Maven3 实战笔记 01 环境配置与使用入门

刘岩

Email:suhuanzheng7784877@163.com

1. 前言

Maven 是 apache 的一个顶级项目，它的出现越来越影响着现在的众多的开源项目，不仅如此，很多公司的很多新项目都采用 Maven 提倡的方式进行管理。Maven 正逐渐侵入我们原先的管理项目的习惯，对于团队的管理，项目的构建，都是一种质的飞跃。当然是我个人的一些项目经验而说的这话。如果原先的团队老大本身的管理非常科学，也有一套其他软件辅助项目的构建、打包、发布等等一系列机制保证。那么 Maven 可能对其并没有实质性的影响或者是质的飞跃。当然在此也并不是说咱们中国人做出来的项目就管理不善……只是说利用 Maven 帮助我们构建项目更科学、更有可维护性、更加模块化、模块功能更加职能单一、做出来的项目更加具有可复用性等等好处。当然笔者也是个刚学习 Maven 的人，我们一起来学习 Maven 到底为我们带来了什么好处。笔者邮箱已经给出，欢迎与笔者进行学术上的交流。

2. Maven 的思想

本来想看完了实战整本，再写总结的，后来觉得还是先写出来吧。哪怕有什么不对的地方再纠正都不晚，就怕到时候看完了一点心得都没有，全忘了就太……所以先将学习的点点滴滴感受写下来，之后结合实例咱们可以反复的推敲。

目前笔者看到的 Maven 的主要功能是：项目构建；项目构建；项目依赖管理；软件项目持续集成；版本管理；项目的站点描述信息管理；

由此可以看出 Maven 最主要的体现在了哪 2 个词？——对了，“项目”和“管理”！加在一起就是项目管理。项目管理是一个很抽象的概念。项目管理既可以指技术上的管理手段，

也可以指“以人为本”的非技术因素。诚然，无论是技术还是非技术，项目的成败最大的责任人其实就是项目经理。这里我们抛开人为因素不提，仅就技术上来说，Maven 提供了一种思想让团队更科学的管理、构建项目。用配置文件的方式对项目的描述、名称、版本号、项目依赖等等信息进行描述。使之项目描述结构清晰，任何人接手的成本比较低。在项目构建时，利用 Maven 的“约定大于配置”的思想，可以比 Ant 脚本构建项目省去不少配置文件的内容。而且一个项目可能依赖于其他的项目和第三方的组件才能顺利完成，Maven 提供了仓库的概念，让这些依赖项放进仓库中，项目想要从仓库中去取，其他项目组也需要，OK，从仓库中去取，不必每个人去开源项目的站点去苦苦搜寻了。如此人员的成本、软件维护的成本、沟通的成本、硬件的成本都降下来了。客户笑了、公司老板也笑了、项目经理笑了、团队里面的人员也笑了、Apache 社区看到这么多人在用也笑了。给笔者的感觉，现在的开源东西越来越向“敏捷开发”、“极限编程”的方向靠拢。通过 Maven 辅助管理项目，更能发现项目中哪些模块是重复的轮子。

3. Maven 的环境搭建

下面我们来看看 Maven 的环境搭建。首先从 Apache 网站下载 Maven。下载 url 是：http://maven.apache.org/download.html。笔者下载的是 Maven 3.0.3 版本。下载下来的包结构如下

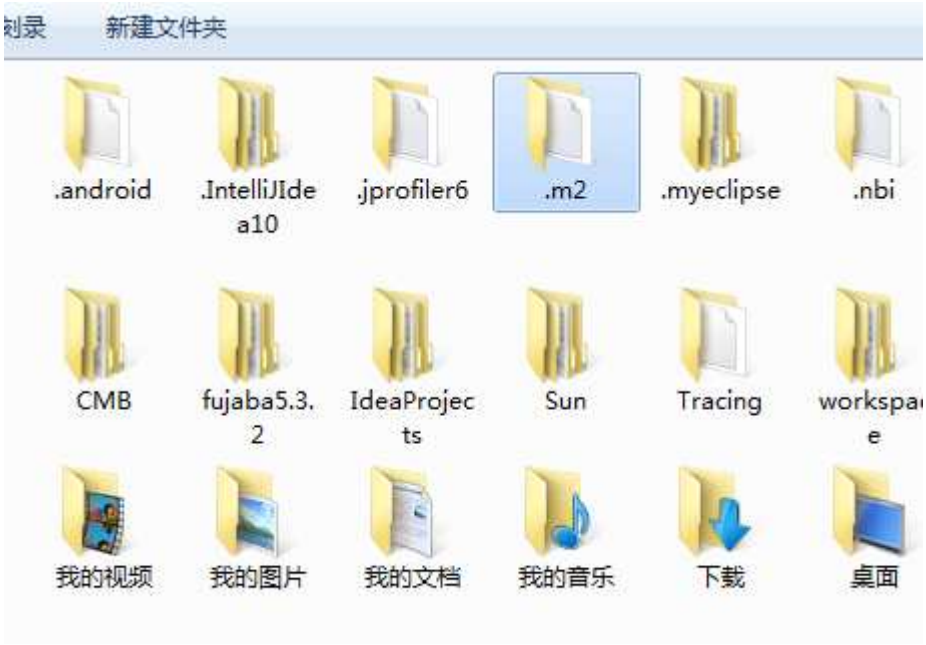
bin 就不用说了，就是 Maven 的一些命令参数，boot 里面是 Maven 自己的类加载器，咱们也不必理会。conf 里面有个 settings.xml 就是本机 Maven 的总配置信息。lib 是 Maven

