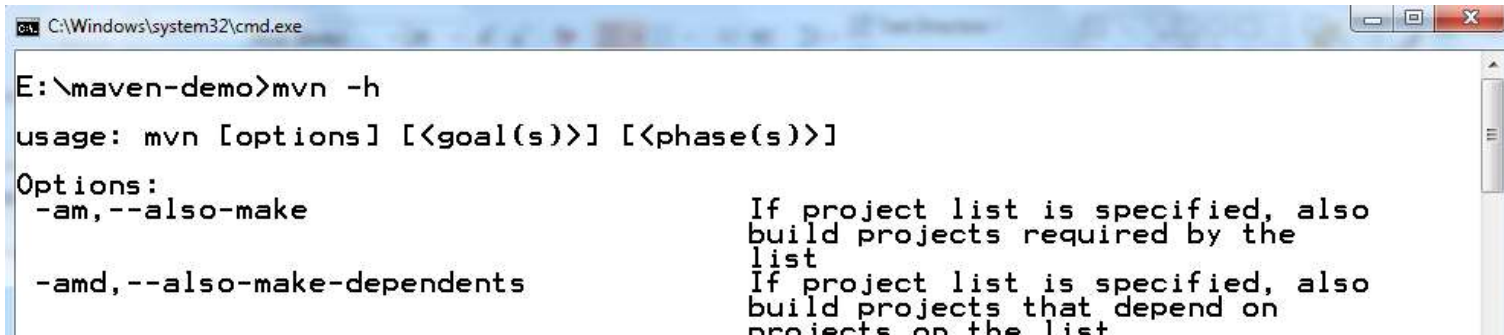
Maven Concepts : Maven Plugins and Goals

- Plugin in maven is an execution framework
- It contains goals to be executed
- Build and reporting are the 2 types of plugins

Core plugins		Packaging types/tools		Reporting plugins		Tools		IDEs	
Plugin Name	Type	Plugin Name	Type	Plugin Name	Type	Plugin Name	Type	Plugin Name	Type
clean	B	jar	B	Changes	B+R	ant	B	eclipse	B
deploy	B	ejb	B	project-info-reports	R	archetype	B		
site	B	war	B	javadoc	B+R	SCM	B		

Maven Concepts : Maven Plugins and Goals (commands)

- Maven commands are life cycle driven commands
- These help in complete project management
- eg: mvn compile
- mvn -h or mvn --help: is used to get help information about maven



```
C:\Windows\system32\cmd.exe
E:\maven-demo>mvn -h
usage: mvn [options] [<goal(s)>] [<phase(s)>]

Options:
  -am,--also-make                If project list is specified, also
                                build projects required by the
                                list
  -amd,--also-make-dependents    If project list is specified, also
                                build projects that depend on
                                projects on the list
```

- To run plugin we have to give the command pluginName:goal name
- eg: mvn archetype:generate
- Here archetype is the plugin and generate is the goal

Maven Dependency

- Dependency management is one of the best feature of Maven
- Maven supports up to transitive dependency level
- In our example we have not added JUNIT jar but Junit test classes have run and results given
- This is maven's dependency management
- Maven repository contains Junit jar along with junit pom file for the dependency settings of junit

Maven Dependency

- In case of more than one project integration with various version differences
- There may be multiple copies of same supporting jar but with different versions
- Then it can be explicitly defined which jar to be used and which to be excluded

```
<dependencyManagement>
<dependencies>
<dependency>
    <groupId>group-a</groupId>
    <artifactId>artifact-a</artifactId>
    <version>1.0</version>
    <exclusions>        <exclusion>
        <groupId>group-c</groupId>
        <artifactId>excluded-artifact</artifactId>
    </exclusion>        </exclusions>
    </dependency>
</dependencies>
</dependencyManagement>
```


Maven Dependency Scope

- Maven has 6 dependency scope
 - Compile: this is the default scope

Referenced user-defined dependencies like classes are available in the classpath during compilation

- Provided: dependencies provided by the system, like J2EE jars are referred under provided scope
- Runtime: these are dependencies required only during runtime and not for compilation
- Test: these dependencies are required only during testing phase
- System: these are dependencies provided by the system but specified by user explicitly
- Import: recently added from Maven 2.0.9 onwards and applicable for pom files

Maven Repository

- In Maven *Repository* is a place to hold the build and dependency related data(project artifacts)
- There 2 types
 - Local
 - Remote
- Local Repository : this is created by the system to hold
 - the remote supporting jars downloaded as local cache
 - build artifacts before launch
- In the default .m2 location under the home directory.

