Maven

Lab Book

Document Revision History

Date	Revision No.	Author	Summary of Changes
01-Nov -11	1.0	Rathnajothi Perumalsamy	

Table of Contents

Getting Started

Overview

This lab book is a guided tour for learning Maven. It comprises solved examples and 'To Do' assignments. Follow the steps provided in the solved examples and work out the 'To Do' assignments given.

Setup Checklist for Maven

Here is what is expected on your machine in order for the lab to work.

Minimum System Requirements

- Intel Pentium 90 or higher (P166 recommended)
- Microsoft Windows 95, 98, or NT 4.0, 2k, XP.
- Memory: 32MB of RAM (64MB or more recommended)
- Internet Explorer 6.0 or higher
- Apache Tomcat Version 5.0.
- Apache Maven 3.0

Please ensure that the following is done:

- A text editor like Notepad or Eclipse is installed.
- JDK 1.5 is installed. (This path is henceforth referred as <java_install_dir>)
- Apache Maven is installed.(Unzip the Apache Maven distribution file downloaded from http://maven.apache.org/download.html)

Learning More (Bibliography if applicable)

- Better Builds with Maven Book by Vincent Massol & Jason van Zyl
- Apache Maven 3.0 Cookbook by Srirangan
- Apache Maven Effective Implementation by Brett Porter, Maria Odea Ching
- Apache Maven by Nicolas De loof, Arnaud Heritier

Lab 1. Getting Started With Maven

Goals	• Learn and Understand the process of
	  • Setting environment variables
	  • Configuring Maven settings
	  • Creating a simple Maven Project using commands

Time	60 minutes

1.1: Setting environmental variables

Step1: set M2_Home to Maven Installation Directory using the following command:

- **set** M2_**HOME= C:\apache-maven-version.**

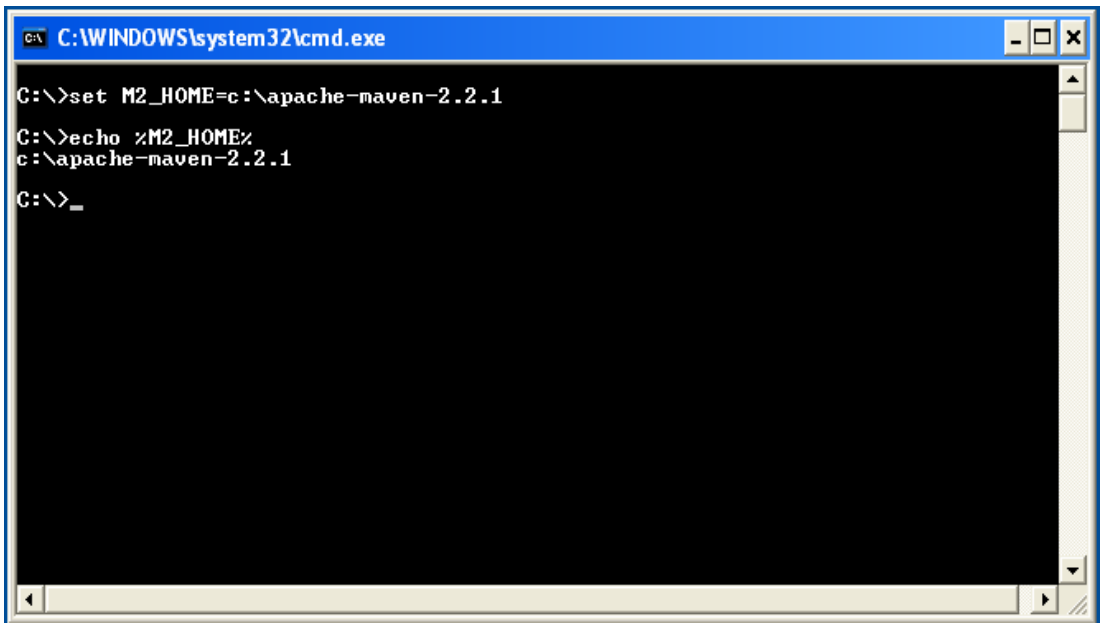


Figure 1: Environmental Variable

Step 2: Set **JAVA_HOME** to Jdk1.5 using the following command:

- **set JAVA_HOME= C:**\Program Files\Java\Jdk1.5.0.07

Step 3: Set PATH environment variable:

- **Set PATH=%PATH%;%JAVA_HOME%\bin;%M2_HOME%\bin;**

Alternatively follow the following steps for setting the environment variables

Alternate approach:

Step 1: Right click **My Computers**, and select **Properties → Environment Variables**.

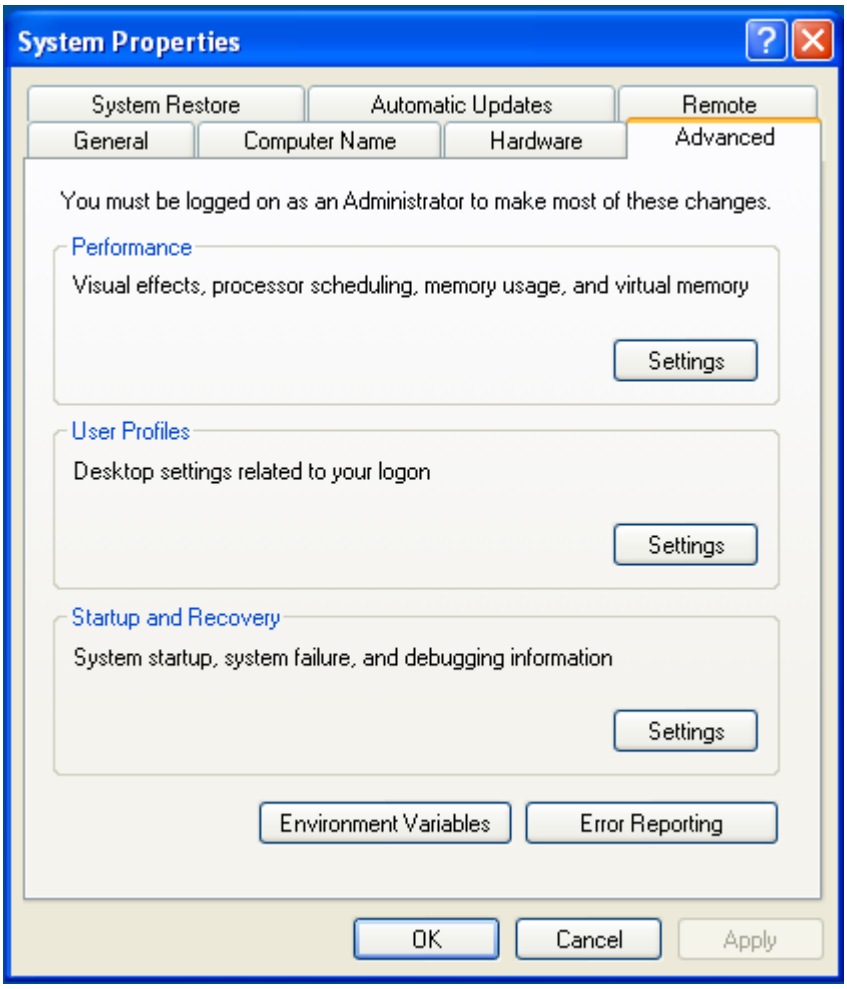


Figure 2: System Properties

Step 2: Click **Environment Variables**. The Environment Variables window will be displayed.

