



|  |  |
| --- | --- |
| Install the maven from <https://maven.apache.org/download.cgi> :  Example: [apache-maven-3.5.2-bin.zip](http://www-eu.apache.org/dist/maven/maven-3/3.5.2/binaries/apache-maven-3.5.2-bin.zip)  **How to create a maven Project:**  **mvn archetype:generate**  The above commands download the required plugins to create a maven project. If the plugins are already there in the m2 directory then it will fetch the plugin from there. If it is not present then maven will download it from the repository.  Archetypes are the templates.  **Group Id:** Unique among the organization. Example if our server name is codesample.com. In group-Id we used to give in the reverse order. Ex: **com.codesaple. It is equivalent to package Name**  **Artifact Id:** Give the project name. Ex: MyFirstMavenDemo. **Equivalent to ClassName**  **Version**: Default is 1.0-SNAPSHOT  **Pom.xml**: Contains the information about the project  **To clean the maven project:**  Run As-> Maven Build…🡪In Goals write **clean**  **To compile the source files in the project:**  Run As-> Maven Build…🡪In Goals write **compile**  **To compile the test class:**  Run As-> Maven Build…🡪In Goals write **test-compile**  **To test the Test classes:**  Run As-> Maven Build…🡪In Goals write **test**  **To run single test (TestApp1), issue this command:**  mvn -Dtest=TestApp1 test  Surefire-report will give the test results  **To create the JAR/War file**   1. Run As->Maven Install **OR** 2. Run As->Maven Build…-> In Goals write **install**   **The above command will compile the java class, test class and create the JAR file.**  **In command Prompt:**   1. **To clean the project: mvn clean** 2. **mvn compile** 3. **mvn test-compile** 4. **mvn test** 5. **mvn install**  How to install your project into Maven local repository In Maven, you can use “mvn install” to package your project and deploy to local repository automatically, so that other developers can use it.  mvn install  When “**install**” phase is executed, all above phases “**validate**“, “**compile**“, “**test**“, “**package**“, “**integration-test**“, “**verify**” phase , including the current “**install**” phase will be executed orderly. Refer to this [Maven lifecycle](http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html) for detail.    **mvn install will install into local maven repository**  **Transitive dependency:**  Dependency of Dependency is called transitive dependency.  **Scope in maven:**  Anything with scope test will be limited to test source file.   1. test 2. compile (default)-Jar will be available at compile time 3. runtime—Dependency is not required for compilation. However, it is required at run time. 4. **Provided- It is more like compile. However, it expects it to provide by jdk.** |  |

A better command line tool:

<http://cmder.net/>

**How to enable proxy setting in Maven**

There is a high chance that the machine is set up with a firewall and an HTTP proxy server to stop user connects to internet directly. If you are behind a proxy, Maven will fail to download any dependencies.

To make it work, we should declare the proxy server setting in Maven configuration file settings.xml.

<!-- proxies

| This is a list of proxies which can be used on this machine to connect to the network.

| Unless otherwise specified (by system property or command-line switch), the first proxy

| specification in this list marked as active will be used.

|-->

<proxies>

<proxy>

<id>optional</id>

<active>true</active>

<protocol>http</protocol>

<username>mkyong</username>

<password>password</password>

<host>proxy.mkyong.com</host>

<port>8888</port>

<nonProxyHosts>local.net|some.host.com</nonProxyHosts>

</proxy>

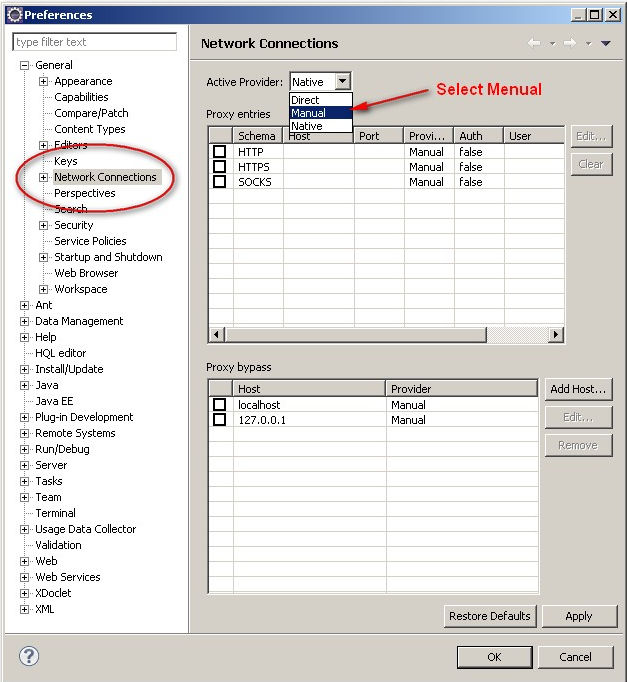
</proxies>

# How to configure Proxy Settings in Eclipse

1) In Eclipse IDE, select “Window –> Preferences”