<!-- localRepository | The path to the local repository maven will use to store artifacts. | | Default: \${user.home}/.m2/repository

<localRepository>/path/to/local/repo</localRepository> -->

- You can check the local repository location from M2-home/conf/system.xml file

Maven Repository

- Local repository location can be altered by changing the value of the tag `localRepository`

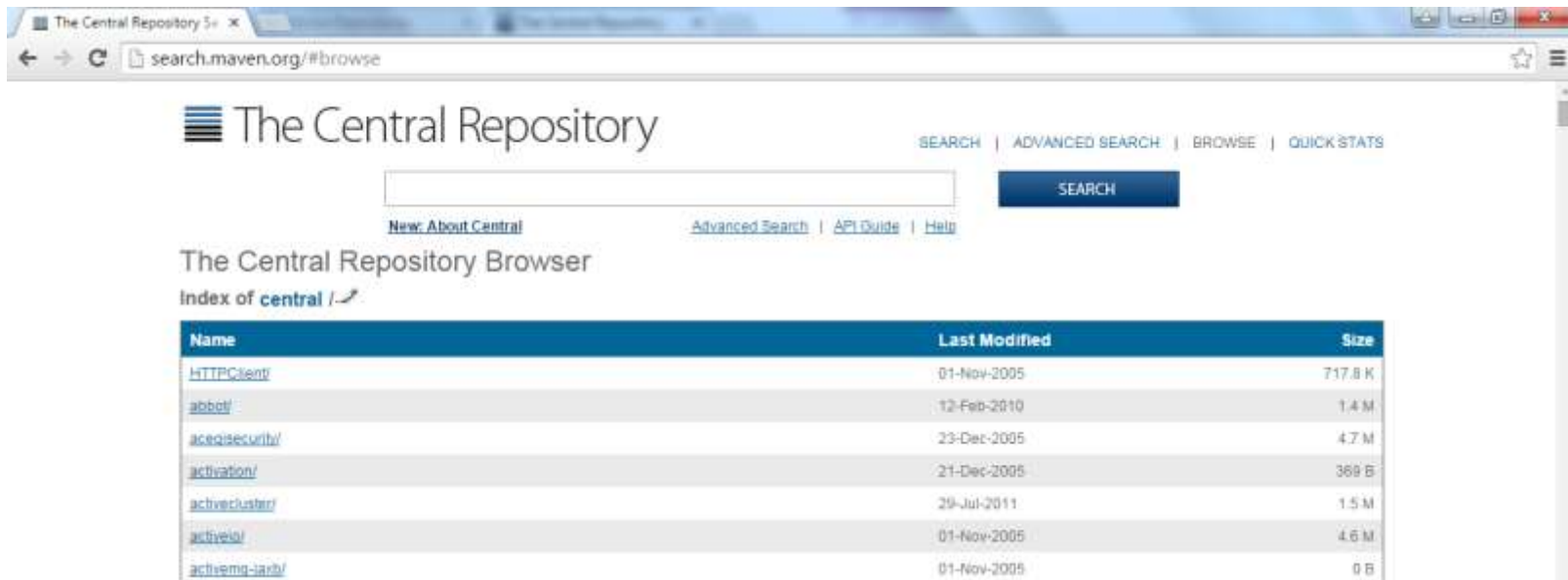
`<!-- localRepository | The path to the local repository maven will use to store artifacts. | | Default: ${user.home}/.m2/repository`

`<localRepository>c:/maven/repo</localRepository> -->`

- Remote Repository:
- When related artifacts are to be loaded from remote or on non local repository location, it is referred from remote repository
- Maven Community provides a repository called Central Repository
- This is a centralized place where all projects load their downloadable artifacts
- It contains many commonly used libraries like JBOSS, Apache etc
- The contents of Central Repository can be found at <http://search.maven.org/#browse>

Maven Repository

- The Central Repository acts as the default repository for many build tools like Apache Maven, SBT etc
- It can be easily used from Apache Ant/Ivy, Gradle like build systems too



Maven Repository

- When dependency is required maven searches first the local repository
- If the required artifacts are not found then it searches the Central Repository
- If the artifact is not found there too then build fails with errors
- In case the required artifact is loaded in a private specific location then it can be provided using the below tag in the pom.xml file

```
<repositories>
  <repository>
    <id>companyname.lib1</id>
    <url>http://download.wipro.com/maven2/mylib1</url>
  </repository>
</repositories>
```

Maven First Project

- Let us create a small maven Project
- First step is to generate the project structure
- Plugin : archetype is used to generate directories of the project structure
- Archetype contains thousands of predefined java related project structures
- choosing one from the list enables maven to create the complete directory structure with necessary starter files for the project
- archetype plugin has 4 direct goals and 3 build related goals

Maven First Project (cont...)

Few archetype goals:

- `archetype:generate`
 - creates a Maven project from an archetype
 - asks the user to choose an archetype from the archetype catalog
 - retrieves it from the remote repository and create a working Maven project
- `archetype:create-from-project`
 - creates an archetype from an existing project
- `archetype:jar`
 - This is bound to the package phase
 - is used to build the archetype jar artifact
- `archetype:update-local-catalog`
 - This is bound to the install phase
 - is used to update the local catalog

Maven First Project (cont...)

- Create a folder called FirstProject, which is the applications root
- From root directory issue the archetype:generate command
- First time when you run this command takes time as it would load all the dependencies from the remote
- It would take few minutes

```
[INFO] No archetype defined. Using maven-archetype-quickstart (org.apache. -  
maven.archetypes:maven-archetype-quickstart:1.0)
```

```
Choose archetype:
```

```
...
```

```
312: remote -> org.apache.maven.archetypes:maven-archetype-quickstart (An -  
archetype which contains a sample Maven project.)
```

```
Choose a number or apply filter (format: [groupId:]artifactId, case -  
sensitive contains): 312:
```

- You could see the reference type as remote

Maven First Project (cont...)

