**Documentation**

**Getting Started with Maven**

* [Getting Started in 5 Minutes](http://maven.apache.org/guides/getting-started/maven-in-five-minutes.html)
* [Getting Started in 30 Minutes](http://maven.apache.org/guides/getting-started/index.html)

**Introductions**

* [The Build Lifecycle](http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html)
* [The POM](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html)
* [Profiles](http://maven.apache.org/guides/introduction/introduction-to-profiles.html)
* [Repositories](http://maven.apache.org/guides/introduction/introduction-to-repositories.html)
* [Standard Directory Layout](http://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html)
* [The Dependency Mechanism](http://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html)
* [Optional Dependencies and Dependency Exclusions](http://maven.apache.org/guides/introduction/introduction-to-optional-and-excludes-dependencies.html)

**Plugins**

* [Plugin Development](http://maven.apache.org/guides/introduction/introduction-to-plugins.html)
* [Configuring Plug-ins](http://maven.apache.org/guides/mini/guide-configuring-plugins.html)
* [Plugin Prefix Resolution](http://maven.apache.org/guides/introduction/introduction-to-plugin-prefix-mapping.html)
* [Developing Java Plugins](http://maven.apache.org/guides/plugin/guide-java-plugin-development.html)

**Site**

* [Creating a Site](http://maven.apache.org/guides/mini/guide-site.html)
* [The APT Format](http://maven.apache.org/doxia/references/apt-format.html)
* [Snippet Macro](http://maven.apache.org/guides/mini/guide-snippet-macro.html)

**Archetypes**

* [What is an Archetype](http://maven.apache.org/guides/introduction/introduction-to-archetypes.html)
* [Creating Archetypes](http://maven.apache.org/guides/mini/guide-creating-archetypes.html)

**Upgrading**

* [Relocation of Artifacts](http://maven.apache.org/guides/mini/guide-relocation.html)

**Repositories**

* [Installing 3rd party JARs to Local Repository](http://maven.apache.org/guides/mini/guide-3rd-party-jars-local.html)
* [Deploying 3rd party JARs to Remote Repository](http://maven.apache.org/guides/mini/guide-3rd-party-jars-remote.html)
* [Coping with Sun JARs](http://maven.apache.org/guides/mini/guide-coping-with-sun-jars.html)
* [Remote repository access through authenticated HTTPS](http://maven.apache.org/guides/mini/guide-repository-ssl.html)

**Guides**

* [Creating Assemblies](http://maven.apache.org/guides/mini/guide-assemblies.html)
* [Configuring Archive Plugins](http://maven.apache.org/guides/mini/guide-archive-configuration.html)
* [Configuring Maven](http://maven.apache.org/guides/mini/guide-configuring-maven.html)
* [Mirror Settings](http://maven.apache.org/guides/mini/guide-mirror-settings.html)
* [Deployment and Security Settings](http://maven.apache.org/guides/mini/guide-deployment-security-settings.html)
* [Generating Sources](http://maven.apache.org/guides/mini/guide-generating-sources.html)
* [Working with Manifests](http://maven.apache.org/guides/mini/guide-manifest.html)
* [Maven Classloading](http://maven.apache.org/guides/mini/guide-maven-classloading.html)
* [Using Multiple Modules in a Build](http://maven.apache.org/guides/mini/guide-multiple-modules.html)
* [Using Multiple Repositories](http://maven.apache.org/guides/mini/guide-multiple-repositories.html)
* [Using Proxies](http://maven.apache.org/guides/mini/guide-proxies.html)
* [Using the Release Plugin](http://maven.apache.org/guides/mini/guide-releasing.html)
* [Using Ant with Maven](http://maven.apache.org/guides/mini/guide-using-ant.html)
* [Using Modello](http://maven.apache.org/guides/mini/guide-using-modello.html)
* [Using Extensions](http://maven.apache.org/guides/mini/guide-using-extensions.html)
* [Building For Different Environments with Maven 2](http://maven.apache.org/guides/mini/guide-building-for-different-environments.html)
* [Using Toolchains](http://maven.apache.org/guides/mini/guide-using-toolchains.html)
* [Encrypting passwords in settings.xml](http://maven.apache.org/guides/mini/guide-encryption.html)
* [Guide to HTTP Connection Settings](http://maven.apache.org/guides/mini/guide-http-settings.html)
* [Guide to Selecting Alternative Wagon Providers](http://maven.apache.org/guides/mini/guide-wagon-providers.html)
* [Guide to Building JDK 1.4 Projects Using JDK 1.5](http://maven.apache.org/guides/mini/guide-building-jdk14-on-jdk15.html)
* [Guide to Configuring Default Mojo Executions](http://maven.apache.org/guides/mini/guide-default-execution-ids.html)

**Maven Tools and IDE Integration**

* [Maven Auto-Completion Using BASH](http://maven.apache.org/guides/mini/guide-bash-m2-completion.html)