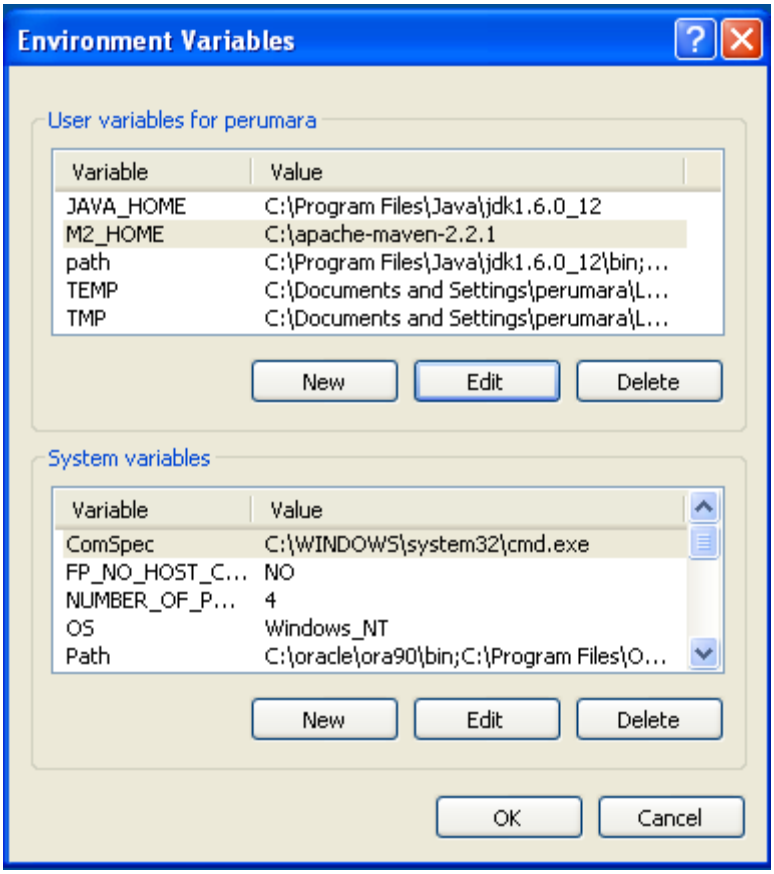


Figure 3: Environment Variables

Step 3: Create a new user variable M2_HOME by clicking on edit and set the path of Apache Maven installation path as shown in the figure.

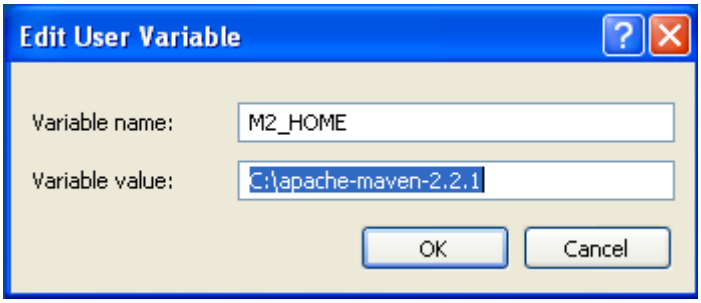


Figure 4: Edit User Variable

Step 4: Click **JAVA_HOME** System Variable if it already exists, or create a new one and set the path of JDK1.5 as shown in the figure.

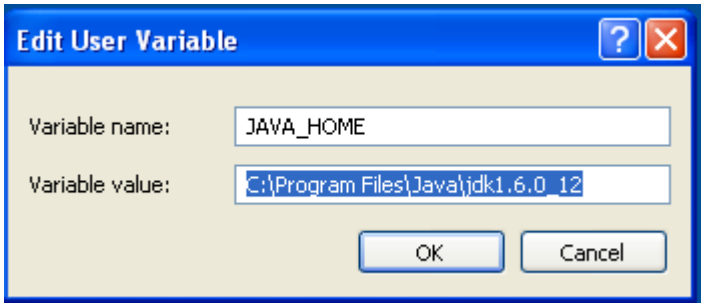


Figure 5: Edit User Variable

Step 5: Click **PATH** System Variable and set it as **%*JAVA_HOME*%*bin*;%M2_HOME%\bin;**

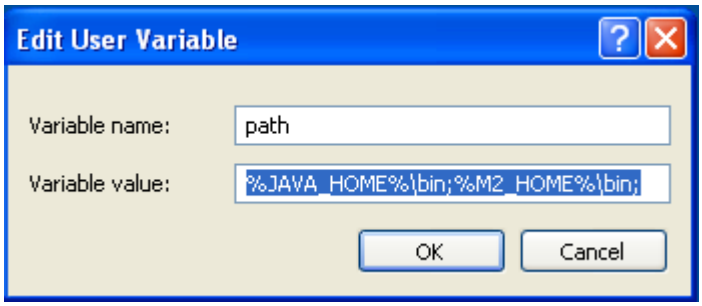


Figure 6: Edit User Variable

1.2 Configuring Maven Settings:

Step 1: Edit settings.xml to configure the proxy settings for downloading artifacts from remote repository.

```
<settings>
....
        <proxies>
                <proxy>
                        <active>true</active>
                        <protocol>http</protocol>
                        <host>proxy.mycompany.com</host>
                        <port>8080</port>
                        <username>your-username</username>
                        <password>your-password</password>
                </proxy>
        </proxies>
...
</settings>
```