- When similar projects have been run previously then the related artifacts would have been downloaded to the local repository

```
C:\workspace\demo2\cmd.exe - run archetype:generate
E:\maven-demo>mkdir FirstProject
E:\maven-demo>cd FirstProject
E:\maven-demo\FirstProject>mvn archetype:generate
[INFO] Scanning for projects...
[INFO] Building Maven Stub Project (No POM) 1
[INFO]
[INFO] >>> maven-archetype-plugin:2.3:generate (default-cli) > generate-sources
[INFO] standalone-pom >>>
```

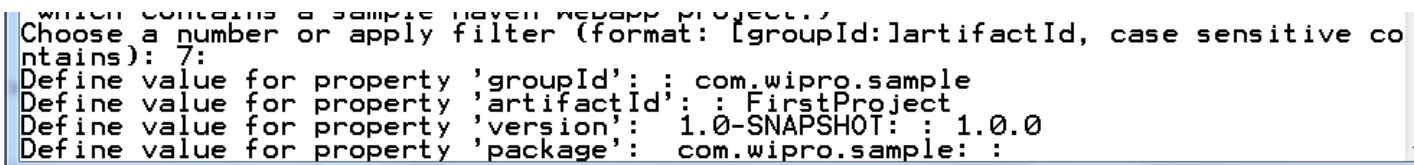
```
type which contains a sample Maven plugin site.
This archetype can be layered upon an existing Maven plugin project.)
5: internal -> org.apache.maven.archetypes:maven-archetype-portlet (An archetype
which contains a sample JSR-268 Portlet.)
6: internal -> org.apache.maven.archetypes:maven-archetype-profiles ()
7: internal -> org.apache.maven.archetypes:maven-archetype-quickstart (An archet
ype which contains a sample Maven project.)
8: internal -> org.apache.maven.archetypes:maven-archetype-site (An archetype wh
ich contains a sample Maven site which demonstrates
some of the supported document types like APT, XDoc, and FML and demonstra
tes how
to i18n your site. This archetype can be layered upon an existing Maven pr
oject.)
9: internal -> org.apache.maven.archetypes:maven-archetype-site-simple (An arche
type which contains a sample Maven site.)
10: internal -> org.apache.maven.archetypes:maven-archetype-webapp (An archetype
which contains a sample Maven Webapp project.)
Choose a number or apply filter (format: [groupId:]artifactId, case sensitive co
ntains): 7: _
```

Reference is from local

- Choose archetype: Just press enter to choose the default project

Maven First Project (cont...)

- As we have not provided any pom.xml maven generates it
- Maven asks details of groupId, artifactId, version and package



```
which contains a sample Maven webapp project.  
Choose a number or apply filter (format: [groupId:artifactId, case sensitive co  
ntains): 7:  
Define value for property 'groupId': : com.wipro.sample  
Define value for property 'artifactId': : FirstProject  
Define value for property 'version': 1.0-SNAPSHOT: : 1.0.0  
Define value for property 'package': com.wipro.sample: :
```

- Maven gives a default version value 1.0-SNAPSHOT, if required we can change it
- Here we have changed it to 1.0.0
- For package it gives a default value as of groupId
- Here we have not changed the package value

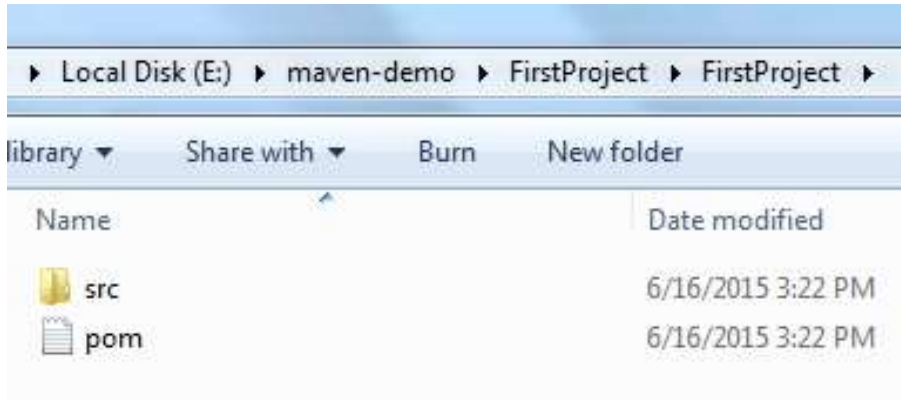
Maven First Project (cont...)

