**Maven Getting Started Guide**

### What is Maven?

At first glance Maven can appear to be many things, but in a nutshell Maven is an attempt *to apply patterns to a project's build infrastructure in order to promote comprehension and productivity by providing a clear path in the use of best practices*. Maven is essentially a project management and comprehension tool and as such provides a way to help with managing:

* Builds
* Documentation
* Reporting
* Dependencies
* SCMs
* Releases
* Distribution

If you want more background information on Maven you can check out [The Philosophy of Maven](http://maven.apache.org/background/philosophy-of-maven.html) and [The History of Maven](http://maven.apache.org/background/history-of-maven.html). Now let's move on to how you, the user, can benefit from using Maven.

### How can Maven benefit my development process?

Maven can provide benefits for your build process by employing standard conventions and practices to accelerate your development cycle while at the same time helping you achieve a higher rate of success.

Now that we have covered a little bit of the history and purpose of Maven let's get into some real examples to get you up and running with Maven!

### How do I setup Maven?

The defaults for Maven are often sufficient, but if you need to change the cache location or are behind a HTTP proxy, you will need to create configuration. See the [Guide to Configuring Maven](http://maven.apache.org/guides/mini/guide-configuring-maven.html) for more information.

**Configuring Maven**

Maven configuration occurs at 3 levels:

* *Project* - most static configuration occurs in pom.xml
* *Installation* - this is configuration added once for a Maven installation
* *User* - this is configuration specific to a particular user

The separation is quite clear - the project defines information that applies to the project, no matter who is building it, while the others both define settings for the current environment.

**Note:** the installation and user configuration cannot be used to add shared project information - for example, setting <organization> or <distributionManagement> company-wide.

For this, you should have your projects inherit from a company-wide parent pom.xml.

You can specify your user configuration in ${user.home}/.m2/settings.xml. A [full reference](http://maven.apache.org/maven-settings/settings.html) to the configuration file is available. This section will show how to make some common configurations. Note that the file is not required - defaults will be used if it is not found.

### Configuring your Local Repository

The location of your local repository can be changed in your user configuration. The default value is ${user.home}/.m2/repository/.

<settings>

...

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

...

</settings>

**Configuring a proxy**

You can configure a proxy to use for some or all of your HTTP requests with Maven. The username and password are only required if your proxy requires basic authentication (note that later releases may support storing your passwords in a secured keystore - in the mean time, please ensure your settings.xml file (usually ${user.home}/.m2/settings.xml) is secured with permissions appropriate for your operating system).

The nonProxyHosts setting accepts wild cards, and each host not to proxy is separated by the | character. This matches the JDK configuration equivalent.

1. <settings>
2. .
3. .
4. <proxies>
5. <proxy>
6. <id>example-proxy</id>
7. <active>true</active>
8. <protocol>http</protocol>
9. <host>proxy.example.com</host>
10. <port>8080</port>
11. <username>proxyuser</username>
12. <password>somepassword</password>
13. <nonProxyHosts>www.google.com|\*.example.com</nonProxyHosts>
14. </proxy>
15. </proxies>
16. .
17. .
18. </settings>

### Configuring Parallel Artifact Resolution

By default, Maven 2.1.0+ will download up to 5 artifacts (from different groups) at once. To change the size of the thread pool, start Maven using -Dmaven.artifact.threads. For example, to only download single artifacts at a time:

mvn -Dmaven.artifact.threads=1 verify

You may wish to set this option permanently, in which case you can use the MAVEN\_OPTS environment variable. For example:

export MAVEN\_OPTS=-Dmaven.artifact.threads=3

### Security and Deployment Settings

Repositories to deploy to are defined in a project in the <distributionManagement> section. However, you cannot put your username, password, or other security settings in that project. For that reason, you should add a server definition to your own settings with an id that matches that of the deployment repository in the project.

In addition, some repositories may require authorization to download from, so the corresponding settings can be specified in a server element in the same way.

Which settings are required will depend on the type of repository you are deploying to. As of the first release, only SCP deployments and file deployments are supported by default, so only the following SCP configuration is needed:

<settings>

...

<servers>

<server>

<id>repo1</id>

<username>repouser</username>

<!-- other optional elements:

<password>my\_login\_password</password>

<privateKey>/path/to/identity</privateKey> (default is ~/.ssh/id\_dsa)

<passphrase>my\_key\_passphrase</passphrase>

-->

</server>

...