运行时需要的类库。将 bin 目录和 JDK 的 bin 目录一样设置为系统的 PATH。这样在命令行就能直接运行 Maven 指令了。

保持网络畅通，在命令行执行一条语句

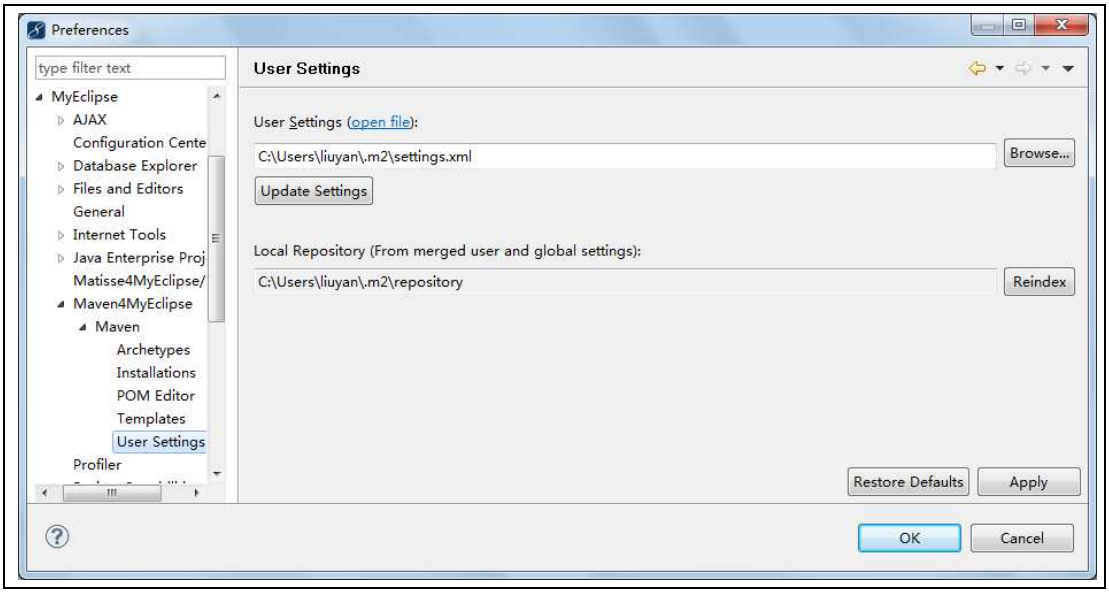
```
mvn help:system
```

会从网上下载很多东西，本地用户的临时文件夹，会生成一个临时 Maven 文件目录，用于存储本地资源仓库。比如在 C:\Users\liuyan 下面有个文件夹叫做.m2



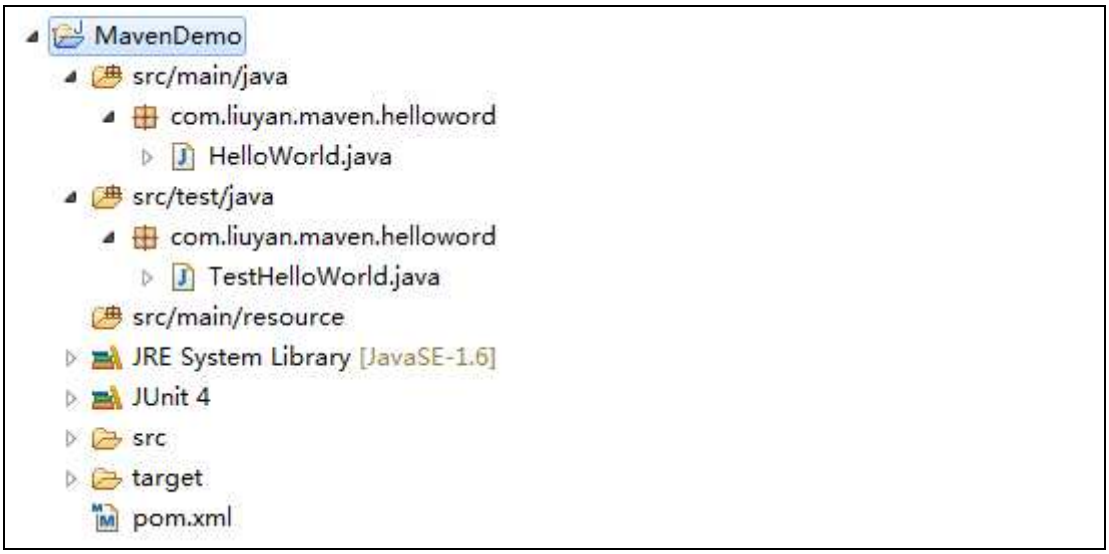
将刚刚提到的 settings.xml 拷贝到 C:\Users\liuyan\.m2 下面，如此 Maven 就算在本机安装好了。