- Added details are shown for confirmation
- When enter key is hit, it builds the project

```
which contains a sample maven webapp project.)
Choose a number or apply filter (format: [groupId:]artifactId, case sensitive co
ntains): 7:
Define value for property 'groupId': : com.wipro.sample
Define value for property 'artifactId': : FirstProject
Define value for property 'version': 1.0-SNAPSHOT: : 1.0.0
Define value for property 'package': com.wipro.sample: :
Confirm properties configuration:
groupId: com.wipro.sample
artifactId: FirstProject
version: 1.0.0
package: com.wipro.sample
Y: :
[INFO] -----
[INFO] Using following parameters for creating project from Old (1.x) Archetype:
maven-archetype-quickstart:1.1
[INFO] -----
[INFO] Parameter: groupId, Value: com.wipro.sample
[INFO] Parameter: packageName, Value: com.wipro.sample
[INFO] Parameter: package, Value: com.wipro.sample
[INFO] Parameter: artifactId, Value: FirstProject
[INFO] Parameter: basedir, Value: E:\maven-demo\FirstProject
[INFO] Parameter: version, Value: 1.0.0
[INFO] project created from Old (1.x) Archetype in dir: E:\maven-demo\FirstProje
ct\FirstProject
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 03:23 min
[INFO] Finished at: 2015-06-16T15:22:13+05:30
[INFO] Final Memory: 11M/26M
[INFO] -----
E:\maven-demo\FirstProject>
```

Maven First Project (cont...)

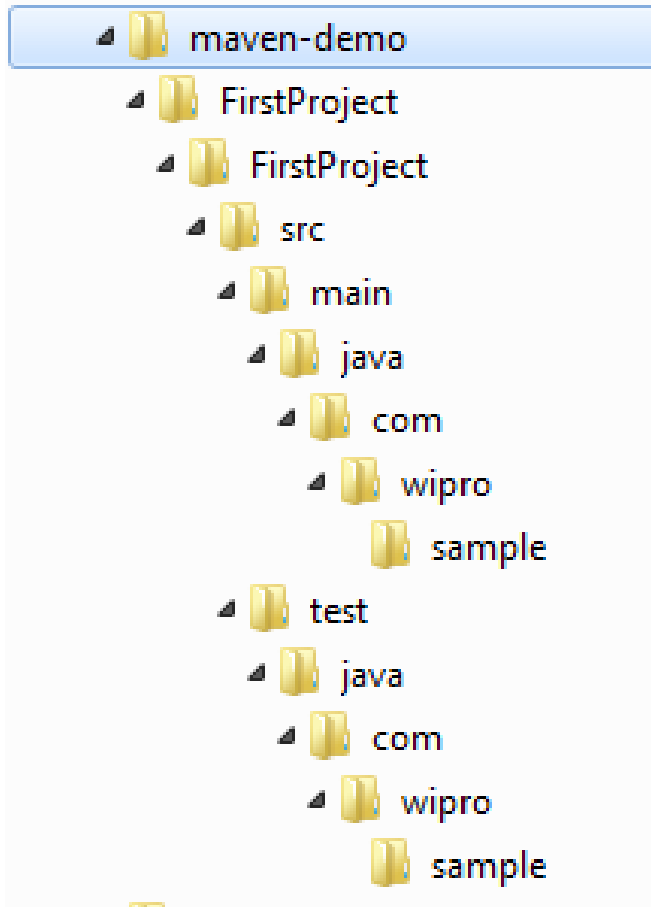
- Now the project is generated, look at the FirstProject Directory
- We can see a folder with the artifactId



- Source folder and pom file is created automatically by maven
- Supporting files and test document is also created

Maven First Project (cont...)