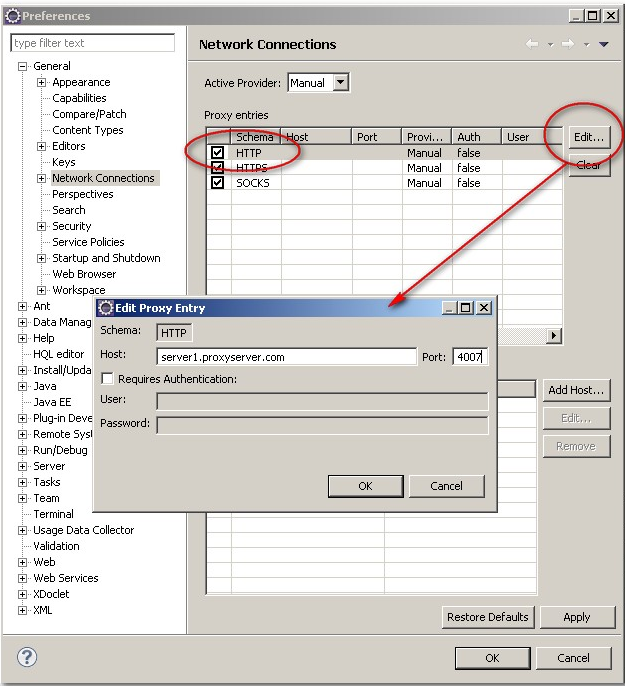
2) Preferences box prompt out, choose “Network Connections”.

3) Select “Manual” from Action Provider drop down list



4) Select Http in the List and click “Edit” button

5) Fill in the proxy server host and port number, (fill in the username and password if any)



# Where is Maven local repository?

The maven local repository is a local folder that is used to store all the project’s dependencies (plugin jars and other files which are downloaded by Maven). In simple, when we build a Maven project, all dependency files will be stored in our Maven local repository.

By default, Maven local repository is default to .m2 folder :

1. Unix/Mac OS X – ~/.m2
2. Windows – C:\Documents and Settings\{your-username}\.m2

## Update Maven Local Repository

Normally, I will change the default local repository folder from default .m2 to another more meaningful name, for example, maven-repo.

Find **{M2\_HOME}\conf\setting.xml**, update localRepository to something else.

<settings>

<!-- localRepository

| The path to the local repository maven will use to store artifacts.

|

| Default: ~/.m2/repository

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

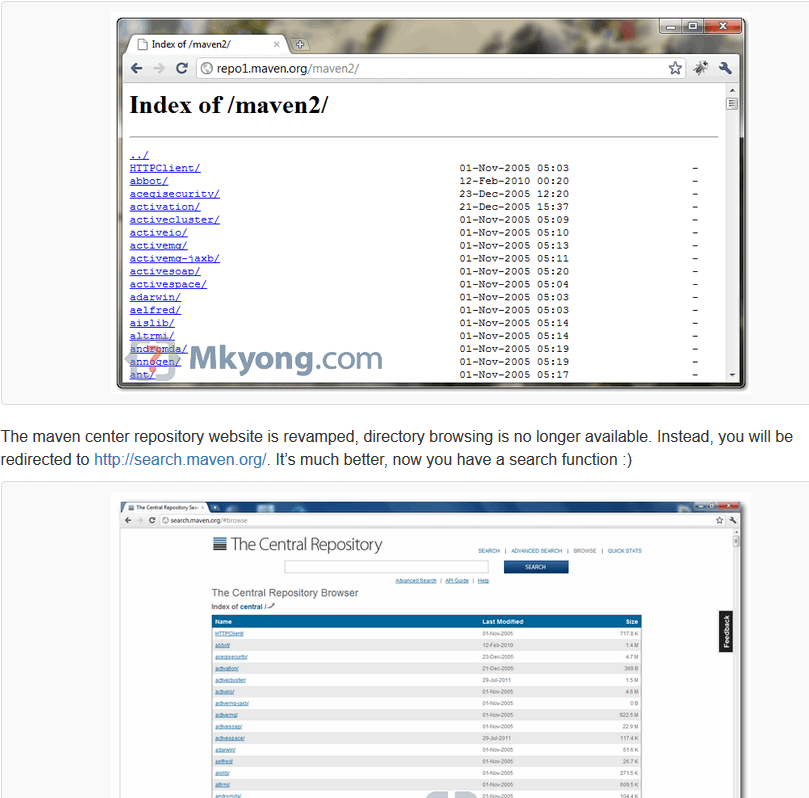
-->

<localRepository>D:/maven\_repo</localRepository>

# Where is maven central repository?

When we build a Maven’s project, Maven will check pom.xml file, to identify which dependency to download. First, Maven will get the dependency from local repository if not found, then get it from the default Maven central repository – [http://repo1.maven.org/maven2/](http://repo1.maven.org/maven/)

This is how the Maven central repository website looks like:



# How to download from Maven remote repository?

# Not all libraries are store in Maven central repository, often times, you need to add some remote repositories to download the libraries from another location instead of the default central repository.