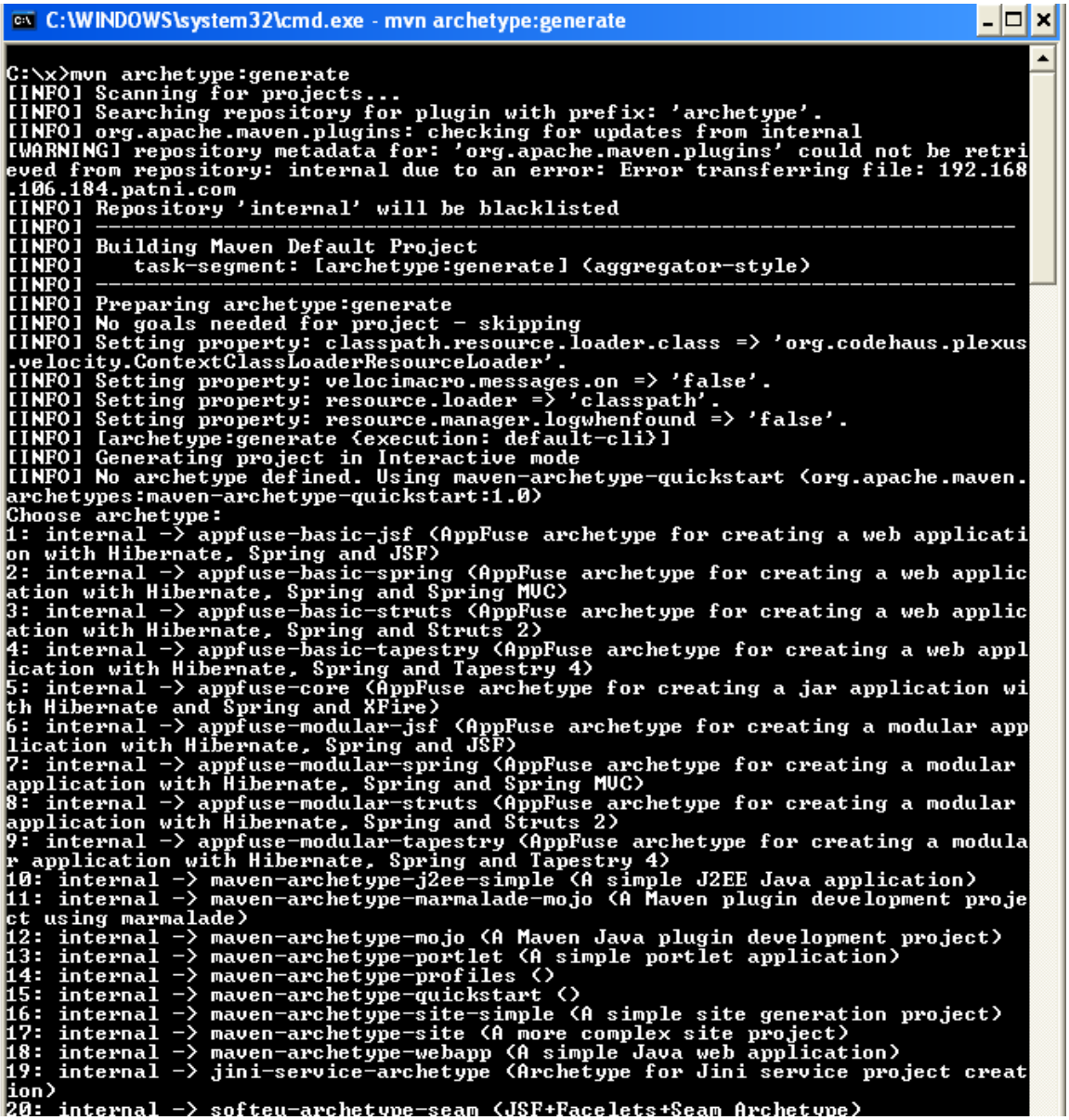
Figure 7: settings.xml

1.3 Creating first standalone Maven application using archetype:

Step 1: Execute the following command to create Maven Project

mvn archetype:generate

The execution of the above command should result into the display of archetypes as shown below:



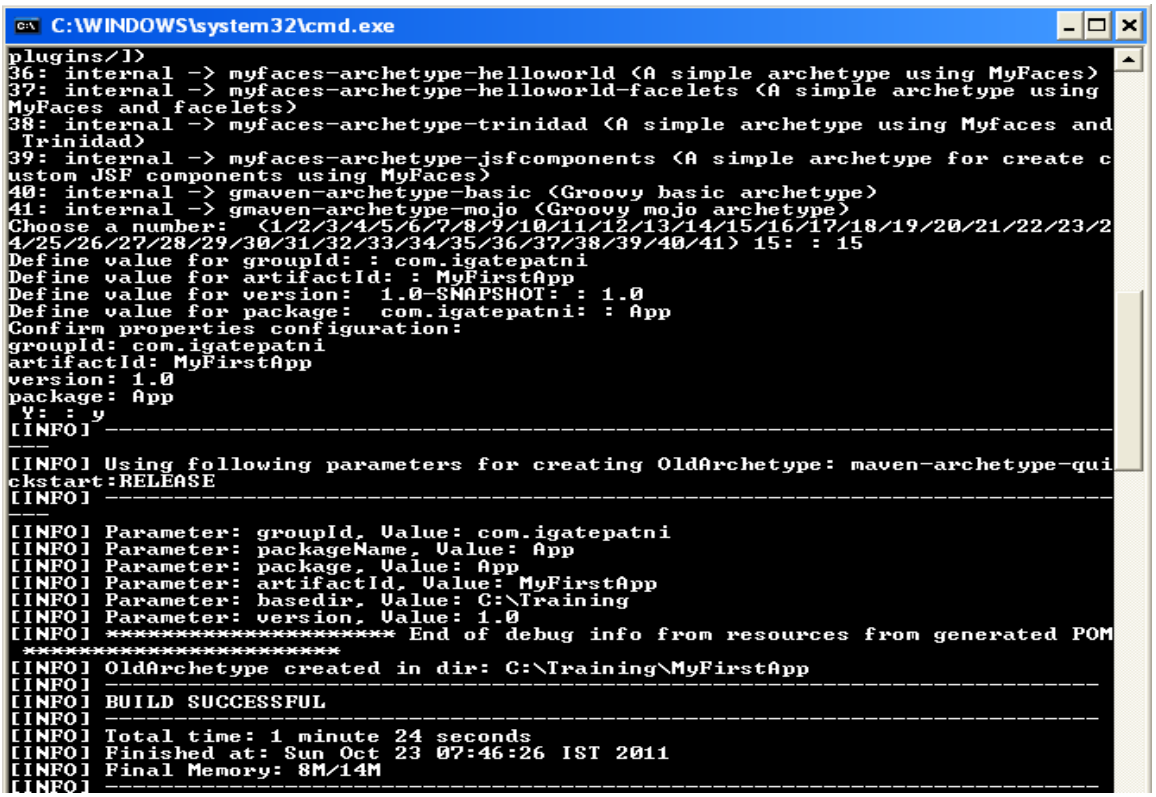
```
C:\WINDOWS\system32\cmd.exe - mvn archetype:generate

C:\x>mvn archetype:generate
[INFO] Scanning for projects...
[INFO] Searching repository for plugin with prefix: 'archetype'.
[INFO] org.apache.maven.plugins: checking for updates from internal
[WARNING] repository metadata for: 'org.apache.maven.plugins' could not be retri
eved from repository: internal due to an error: Error transferring file: 192.168
.106.184.patni.com
[INFO] Repository 'internal' will be blacklisted
[INFO] ----------------------------------------------------------------------------
[INFO] Building Maven Default Project
[INFO]    task-segment: [archetype:generate] (aggregator-style)
[INFO] ----------------------------------------------------------------------------
[INFO] Preparing archetype:generate
[INFO] No goals needed for project - skipping
[INFO] Setting property: classpath.resource.loader.class => 'org.codehaus.plexus
.velocity.ContextClassLoaderResourceLoader'.
[INFO] Setting property: velocimacro.messages.on => 'false'.
[INFO] Setting property: resource.loader => 'classpath'.
[INFO] Setting property: resource.manager.logwhenfound => 'false'.
[INFO] [archetype:generate (execution: default-cli)]
[INFO] Generating project in Interactive mode
[INFO] No archetype defined. Using maven-archetype-quickstart (org.apache.maven.
archetypes:maven-archetype-quickstart:1.0)
Choose archetype:
1: internal -> appfuse-basic-jsf (AppFuse archetype for creating a web applicati
on with Hibernate, Spring and JSF)
2: internal -> appfuse-basic-spring (AppFuse archetype for creating a web applic
ation with Hibernate, Spring and Spring MVC)
3: internal -> appfuse-basic-struts (AppFuse archetype for creating a web applic
ation with Hibernate, Spring and Struts 2)
4: internal -> appfuse-basic-tapestry (AppFuse archetype for creating a web appl
ication with Hibernate, Spring and Tapestry 4)
5: internal -> appfuse-core (AppFuse archetype for creating a jar application wi
th Hibernate and Spring and XFire)
6: internal -> appfuse-modular-jsf (AppFuse archetype for creating a modular app
lication with Hibernate, Spring and JSF)
7: internal -> appfuse-modular-spring (AppFuse archetype for creating a modular
application with Hibernate, Spring and Spring MVC)
8: internal -> appfuse-modular-struts (AppFuse archetype for creating a modular
application with Hibernate, Spring and Struts 2)
9: internal -> appfuse-modular-tapestry (AppFuse archetype for creating a modula
r application with Hibernate, Spring and Tapestry 4)
10: internal -> maven-archetype-j2ee-simple (A simple J2EE Java application)
11: internal -> maven-archetype-marmalade-mojo (A Maven plugin development proje
ct using marmalade)
12: internal -> maven-archetype-mojo (A Maven Java plugin development project)
13: internal -> maven-archetype-portlet (A simple portlet application)
14: internal -> maven-archetype-profiles ()
15: internal -> maven-archetype-quickstart ()
16: internal -> maven-archetype-site-simple (A simple site generation project)
17: internal -> maven-archetype-site (A more complex site project)
18: internal -> maven-archetype-webapp (A simple Java web application)
19: internal -> jini-service-archetype (Archetype for Jini service project creat
ion)
20: internal -> softeu-archetype-seam (JSF+Facelets+Seam Archetype)
```

Figure 8: Selecting an archetype

Step 2: Choose a number for selecting an archetype.