**Development Guides**

* [Building Maven from Scratch](http://maven.apache.org/guides/development/guide-building-maven.html)
* [Developing Maven](http://maven.apache.org/guides/development/guide-maven-development.html)
* [The Plugin Documentation Standard](http://maven.apache.org/guides/development/guide-plugin-documentation.html)
* [Maven Documentation Style](http://maven.apache.org/guides/development/guide-documentation-style.html)

**The Maven Community**

* [The Maven Community](http://maven.apache.org/community.html)
* [Helping with Maven](http://maven.apache.org/guides/development/guide-helping.html)
* [Guide for New Committers](http://maven.apache.org/guides/mini/guide-new-committers.html)
* [Testing Development Versions of Plugins](http://maven.apache.org/guides/development/guide-testing-development-plugins.html)
* [3rd Party Resources](http://maven.apache.org/articles.html)

**Conventions**

* [Maven Conventions](http://maven.apache.org/maven-conventions.html)
* [Naming Conventions](http://maven.apache.org/guides/mini/guide-naming-conventions.html)
* [When You Can't Use the Conventions](http://maven.apache.org/guides/mini/guide-using-one-source-directory.html)

**The Central Repository**

* [Uploading Artifacts to the Central Repository](http://maven.apache.org/guides/mini/guide-central-repository-upload.html)
* [Improving the Repository](http://maven.apache.org/guides/mini/guide-maven-evangelism.html)

**References**

* [POM Overview](http://maven.apache.org/pom.html) ([Technical Project Descriptor](http://maven.apache.org/ref/current/maven-model/maven.html))
* [Settings Overview](http://maven.apache.org/settings.html) ([Technical Settings Descriptor](http://maven.apache.org/ref/current/maven-settings/settings.html))
* [Core Plug-ins List](http://maven.apache.org/plugins/index.html)
* [Mojo API](http://maven.apache.org/developers/mojo-api-specification.html)
* [Glossary](http://maven.apache.org/glossary.html)
* [Maven Quick Reference Card - PDF](http://maven.apache.org/guides/MavenQuickReferenceCard.pdf)

**Javadoc API**

Here is some useful Javadoc API links to the current version of Maven:

* [Maven Artifact](http://maven.apache.org/ref/current/maven-artifact/apidocs/)
* [Maven Reporting](http://maven.apache.org/shared/maven-reporting-api/apidocs/)
* [Maven Plugin API](http://maven.apache.org/ref/current/maven-plugin-api/apidocs/)
* [Maven Model](http://maven.apache.org/ref/current/maven-model/apidocs/)
* [Maven Core](http://maven.apache.org/ref/current/maven-core/apidocs/)
* [Maven Settings](http://maven.apache.org/ref/current/maven-settings/apidocs/)

You could also browse the [full technical documentation references](http://maven.apache.org/ref/current/) of the current version of Maven.

**Maven in 5 Minutes**

**Prerequisites**

You must have an understanding of how to install software on your computer. If you do not know how to do this, please ask someone at your office, school, etc or pay someone to explain this to you. The Maven mailing lists are not the best place to ask for this advice.

**Installation**

*Maven is a Java tool, so you must have*[*Java*](http://www.oracle.com/technetwork/java/javase/downloads/index.html)*installed in order to proceed.*

First, [download Maven](http://maven.apache.org/download.html) and follow the [installation instructions](http://maven.apache.org/download.html#Installation). After that, type the following in a terminal or in a command prompt:

1. mvn –-version 或mvn –v

It should print out your installed version of Maven, for example:

1. Apache Maven 3.0.5 (r01de14724cdef164cd33c7c8c2fe155faf9602da; 2013-02-19 14:51:28+0100)
2. Maven home: D:\apache-maven-3.0.5\bin\..
3. Java version: 1.6.0\_25, vendor: Sun Microsystems Inc.
4. Java home: C:\Program Files\Java\jdk1.6.0\_25\jre
5. Default locale: nl\_NL, platform encoding: Cp1252
6. OS name: "windows 7", version: "6.1", arch: "amd64", family: "windows"

Depending upon your network setup, you may require extra configuration. Check out the [Guide to Configuring Maven](http://maven.apache.org/guides/mini/guide-configuring-maven.html) if necessary.

**If you are using Windows, you should look at** [Windows Prerequisites](http://maven.apache.org/guides/getting-started/windows-prerequisites.html) **to ensure that you are prepared to use Maven on Windows.**

**Creating a Project**

You will need somewhere for your project to reside, create a directory somewhere and start a shell in that directory. On your command line, execute the following Maven goal:

1. mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

groupId 组织名

aritfactId 项目名

archetypeArtifactId 项目类型

interactiveMode 交互模式

如果直接执行mvn init，则会进入交互模式，然后在交互模式中指定组织名、项目名等

*If you have just installed Maven, it may take a while on the first run. This is because Maven is downloading the most recent artifacts (plugin jars and other files) into your local repository（本地仓库，可以自己配置） You may also need to execute the command a couple of times before it succeeds. This is because the remote server may time out before your downloads are complete. Don't worry, there are ways to fix that.*

You will notice that the *generate* goal created a directory with the same name given as the artifactId. Change into that directory.