</servers>

...

</settings>

To encrypt passwords in these sections, refer to [Encryption Settings](http://maven.apache.org/guides/mini/guide-encryption.html).

### How do I make my first Maven project?

We are going to jump headlong into creating your first Maven project! To create our first Maven project we are going to use Maven's archetype mechanism. An archetype is defined as *an original pattern or model from which all other things of the same kind are made*. In Maven, an archetype is a template of a project which is combined with some user input to produce a working Maven project that has been tailored to the user's requirements. We are going to show you how the archetype mechanism works now, but if you would like to know more about archetypes please refer to our [Introduction to Archetypes](http://maven.apache.org/guides/introduction/introduction-to-archetypes.html).

On to creating your first project! In order to create the simplest of Maven projects, execute the following from the command line:

1. mvn -B archetype:generate \
2. -DarchetypeGroupId=org.apache.maven.archetypes \
3. -DgroupId=com.mycompany.app \
4. -DartifactId=my-app

Once you have executed this command, you will notice a few things have happened. First, you will notice that a directory named my-app has been created for the new project, and this directory contains a file named pom.xml that should look like this:

1. <project xmlns="http://maven.apache.org/POM/4.0.0"
2. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
4. http://maven.apache.org/xsd/maven-4.0.0.xsd">
5. <modelVersion>4.0.0</modelVersion>
6. <groupId>com.mycompany.app</groupId>
7. <artifactId>my-app</artifactId>
8. <packaging>jar</packaging>
9. <version>1.0-SNAPSHOT</version>
10. <name>Maven Quick Start Archetype</name>
11. <url>http://maven.apache.org</url>
12. <dependencies>
13. <dependency>
14. <groupId>junit</groupId>
15. <artifactId>junit</artifactId>
16. <version>4.11</version>
17. <scope>test</scope>
18. </dependency>
19. </dependencies>
20. </project>

### What is a POM?

A Project Object Model or POM is the fundamental unit of work in Maven. It is an XML file that contains information about the project and configuration details used by Maven to build the project. It contains default values for most projects. Examples for this is the build directory, which is target; the source directory, which is src/main/java; the test source directory, which is src/test/java; and so on. When executing a task or goal, Maven looks for the POM in the current directory. It reads the POM, gets the needed configuration information, then executes the goal.

Some of the configuration that can be specified in the POM are the project dependencies, the plugins or goals that can be executed, the build profiles, and so on. Other information such as the project version, description, developers, mailing lists and such can also be specified.

This is a very simple POM but still displays the key elements every POM contains, so let's walk through each of them to familiarize you with the POM essentials:

* **project** This is the top-level element in all Maven pom.xml files.
* **modelVersion** This element indicates what version of the object model this POM is using. The version of the model itself changes very infrequently but it is mandatory in order to ensure stability of use if and when the Maven developers deem it necessary to change the model.
* **groupId** This element indicates the unique identifier of the organization or group that created the project. The groupId is one of the key identifiers of a project and is typically based on the fully qualified domain name of your organization. For example org.apache.maven.plugins is the designated groupId for all Maven plugins.
* **artifactId** This element indicates the unique base name of the primary artifact being generated by this project. The primary artifact for a project is typically a JAR file. Secondary artifacts like source bundles also use the artifactId as part of their final name. A typical artifact produced by Maven would have the form <artifactId>-<version>.<extension> (for example, myapp-1.0.jar).
* **packaging** This element indicates the package type to be used by this artifact (e.g. JAR, WAR, EAR, etc.). This not only means if the artifact produced is JAR, WAR, or EAR but can also indicate a specific lifecycle to use as part of the build process. (The lifecycle is a topic we will deal with further on in the guide. For now, just keep in mind that the indicated packaging of a project can play a part in customizing the build lifecycle.) The default value for the packaging element is JAR so you do not have to specify this for most projects.
* **version** This element indicates the version of the artifact generated by the project. Maven goes a long way to help you with version management and you will often see the SNAPSHOT designator in a version, which indicates that a project is in a state of development. We will discuss the use of [snapshots](http://maven.apache.org/guides/getting-started/index.html#What_is_a_SNAPSHOT_version) and how they work further on in this guide.
* **name** This element indicates the display name used for the project. This is often used in Maven's generated documentation.
* **url** This element indicates where the project's site can be found. This is often used in Maven's generated documentation.
* **description** This element provides a basic description of your project. This is often used in Maven's generated documentation.