Example: selecting a number 15 allows creating a sample quick start project and it is the default archetype in maven. Similarly selecting a number 18 allows creating a sample web application project.

Step 3: After selecting an archetype, enter the project coordinates which is used to identify the current project by other projects in maven.



```
C:\WINDOWS\system32\cmd.exe
plugins/])
36: internal -> myfaces-archetype-helloworld (A simple archetype using MyFaces)
37: internal -> myfaces-archetype-helloworld-facelets (A simple archetype using
MyFaces and facelets)
38: internal -> myfaces-archetype-trinidad (A simple archetype using Myfaces and
 Trinidad)
39: internal -> myfaces-archetype-jsfcomponents (A simple archetype for create c
ustom JSF components using MyFaces)
40: internal -> gmaven-archetype-basic (Groovy basic archetype)
41: internal -> gmaven-archetype-mojo (Groovy mojo archetype)
Choose a number:  (1/2/3/4/5/6/7/8/9/10/11/12/13/14/15/16/17/18/19/20/21/22/23/2
4/25/26/27/28/29/30/31/32/33/34/35/36/37/38/39/40/41) 15: : 15
Define value for groupId: : com.igatepatni
Define value for artifactId: : MyFirstApp
Define value for version:  1.0-SNAPSHOT: : 1.0
Define value for package:  com.igatepatni: : App
Confirm properties configuration:
groupId: com.igatepatni
artifactId: MyFirstApp
version: 1.0
package: App
 Y: : y
[INFO] ----------------------------------------------------------------------
---
[INFO] Using following parameters for creating OldArchetype: maven-archetype-qui
ckstart:RELEASE
[INFO] ----------------------------------------------------------------------
---
[INFO] Parameter: groupId, Value: com.igatepatni
[INFO] Parameter: packageName, Value: App
[INFO] Parameter: package, Value: App
[INFO] Parameter: artifactId, Value: MyFirstApp
[INFO] Parameter: basedir, Value: C:\Training
[INFO] Parameter: version, Value: 1.0
[INFO] ********************* End of debug info from resources from generated POM
 **********************
[INFO] OldArchetype created in dir: C:\Training\MyFirstApp
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 1 minute 24 seconds
[INFO] Finished at: Sun Oct 23 07:46:26 IST 2011
[INFO] Final Memory: 8M/14M
[INFO] ------------------------------------------------------------------------
```

Figure 9: Project Information Specified

Step 4: After successful execution of the command, Maven will create the project directory MyFirstApp having pom.xml with the contents shown below.

Keep the application specific files in ${basedir}/src/main/java and test sources reside in ${basedir}/src/test/java, where ${basedir} represents the directory containing pom.xml.

```
<project>
        <modelVersion>4.0.0</modelVersion>
        <groupId>com.igatepatni</groupId>
        <artifactId>MyFirstApp</artifactId>
         <packaging>jar</packaging>
         <version>1.0</version>
         <name>MyFirstApp</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>
```

Figure 10: Sample pom.xml

Step 5: Execute the "mvn compile" command to compile your application sources as shown below:

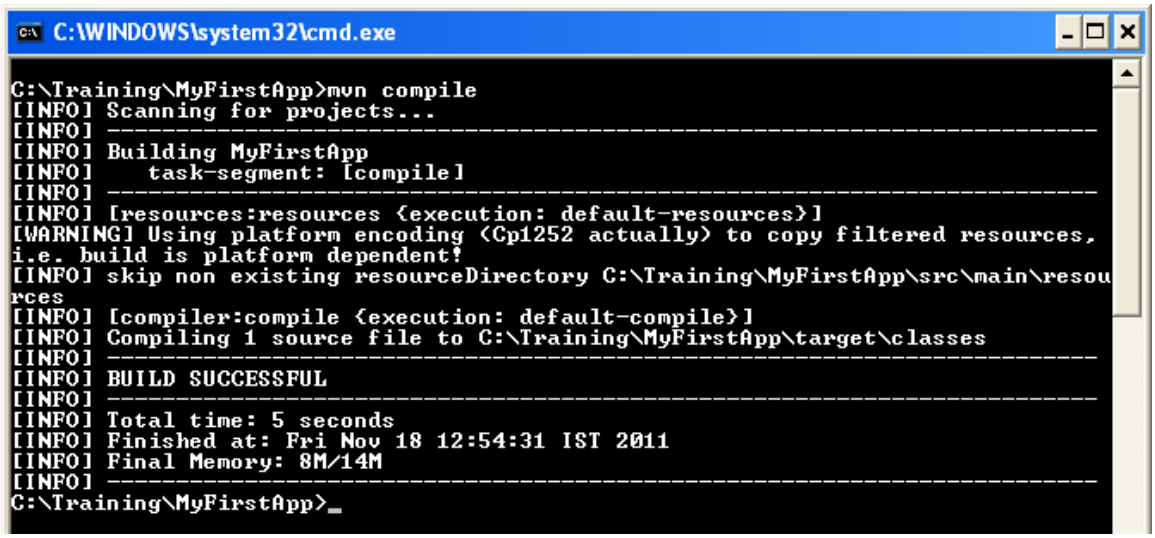


Figure 11:Compiling Sample Project

Step 6: Unit testing of the project can be performed by executing the following command:

mvn test

After successful execution of the command, the test result will be displayed which comprises the details such as test run,failures, errors...

Step 7: Artifact can be packaged and installed into local repository using commands such as mvn package and mvn install.

- mvn package packages the artifact based on the packaging type specified in the pom.xml.
- mvn install installs the packaged artifact into the local repository.

Step 8: Basic Standard site for project can also be created using command mvn site.

<<TODO>>

Assignment-1: Create a Banking System project in maven which maintains two kinds of accounts for customers, one called savings account and the other as current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book but no interest. Current account holders should have a minimum balance else they should pay service charges. Build, test and deploy the project into local repository.

Lab 2. Creating Multi Module Applications with Maven

Goals	• Learn and Understand the process of
	  • Creating Multi Module Project
	  • Generating Website for Multi Module Project
	  • Setting Up Own Repository
	  • Deploying an application into local repository

Time	60 minutes

2.1: Creating Multi Module Project

Step1: Create a directory named as “MainModule”. Change the working directory into MainModule.
Step2: Create a project named “submodule1” and “submodule2” by following the steps mentioned in Lab 1.3.
Step3: Create a parent POM file in MainModule as shown below which should consists of

- Packaging element in POM and
- Module elements describing about sub-modules.

```
<project>
   ...
   <packaging>pom</packaging>
   <modules>
      <module>submodule1</module>
      <module>submodule2</module>
   </modules>
</project>
```

Figure 12: Sample Parent POM

Step4: Customize the pom.xml in submodules to inherit the resources by including parent element as shown below:

```
……
<parent>
<groupId>com.igatepatni.app</groupId>
<artifactId>MainModule</artifactId>
<version>1.0-SNAPSHOT</version>
</parent>
……
```

Figure 13:Sample SubModule POM

Step5: Follow the steps as mentioned in Lab 1.3 to build the multi module project.