- Created Folder Structure

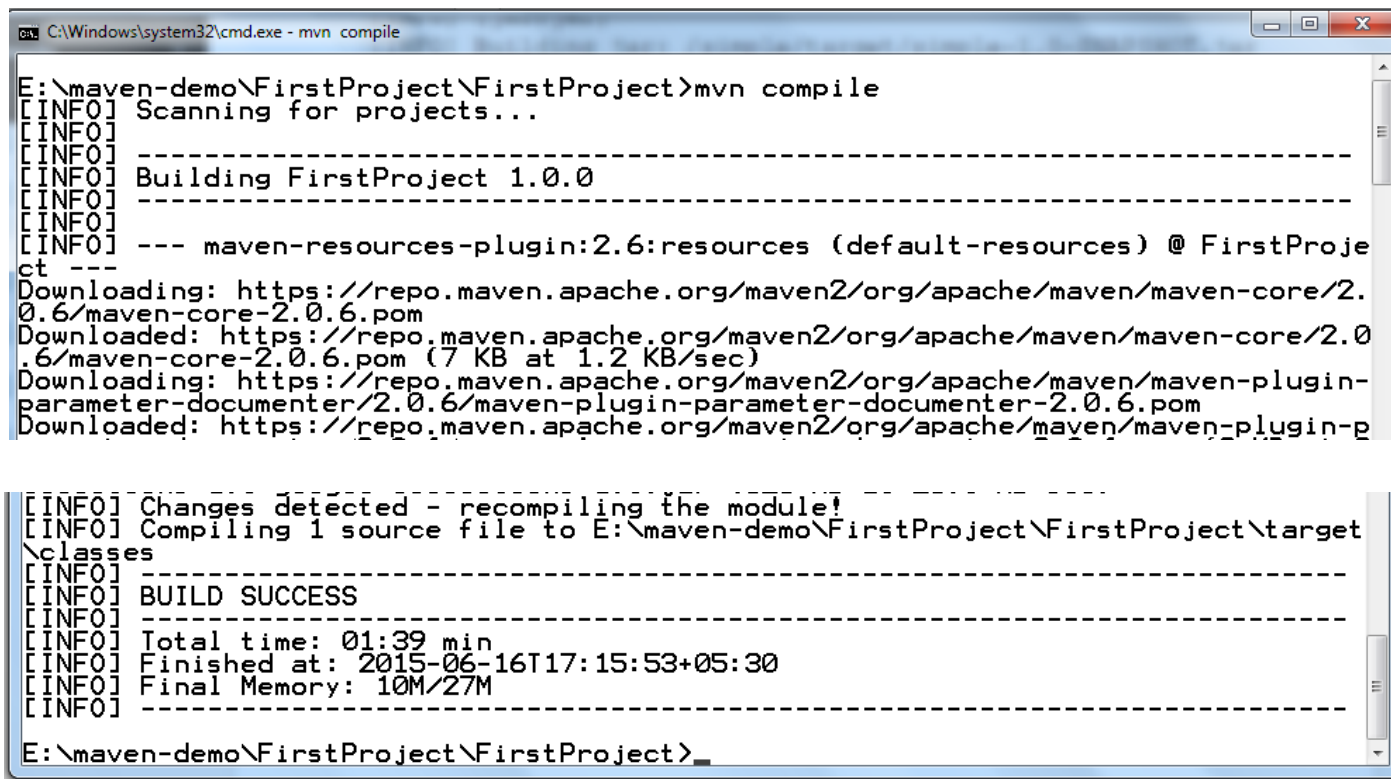


Maven creates all dependencies including JUNIT test files for the current project

Maven First Project (cont...)

- Maven creates a sample main class called App.java containing main method
- AppTest.java Junit Test class for Testing purpose
- To compile we'll issue

mvn compile

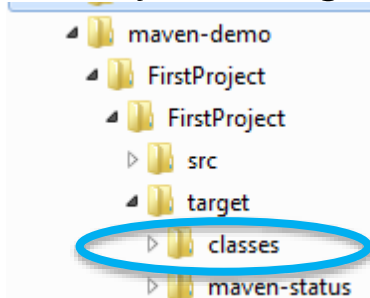


```
C:\Windows\system32\cmd.exe - mvn compile

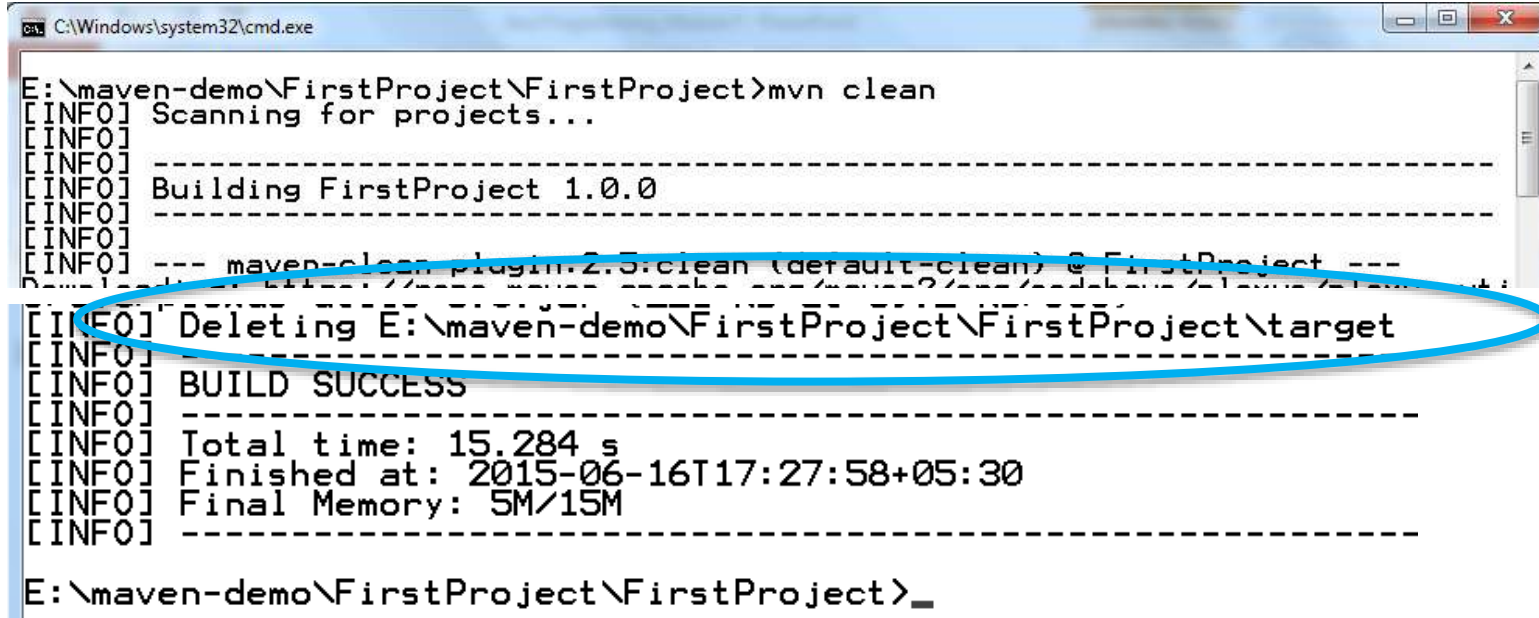
E:\maven-demo\FirstProject\FirstProject>mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building FirstProject 1.0.0
[INFO] -----
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ FirstProject ---
[INFO] Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/maven-core/2.0.6/maven-core-2.0.6.pom
[INFO] Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/maven-core/2.0.6/maven-core-2.0.6.pom (7 KB at 1.2 KB/sec)
[INFO] Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/maven-plugin-parameter-documenter/2.0.6/maven-plugin-parameter-documenter-2.0.6.pom
[INFO] Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/maven-plugin-parameter-documenter/2.0.6/maven-plugin-parameter-documenter-2.0.6.pom
[INFO]
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to E:\maven-demo\FirstProject\FirstProject\target\classes
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO]
[INFO] Total time: 01:39 min
[INFO] Finished at: 2015-06-16T17:15:53+05:30
[INFO] Final Memory: 10M/27M
[INFO]
[INFO] -----
E:\maven-demo\FirstProject\FirstProject>
```

