# Maven Cheatsheet

## Useful Directories and Setup

The local repo cache (where Maven caches the artifacts it downloads) is defined in: `<home-dir>/.m2/repository`:

- `~/.m2/repository` (on Linux/Mac)
- `C:\Documents and Settings\.m2\repository` (on Windows)

The system should define an environment variable: `M2_HOME` which is set to the directory where Maven is unzipped to.

The system should also use this environment variable in its system path to point to the Maven commandline tools. This is specifed as:

- `$M2_HOME/bin` (in Linux/Mac)
- `%M2_HOME%\bin` (in Windows)

## Command Invocation Format

Maven either runs all build steps to a specified *phase*:

- `mvn compile`
- `mvn package` etc

This will run all steps up to and including the specified phase (see the Build Lifecycle for phases).

or runs a *specific goal of a specified plugin*:

- `mvn help:effective-pom`

You can also chain invocations together:

`mvn clean install` will clean the project, then install it to the local repo cache.

## Creating a Project

Create a project with the Archetype plugin. This can be done

- **interactively** by entering the details as prompted like this `mvn archectype:generate` and following the onscreen prompts
- **unattended** by using this command which specifies all details to generate a general Java project:

```
mvn archetype:generate -DarchetypeGroupId=org.apache.maven.archetypes -DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersio
```

## Essential Commands

| Command | Description |
|---|---|
| `mvn clean` | Cleans the project (removes build artifacts like classfiles etc.) |
| `mvn compile` | Compiles just the production source code |
| `mvn test-compile` | Compiles just the test source code |
| `mvn test` | Tests the project, running unit tests with the Surefire plugin |
| `mvn dependency:tree` | Shows the dependency tree of the project (all dependencies the project uses, and those dependencies dependencies) |
| `mvn install` | Installs the project to the local Maven repo cache |

## Getting Help

You can see the *goals* a plugin has like this:

```
mvn help:describe -Dplugin=compiler
```

and the *parameters/options* which are available on a specific goal like this:

```
mvn help:describe -Dplugin=compiler -Dgoal=compile -Ddetail
```

## Useful Commandline Switches

The following switches can be passed in to the Maven commandline to affect its behaviour.

## Testing

| Example | Description |
| --- | --- |
| `-DskipTests` | Skip running the tests |
| `-Dmaven.test.skip=true` | Skip the whole test side (includes compilation of test source code and execution of tests) |