2.2: Generating Website for Multi Module Project

While executing mvn site command, multiple sites will be created. To create a single site with links to multiple modules, use the following command

mvn site:stage -DstagingDirectory=C:\MainModule

2.3: Setting up own repository

Step1: Create your own repository by installing archiva. Download archiva as a standalone application from url: http://archiva.apache.org/download.html and unzip the same.

Note: Archiva is a repository manager, which is used as a local host of your deployment repository.

Step2:

- Before we start Archiva, we must change the port it will run on, to avoid conflicting with other applications on the default port 8080.
- To change the port, edit the configuration below in the conf/jetty.xml file and change the default value of the jetty.port system property from 8080 to 8081.

```
<Call name="addConnector">
<Arg>
<New class="org.mortbay.jetty.nio.SelectChannelConnector">
<Set name="host"><SystemProperty name="jetty.host" /></Set>
<Set name="port"><SystemProperty name="jetty.port" default="8081"/></Set>
...
</New>
</Arg>
</Call>
```

Step3: To install archiva, go to. /bin in console and type archiva install

Step4: To start archiva, type archiva start in console under archiva bin directory.

Step5: Archiva can now be accessed at http://localhost:8081/archiva/. The **Create Admin User** page is the first page we will see when we access the URL.

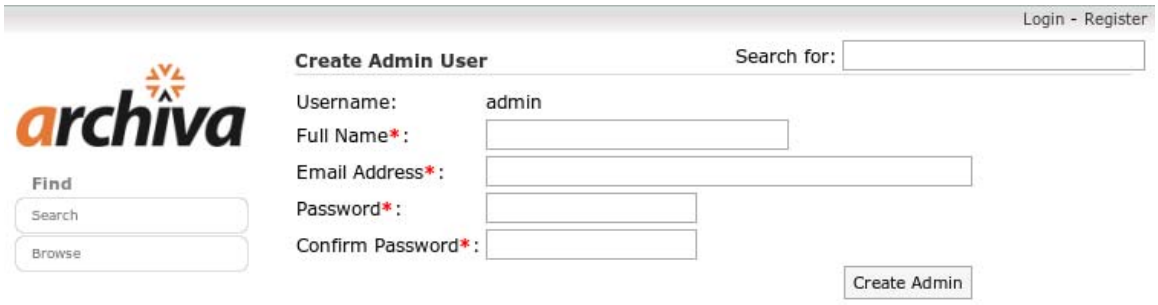


Figure 14: Home Page of archiva

The **admin** user is the ultimate god of Archiva—the **system administrator**. Let's fill out the **Create Admin User** form as follows:

Full Name: **Administrator**
Email Address: **admin@example.com**
Password: **admin1**
Confirm Password: **admin1**

After we create the **admin** user, log in to Archiva as admin.

Step6: Creating Network Proxy

- Click on Network Proxies in left hand side in admin login and click on add Network proxy link.
- Fill the network proxy details as shown below:

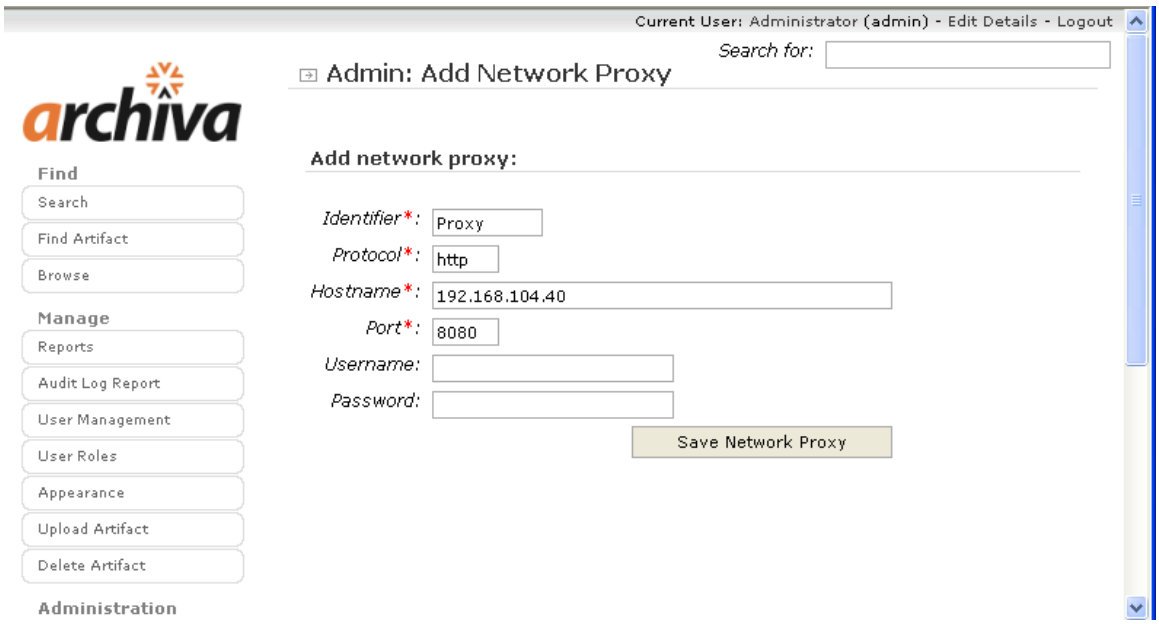


Figure 15: Editing Network Proxy

Note: Username and password is domain username and password.

Step7: Configuring Proxy Connectors

If proxy connection is used in local area network, you should specify a Network Proxy as Proxy name which you mentioned earlier while configuring Network proxy.

Follow the steps to configure the proxy connectors:

- Click on proxy connectors in left hand side and it will be displayed with configuration for internal repository proxies maven2-repository.dev.java.net and central as shown below

Figure 16: Repository Proxy Connectors

- Click on edit proxy connector and select the configured Network proxy as shown below instead of direct connection.

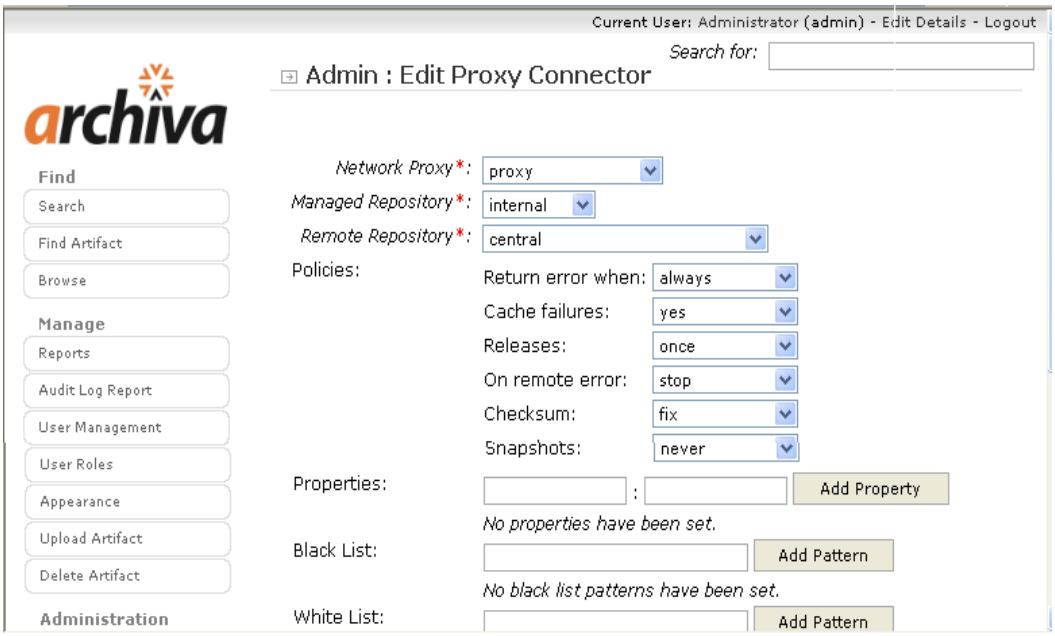


Figure 17: Editing Proxy Connector

2.4: Deploying an Application into own repository

- Create a user in Archiva repository with Role as “Repository Manager” to use for deployment.
- The deployment user needs the Role 'Repository Manager' for each repository that you want to deploy to