To tell Maven to get the dependency from Java.net, you need to declared a remote repository in your pom.xml file like this :

<repositories>

<repository>

<id>java.net</id>

<url>https://maven.java.net/content/repositories/public/</url>

</repository>

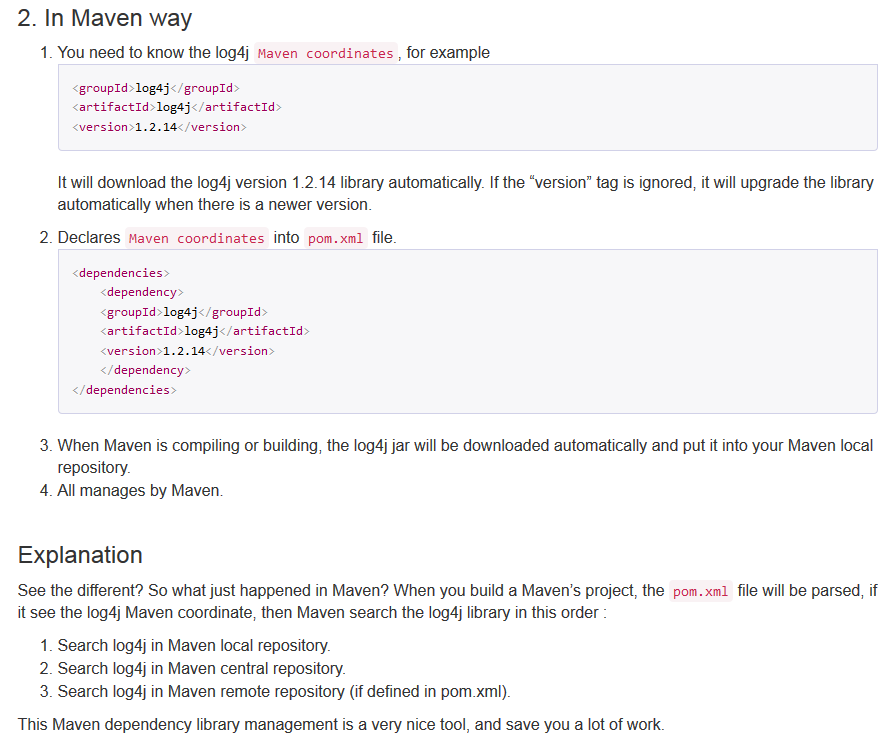
</repositories>

Now, Maven’s dependency library look-up sequences is changed to :

1. Search in Maven local repository, if not found, go step 2, else exit.
2. Search in Maven central repository, if not found, go step 3, else exit.
3. Search in java.net Maven remote repository, if not found, prompt error message, else exit.

# Maven dependency mechanism, how it works

Maven’s dependency mechanism help to download all the necessary dependency libraries automatically, and maintain the version upgrade as well.

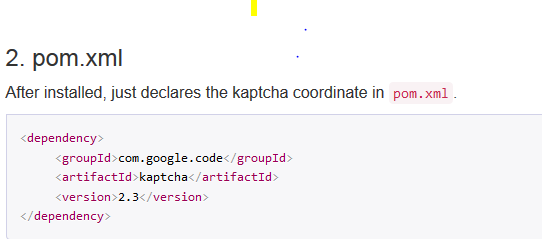


# How to include custom library into maven local repository?

There are 2 cases that you need to issue Maven’s command to include a jar into the Maven local repository manually.

1. The jar you want to use doesn’t exist in the Maven center repository.
2. You created a custom jar, and need to use for another Maven project.
3. For example, [kaptcha](http://code.google.com/p/kaptcha/), a popular third party Java library, which is used to generate “captcha” image to stop spam, but it’s not available in the Maven center repository.
4. In this tutorial, we will show you how to install the “kaptcha” jar into your Maven’s local repository.





**Deploying the WAR file in the server:**

<http://www.mkyong.com/maven/how-to-deploy-site-with-mvn-site-deploy-webdav-example/>

# How to deploy Maven based war file to Tomcat

# <http://www.mkyong.com/maven/how-to-deploy-maven-based-war-file-to-tomcat/>

# 

# Archtype: It decides which type of application we want to create

# Group Id: All the artifacts will come under this group

# Artifact Id: It is the output name of our project. Jar or WAR

# Version: While releasing the code the version can be tracked

# Package: What package the class should belong to.

# Compiler in Maven is a plugin. Configure it to tell how it should behave.

# By default org.apache.maven.plugins is the group Id. maven-compiler-plugin is the artifact id.

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.7.0</version>

<configuration>

<source>1.4</source>

</configuration>

</plugin>

</plugins>

</build>

**Default Java version is 1.4 in Maven 2 and 1.5 in Maven 3**

**The following line is part of Jetty which will auto compile and deploy the code change**

**<scanIntervalSeconds>5</ scanIntervalSeconds>**

**Maven Eclipse Plugin:**

maven eclipse:eclipse