[maven运行时的配置及命令详解](http://blog.csdn.net/sinat_29581293/article/details/51700093)

你先需要理解maven的生命周期与插件目标这两个概念。

拿Maven clean来说吧。生命周期为clean.插件目标为maven-clean-plugin:clean。

Maven build是这个插件让你自己去配置执行目标的。

Maven clean 清除上一次Maven执行的结果

Maven generate-sources会根据pom配置去生成[源代码](https://www.baidu.com/s?wd=%E6%BA%90%E4%BB%A3%E7%A0%81&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBuHRvrjRkuhFhP1nYujKh0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1cvPWfLrHmYnWbkPj0dPHcd)格式的包

Maven install将项目输出构件部署到本地仓库

**maven最主要的作用有两个方面，一个是对jar包的依赖解决功能，自己管理jar包，另一个功能就是项目的构建，打包部署。现在我觉得最重要的还是maven的生命周期和插件机制，下面就来总结一下吧。**

mvn install 是将你打好的jar包安装到你的本地库中，一般没有设置过是在 用户目录下的 .[m2](https://www.baidu.com/s?wd=m2&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWnvNbnjcsPvcLrH0LuH9W0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHTzrjDsP1bk)\下面。

mvn package 只是将你的代码打包到输出目录，一般的是 target下面。

**Eclipse中maven常用的命令**

　　点击Run As就可以发现几个Maven的命令：

**Maven Build:**

　　这个命令用于编译Maven工程，执行命令后会在target文件夹中的classes中生成对应的class文件。

**Maven Clean:**

　　删除target文件夹，即删除生成的package包以及class等文件。

**Maven Test:**

　　先自动进行编译，在运行所有的[**测试**](http://lib.csdn.net/base/softwaretest)用例。

**Maven install:**

　　发布生成对应的package包。

**注意：**

　　留意上面的命令的作用，build和test都会生成相应的class文件。也就是说，当你新建一个maven工程，或者clean一个maven工程后，如果没有使用这两个命令，直接针对类进行测试，会跑出[**Java**](http://lib.csdn.net/base/java).class.notfound的错误。因为此时还没有编译生成class文件。

　　只有使用了上面的两个命令后，才能针对某个类进行单元测试。

## 什么是maven?

　　Maven是一个用于项目构建的工具，通过它便捷的管理项目的生命周期。即项目的jar包依赖，开发，测试，发布打包。

　　下面我自己总结一下它的几个特点，看了这些特点，也许对maven有更多的了解。

　　1 jar包依赖

　　这个也许会maven最突出的特点了使用maven不需要上网单独下载jar包，只需要在配置文件pom.xml中配置jar包的依赖关系，就可以自动的下载jar包到我们的项目中。这样，别人开发或者使用这个工程时，不需要来回的拷贝jar包，只需要复制这个pom.xml就可以自动的下载这些jar包。

　　而且，我们自己下载jar包，还有可能造成版本的不一致，这样在协同开发的过程中就有可能造成代码运行的不一致。通过使用maven精确的匹配jar包，就不会出现这种问题了。

　　2 项目坐标

　　Maven通过特定的标识来定义项目名称，这样既可以唯一的匹配其他的jar包，也可以通过发布，使别人能使用自己的发布产品。这个标识就被叫做坐标，长的其实很普通，就是简单的xml而已：

复制代码

1 <groupId>com.test</groupId>

2 <artifactId>maventest</artifactId>

3 <version>0.0.1-SNAPSHOT</version>

4 <packaging>jar</packaging>

5

6 <name>maventest</name>

7 <url>http://maven.apache.org</url>

复制代码

**groupId**：所述的项目名称，由于有的项目并不是一个jar包构成的，而是由很多的jar包组成的。因此这个groupId就是整个项目的名称。

**artifactId**：包的名称。

**version**：版本号。

**packaging**：包的类型，一般都是jar，也可以是war之类的。如果不填，默认就是jar。

**name**和**url**，一个是名称，一个是maven的地址。主要就是上面的几个参数。

　　当想要依赖什么jar的时候就可以通过下面的方式依赖：

复制代码

1 <dependencies>

2 <dependency>

3 <groupId>junit</groupId>

4 <artifactId>junit</artifactId>

5 <version>3.8.1</version>

6 <scope>test</scope>

7 </dependency>

8 </dependencies>

复制代码

　　各个属性的内容基本上都是一样的。

　　这里要注意的是jar包的命名规则：

　　artifactId-version[-classifier].packaging

　　比如上面的pom.xml生成的jar包名字就是：maventest-0.0.1-SNAPSHOT.jar。

　　这里的classifier是可选的，但是有的项目可能还需要导出附属的一些文件，如javadoc，source等等，那么这个地方就需要配置一个字符串。一般都是JDKXXX之类的。