Maven First Project (cont...)

- mvn compile compiles the project files to confirm check the project directory for target folder with .class files



- mvn clean – helps to clear all build created files and folders, changing what we have done leading to empty folders etc

A screenshot of a Windows command prompt window titled 'C:\Windows\system32\cmd.exe'. The command 'mvn clean' has been executed in the directory 'E:\maven-demo\FirstProject\FirstProject'. The output shows the Maven build process, including scanning for projects, building 'FirstProject 1.0.0', and executing the 'maven-clean-plugin:2.5:clean' goal. A line in the output, 'Deleting E:\maven-demo\FirstProject\FirstProject\target', is circled in blue. The process concludes with 'BUILD SUCCESS', a total time of 15.284 seconds, and a final memory usage of 5M/15M.

```
E:\maven-demo\FirstProject\FirstProject>mvn clean
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building FirstProject 1.0.0
[INFO] -----
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ FirstProject ---
[INFO] Deleting E:\maven-demo\FirstProject\FirstProject\target
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 15.284 s
[INFO] Finished at: 2015-06-16T17:27:58+05:30
[INFO] Final Memory: 5M/15M
[INFO] -----
E:\maven-demo\FirstProject\FirstProject>
```

Maven First Project (cont...)

- mvn install creates project jar file

```
C:\Windows\system32\cmd.exe - mvn install
E:\maven-demo\FirstProject\FirstProject>mvn install
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building FirstProject 1.0.0
[INFO] -----
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ FirstProject ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory E:\maven-demo\FirstProject\FirstProject\src\main\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ FirstProject ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to E:\maven-demo\FirstProject\FirstProject\target\classes
```

Compilation

```
-----
TESTS
-----
Running com.wipro.sample.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.052 sec

Results :

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

Testing

```
[INFO] Installing E:\maven-demo\FirstProject\FirstProject\target\FirstProject-1.0.0.jar to C:\Users\surajes\.m2\repository\com\wipro\sample\FirstProject\1.0.0\FirstProject-1.0.0.jar
[INFO] Installing E:\maven-demo\FirstProject\FirstProject\pom.xml to C:\Users\surajes\.m2\repository\com\wipro\sample\FirstProject\1.0.0\FirstProject-1.0.0.pom
[INFO] BUILD SUCCESS
[INFO]
[INFO] -----
[INFO] Total time: 42.919 s
[INFO] Finished at: 2015-06-16T17:42:01+05:30
[INFO] Final Memory: 8M/21M
[INFO]
E:\maven-demo\FirstProject\FirstProject>
```

Creating jar file

Maven First Project (cont...)

- If you observe we gave *mvn install* but the system took care of compilation, Test and then install.
- Lifecycle phases are depended on each other
- Thus when creating jar if the project is not compiled and tested, system does that
- Maven create can also be given using deprecated *create* goal of *archetype* using any of the following syntax

```
mvn archetype:create
```

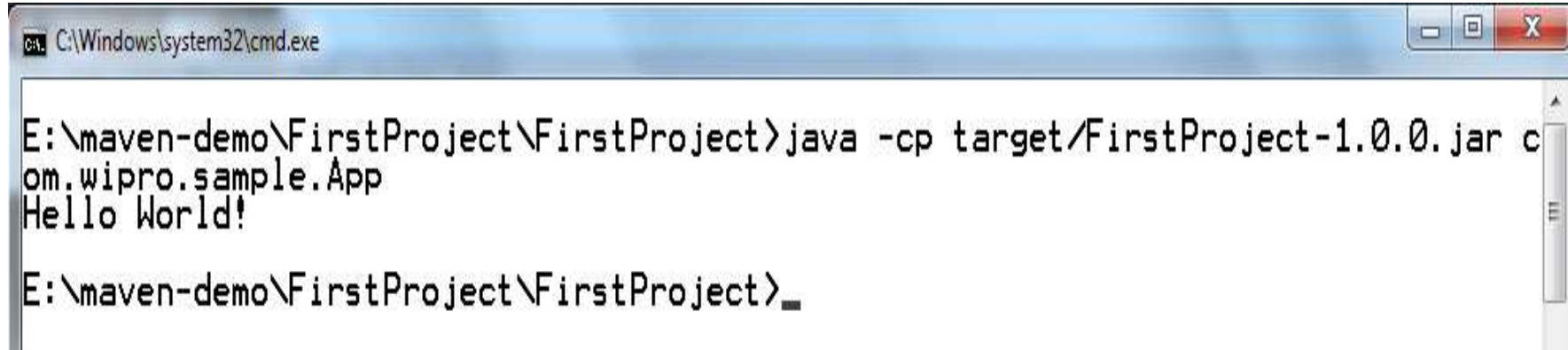
```
-DgroupId=<value>  
-DartifactId=<value>  
-Dversion=<value>  
-DpackageName=<value>  
-DarchetypeArtifactId=<value>
```