- Define the server for deployment inside your 'settings.xml', as shown below

```
<settings>
        ...
        <servers>
            <server>
                <id>archiva.internal</id>
                <username>{archiva-deployment-user}</username>
        <password>{archiva-deployment-pwd}</password>
            </server>
            <server>
                <id>archiva.snapshots</id>
                <username>{archiva-deployment-user}</username>
        <password>{archiva-deployment-pwd}</password>
            </server>
        </servers>
         ...
</settings>
```

Figure 18: Deployment user configuration in settings.xml

- Configure the distributionManagement part of your pom.xml to deploy an artifact using http protocol as mentioned below

```
<distributionManagement>
    <repository>
        <id>archiva.internal</id>
        <name>Internal Release Repository</name>
        <url>http://localhost:8081/archiva/repository/internal/</url>
    </repository>
    <snapshotRepository>
        <id>archiva.snapshots</id>
        <name>Internal Snapshot Repository</name>
        <url>http://localhost:8081/archiva/repository/snapshots/</url>
    </snapshotRepository>
</distributionManagement>
```

Figure 19: Configuring Distribution Management in pom.xml

- Use *mvn deploy* command to deploy the application.

<<TODO>>

Assignment-1: Use Assignment 1 in Lab 1 as one module and create another module for admin to perform report generation and managing with accounts.

- Create and build multi module project as described above.
- Generate a standard website for multi module project
- Customize the website in order to include the description of the project and developers details.
- Deploy the project into your own repository.

Lab 3. Integrating Maven with Eclipse

Goals	• Learn and Understand the process of
	  • Installing m2eclipse plugin
	  • Configure environment to run Maven Project
	  • Creating a simple Maven Project in eclipse

Time	60 minutes

3.1: Installing m2eclipse plugin

Step1: Select Help > Install New Software in eclipse. This should display the "Install" dialog.

Step2: Type the URL http://m2eclipse.sonatype.org/sites/m2e into the field named "Work with:" and press Enter to update list of available plugins and components

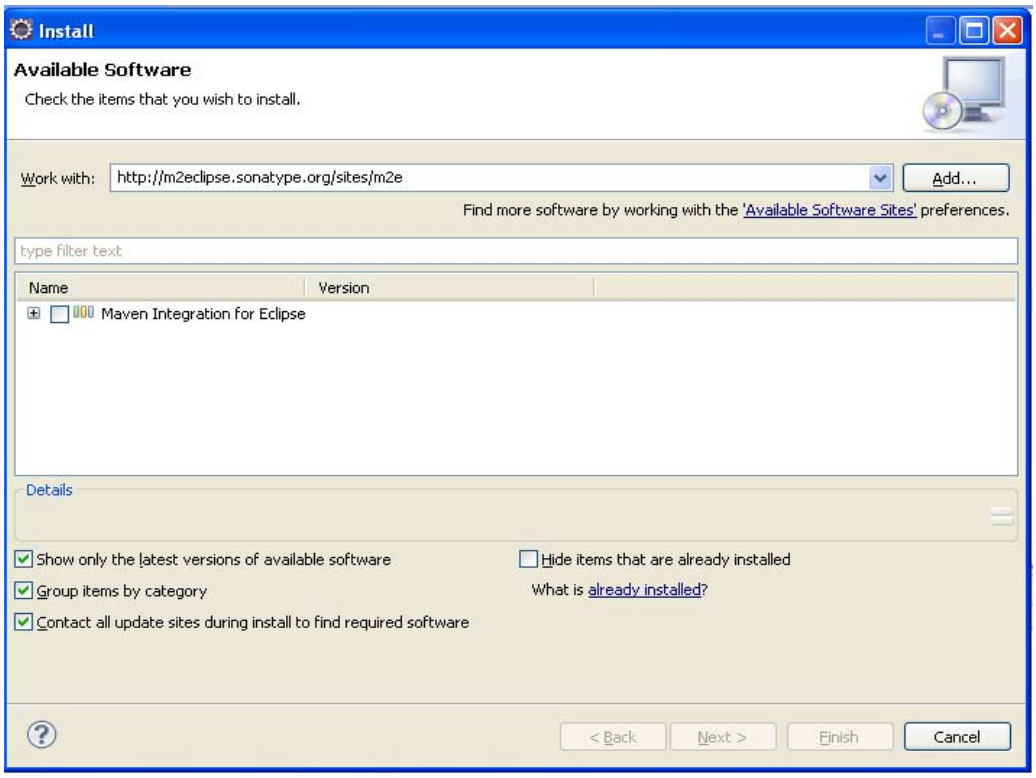


Figure 20: Installing m2eclipse plugin in Eclipse

Step3: Choose the component listed under m2eclipse: "Maven Integration for Eclipse (Required)".

Step4: Click Next. Eclipse will then check to see if there are any issues which would prevent a successful installation.

Step5: Click next and agree to the terms of the Eclipse Public License v1.0.

Step6: Click Finish to begin the installation process.

Step7: Once the installation process is finished, Eclipse will recommend restarting the IDE.

3.2: Configure environment to run Maven project

Step1: To execute Maven project, eclipse needs to be run using JDK instead of JRE.

- Go to the eclipse installation directory and open the file eclipse.ini in a text editor.
- Search for the line "-vmargs"
- Before the line "-vmargs", add two lines:
- On the first line, write "-vm".
- On the second line, write the path to your JDK installation

Step2: Start the eclipse now, to create a Maven Project and run successfully.

3.3: Creating a sample Maven project

Create a simple java project named 'myproject'..

Solution:

Step 1: Open **eclipse3.3**.

Step 2: Select **File → New → Project → Java project**.

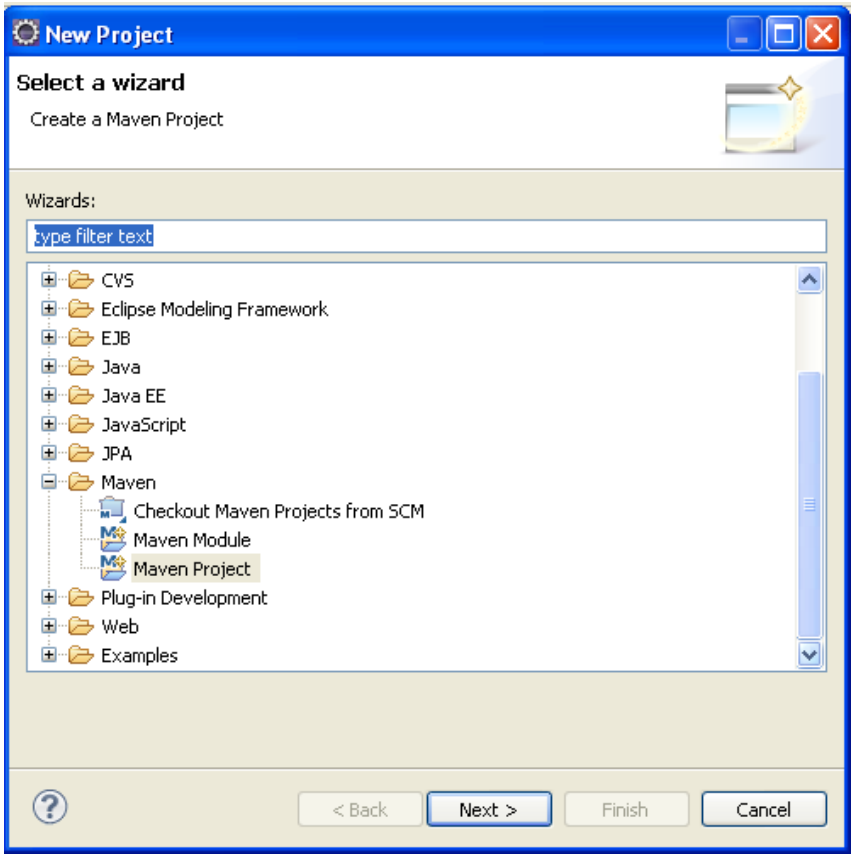


Figure 21: Select Wizard in Eclipse

Step 3: Choose Maven Project and use the default Workspace location or specify the location if necessary.