1. cd my-app

Under this directory you will notice the following [standard project structure](http://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html).

1. my-app
2. |-- pom.xml
3. `-- src
4. |-- main
5. | `-- java
6. | `-- com
7. | `-- mycompany
8. | `-- app
9. | `-- App.java
10. `-- test
11. `-- java
12. `-- com
13. `-- mycompany
14. `-- app
15. `-- AppTest.java

The src/main/java directory contains the project source code, the src/test/java directory contains the test source, and the pom.xml file is the project's Project Object Model, or POM.

**The POM**

The pom.xml file is the core of a project's configuration in Maven. It is a single configuration file that contains the majority of information required to build a project in just the way you want. The POM is huge and can be daunting in its complexity, but it is not necessary to understand all of the intricacies just yet to use it effectively. This project's POM is:

1. <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2. xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
3. <modelVersion>4.0.0</modelVersion>
5. <groupId>com.mycompany.app</groupId>
6. <artifactId>my-app</artifactId>
7. <version>1.0-SNAPSHOT</version>
8. <packaging>jar</packaging>
10. <name>Maven Quick Start Archetype</name>
11. <url>http://maven.apache.org</url>
13. <dependencies>
14. <dependency>
15. <groupId>junit</groupId>
16. <artifactId>junit</artifactId>
17. <version>4.8.2</version>
18. <scope>test</scope>
19. </dependency>
20. </dependencies>
21. </project>

**What did I just do?**

You executed the Maven goal *archetype:generate*, and passed in various parameters to that goal. The prefix *archetype* is the [plugin](http://maven.apache.org/plugins/index.html) that contains the goal. If you are familiar with [Ant](http://ant.apache.org/), you may conceive of this as similar to a task. This goal created a simple project based upon an archetype. Suffice it to say for now that a *plugin* is a collection of *goals* with a general common purpose. For example the jboss-maven-plugin, whose purpose is "deal with various jboss items".

**Build the Project**

1. mvn package

The command line will print out various actions, and end with the following:

1. ...
2. [INFO] ------------------------------------------------------------------------
3. [INFO] BUILD SUCCESSFUL
4. [INFO] ------------------------------------------------------------------------
5. [INFO] Total time: 2 seconds
6. [INFO] Finished at: Thu Jul 07 21:34:52 CEST 2011
7. [INFO] Final Memory: 3M/6M
8. [INFO] ------------------------------------------------------------------------

Unlike the first command executed (*archetype:generate*) you may notice the second is simply a single word - *package*. Rather than a goal, this is a *phase*. A phase is a step in the [build lifecycle](http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html), which is an ordered sequence of phases. When a phase is given, Maven will execute every phase in the sequence up to and including the one defined. For example, if we execute the *compile* phase, the phases that actually get executed are:

1. validate
2. generate-sources
3. process-sources
4. generate-resources
5. process-resources
6. compile

You may test the newly compiled and packaged JAR with the following command:

1. java -cp target/my-app-1.0-SNAPSHOT.jar com.mycompany.app.App

Which will print the quintessential:

1. Hello World!

**Running Maven Tools**

**Maven Phases**

Although hardly a comprehensive list, these are the most common *default* lifecycle phases executed.

* **validate**: validate the project is correct and all necessary information is available
* **compile**: compile the source code of the project
* **test**: test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
* **package**: take the compiled code and package it in its distributable format, such as a JAR.
* **integration-test**: process and deploy the package if necessary into an environment where integration tests can be run
* **verify**: run any checks to verify the package is valid and meets quality criteria
* **install**: install the package into the local repository, for use as a dependency in other projects locally
* **deploy**: done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.

There are two other Maven lifecycles of note beyond the *default* list above. They are

* **clean**: cleans up artifacts created by prior builds
* **site**: generates site documentation for this project

Phases are actually mapped to underlying goals. The specific goals executed per phase is dependant upon the packaging type of the project. For example, *package* executes *jar:jar* if the project type is a JAR, and *war:war* if the project type is - you guessed it - a WAR.

An interesting thing to note is that phases and goals may be executed in sequence.

1. mvn clean dependency:copy-dependencies package

This command will clean the project, copy dependencies, and package the project (executing all phases up to *package*, of course).

**Generating the Site**

1. mvn site

This phase generates a site based upon information on the project's pom. You can look at the documentation generated under target/site.

**Conclusion**

We hope this quick overview has piqued your interest in the versatility of Maven. Note that this is a very truncated quick-start guide. Now you are ready for more comprehensive details concerning the actions you have just performed. Check out the [Maven Getting Started Guide](http://maven.apache.org/guides/getting-started/index.html).