```
mvn archetype:create
```

```
-DgroupId=<value>  
-DartifactId=<value>
```

Maven First Project (cont...)

- To test the created jar , we can execute using java -cp command

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Windows\system32\cmd.exe'. The command prompt shows the current directory as 'E:\maven-demo\FirstProject\FirstProject'. The user has entered the command 'java -cp target\FirstProject-1.0.0.jar com.wipro.sample.App', and the output is 'Hello World!'. The prompt is now waiting for the next command.

```
C:\Windows\system32\cmd.exe

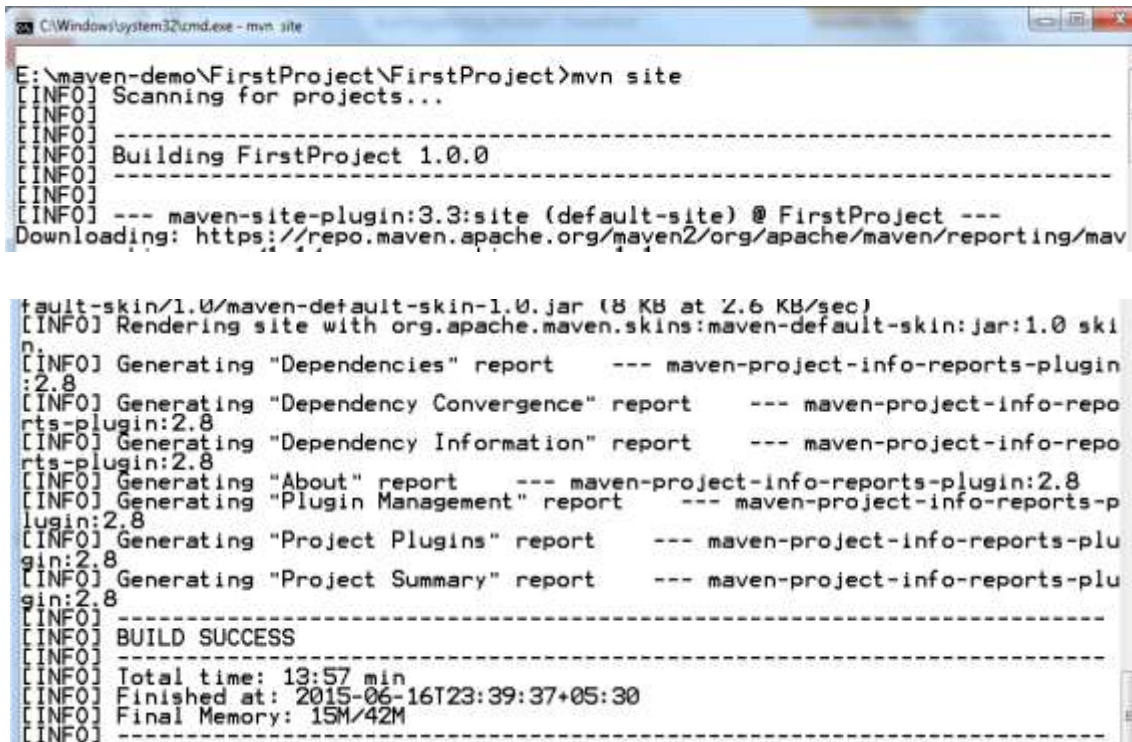
E:\maven-demo\FirstProject\FirstProject>java -cp target\FirstProject-1.0.0.jar c
om.wipro.sample.App
Hello World!