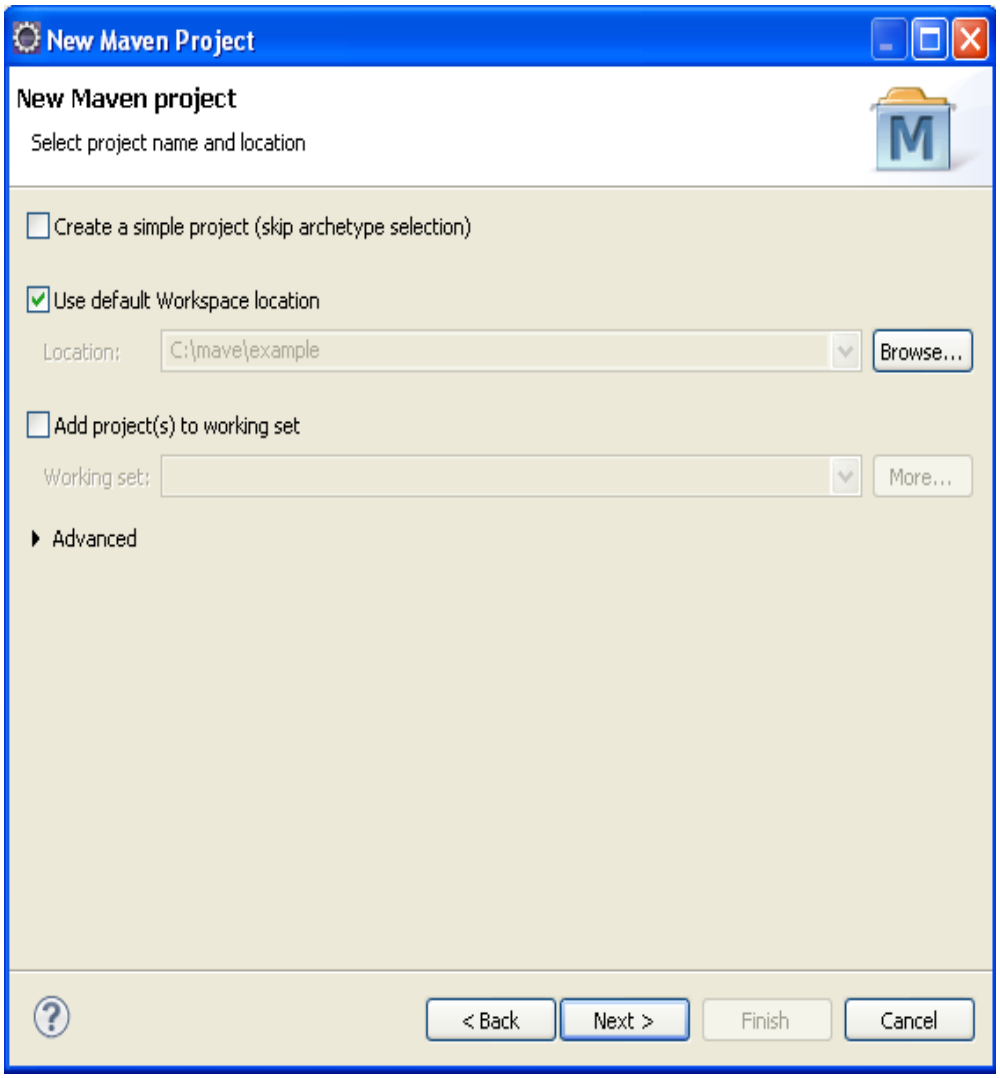


Figure 22: New Maven Project

Step 4: Select the maven-archetype-quickstart archetype from the list.

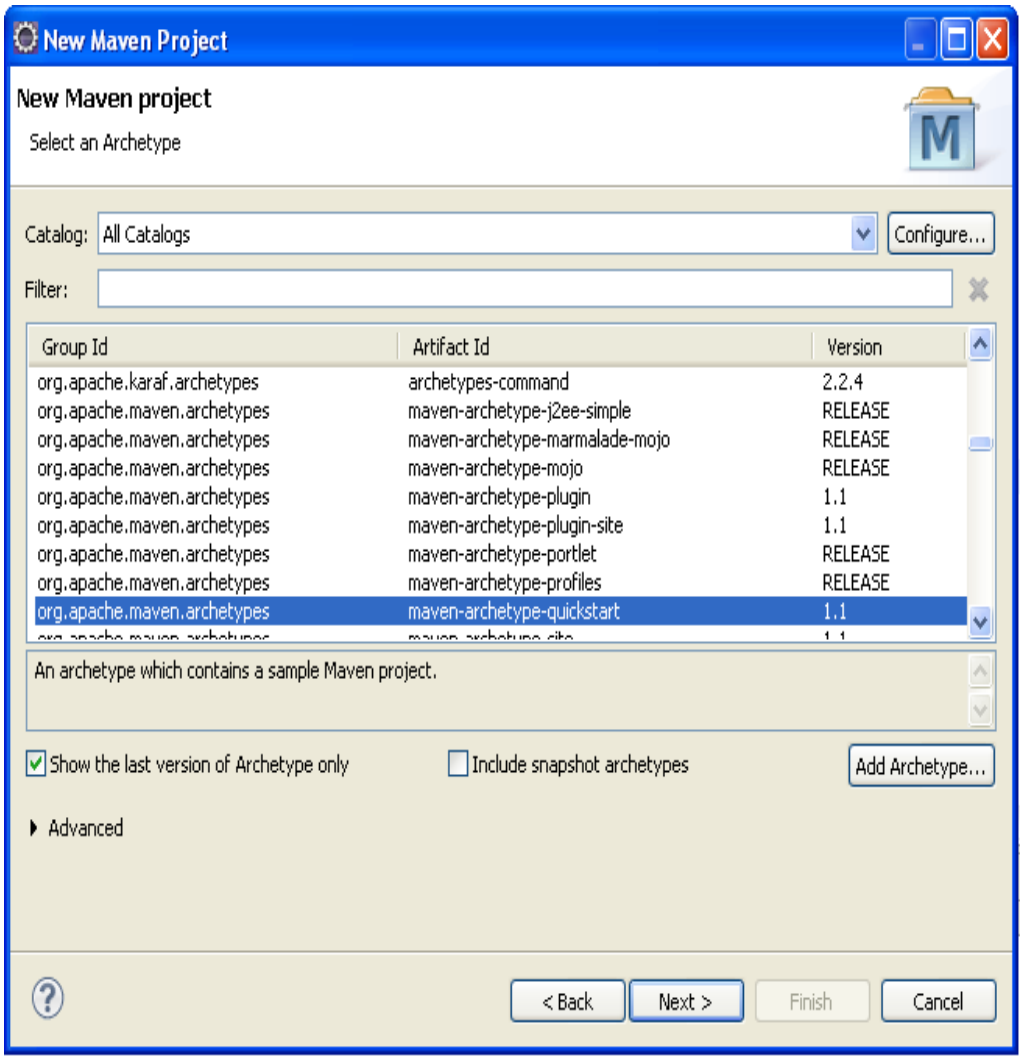


Figure 23: Selecting an archetype

Step 5: Enter the project coordinate details such as Group Id, Artifact Id and click 'Finish'

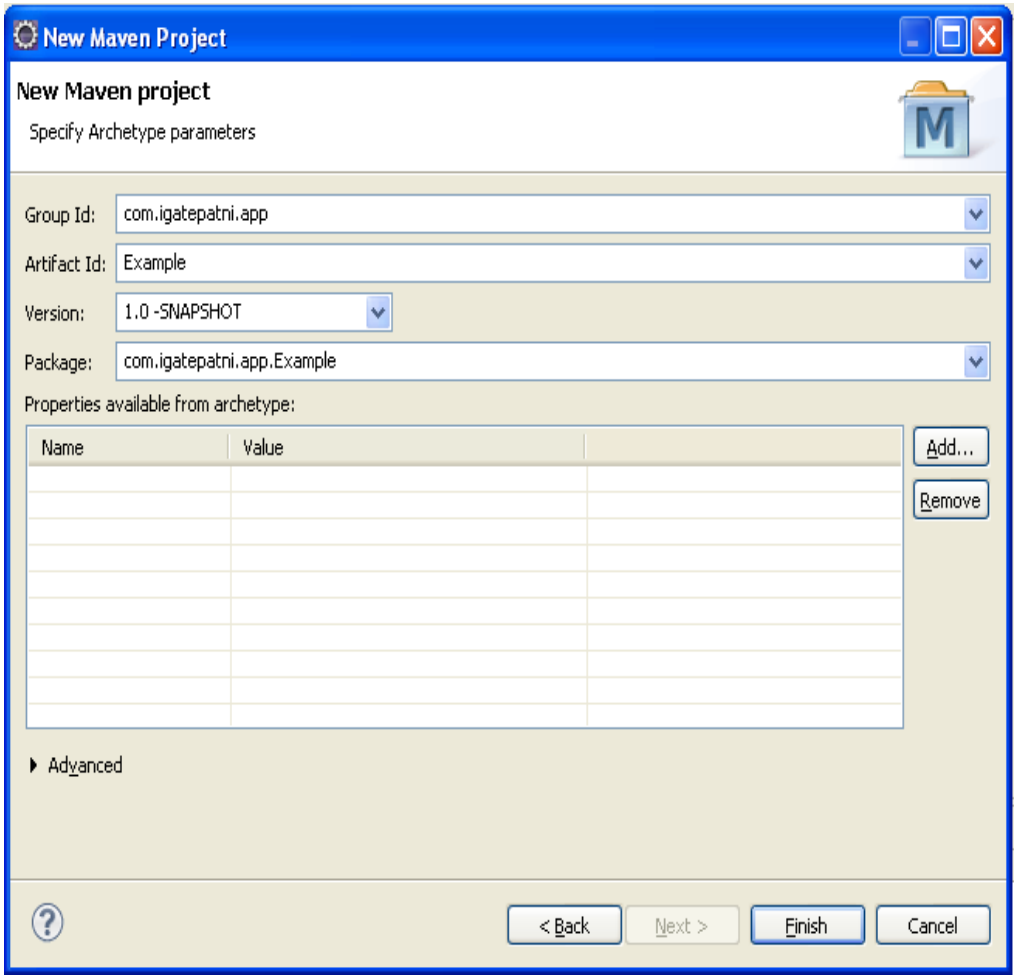


Figure 24: Specifying Archetype Parameters

Step 6: To build the project, right click on project named “examples” and choose Maven Build under Run As option.

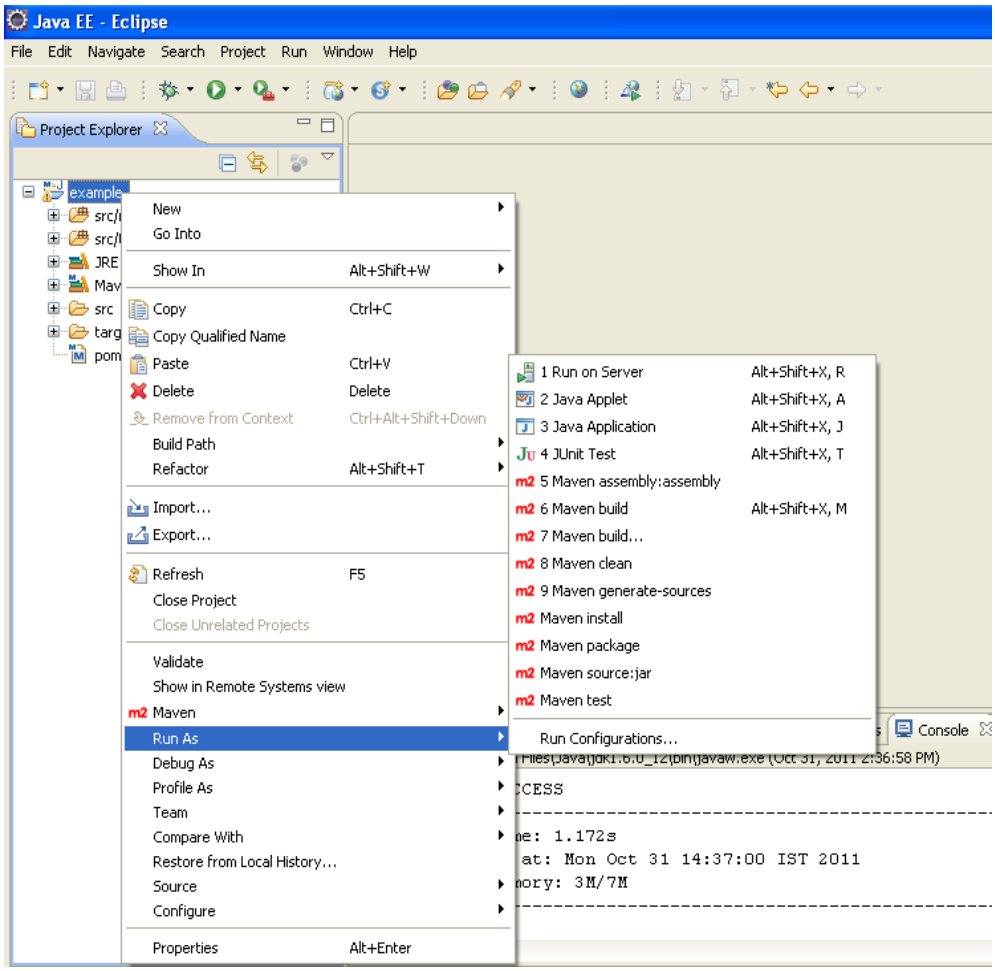


Figure 25: Building an Application

<<TODO>>

Assignment 1: Create a web application which displays product details such as Product Name, Product description, and its price. Users can place orders specifying the quantity of each product. Once the order is placed by customer, the invoice for the current products transaction showing the product name, quantity ordered, price and total amount should be displayed. Build and execute the common life cycle phases for the web application in eclipse.

Assignment 2: Use the Assignment 1 in Lab 1, import the Banking system project into eclipse and build the project in eclipse environment.

Appendices

Appendix A: Table of Figures