E:\maven-demo\FirstProject\FirstProject>_
```

Maven First Project (cont...)

- Create site documentation for the current project

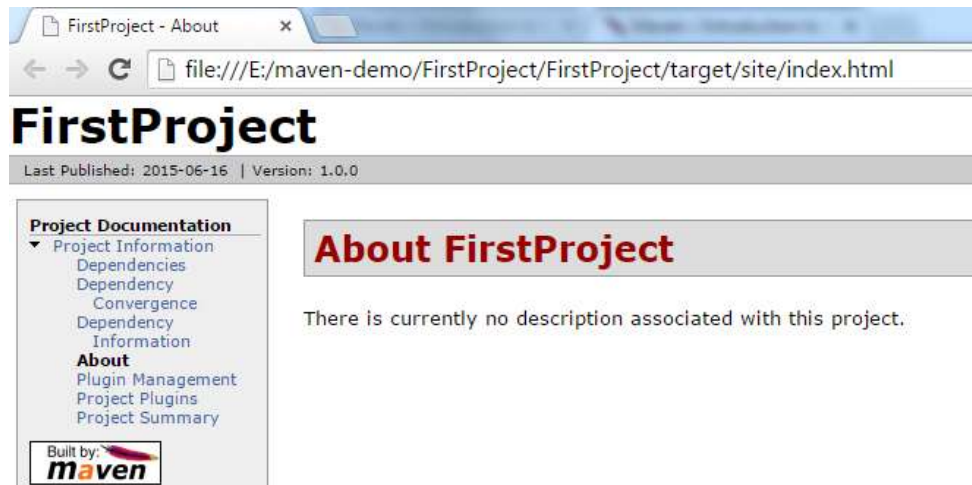
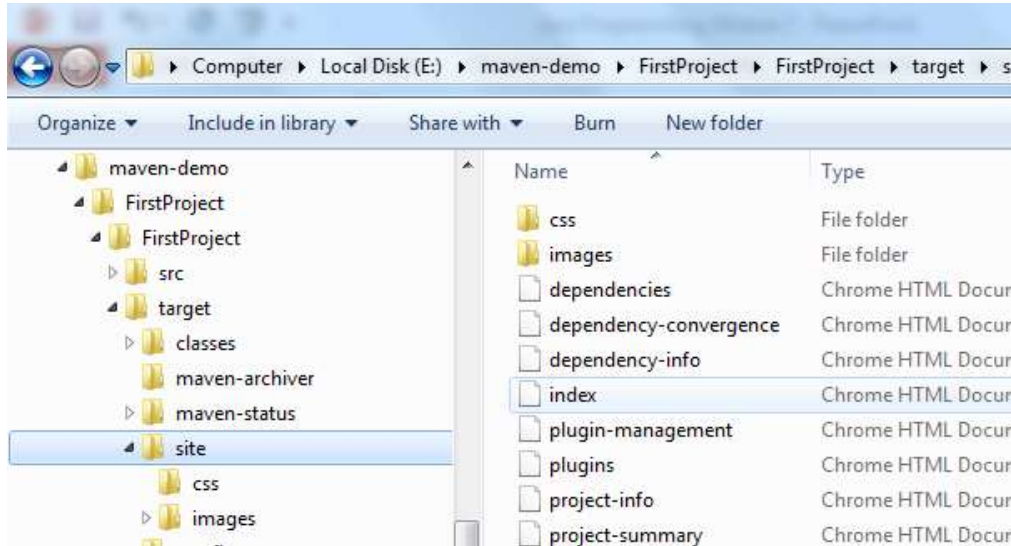
mvn site



```
C:\Windows\system32\cmd.exe - mvn_site
E:\maven-demo\FirstProject\FirstProject>mvn site
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building FirstProject 1.0.0
[INFO] -----
[INFO]
[INFO] --- maven-site-plugin:3.3:site (default-site) @ FirstProject ---
[INFO] Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/reporting/maven-
fault-skin/1.0/maven-default-skin-1.0.jar (8 KB at 2.6 KB/sec)
[INFO] Rendering site with org.apache.maven.skins:maven-default-skin:jar:1.0 ski
n
[INFO] Generating "Dependencies" report    --- maven-project-info-reports-plugin
:2.8
[INFO] Generating "Dependency Convergence" report    --- maven-project-info-repo
rts-plugin:2.8
[INFO] Generating "Dependency Information" report    --- maven-project-info-repo
rts-plugin:2.8
[INFO] Generating "About" report    --- maven-project-info-reports-plugin:2.8
[INFO] Generating "Plugin Management" report    --- maven-project-info-reports-p
ugin:2.8
[INFO] Generating "Project Plugins" report    --- maven-project-info-reports-plu
gin:2.8
[INFO] Generating "Project Summary" report    --- maven-project-info-reports-plu
gin:2.8
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO]
[INFO] Total time: 13:57 min
[INFO] Finished at: 2015-06-16T23:39:37+05:30
[INFO] Final Memory: 15M/42M
[INFO] -----
```

Maven First Project (cont...)

- To view the create html document open the target folder-> site



ANT Vs Maven

ANT	MAVEN
It is mainly a build tool from Apache	It is mainly a project management tool from Apache
Ant doesn't have any formal conventions of project structure, so we need to provide information of the project structure in build.xml file	Maven has a convention to place source code, compiled code etc. So we don't need to provide information about the project structure in pom.xml file.
Ant is procedural, you need to provide information about what to do and when to do through code. You need to provide order.	Maven is declarative, everything you define in the pom.xml file
There is no life cycle in Ant.	There is life cycle in Maven.
It is a tool box.	It is a framework.
The ant scripts are not reusable.	The maven plugins are reusable.
It is less preferred than Maven.	It is more preferred than Ant.

Quiz

1. Maven is
 - a) an IDE
 - b) Software Architecture
 - c) a Build and Software Management Tool
 - d) a Programming Language

2. There are two types of Maven Repositories. They are:
 - a) Local and Hidden
 - b) Local and Remote
 - c) Remote and Hidden
 - d) Open and Hidden

Summary

In this session, you were able to :

- Learn about Maven, its features and usage as a build tool
- Install MAVEN and set up the environment to run sample builds
- Configure & build projects using maven

References

1. Apache Maven guide revised on 15-06-2015, from <https://maven.apache.org/guides/>
2. Tutorialpoint link from <http://www.tutorialspoint.com/maven/>



Thank You