开发工具的选用，笔者使用的是 Myeclipse8.6 版本，此 IDE 已经集成了 Maven 插件 Maven4Myeclipse，仅需要将总配置信息配置上即可。在 Myeclipse 点击 window 菜单，打开 preferences 选项。之后点击到 myeclipse 的 maven4myeclipse 节点，如下图所示。



4. Maven 使用入门

首先我们先写一个非常简单的小项目（我们姑且称之为项目啊），项目结构如下图所示



src.main.java 是项目代码文件夹、src.test.java 是项目单元测试文件夹、src.main.resource 是放置项目资源文件、配置文件文件夹。下面我们看项目代码

一个简单的类 com.liuyan.maven.helloword.HelloWorld

```
package com.liuyan.maven.helloword;

public class HelloWorld {
    public String sayHello() {
        return "hello maven";
    }
```

```
    /**
     * @param args
     */
    public static void main(String[] args) {
        System.out.println(new HelloWorld().sayHello());
    }

}
```

在此项目的根目录下有一个 pom.xml 文件，内容如下

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.liuyan.maven</groupId>
    <artifactId>MavenDemo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
</project>
```

进入控制台，将当前目录 cd 到与 pom.xml 同级的文件夹下面。执行命令

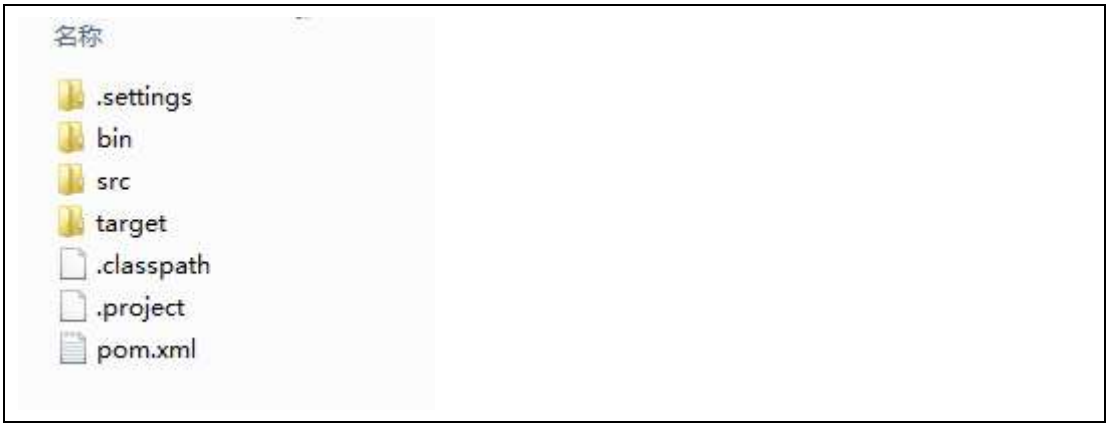
```
mvn clean compile
```

因为笔者写总结前之前下载了相关包，所以此时相关输出 download 包的信息少一些

```
[INFO] Scanning for projects...

[INFO]

[INFO] ------------------------------------------------------------------------

[INFO] Building MavenDemo 0.0.1-SNAPSHOT

[INFO] ------------------------------------------------------------------------

[INFO]

[INFO] --- maven-clean-plugin:2.4.1:clean (default-clean) @ MavenDemo ---

[INFO] Deleting E:\Genuitec\ws\MavenDemo\target

[INFO]
```

```
[INFO] --- maven-resources-plugin:2.4.3:resources (default-resources) @ MavenDem
o ---
[WARNING] Using platform encoding (GBK actually) to copy filtered resources, i.e
. build is platform dependent!
[INFO] skip non existing resourceDirectory E:\Genuitec\ws\MavenDemo\src\main\res
ources
[INFO]
[INFO] --- maven-compiler-plugin:2.3.2:compile (default-compile) @ MavenDemo ---

[WARNING] File encoding has not been set, using platform encoding GBK, i.e. buil
d is platform dependent!
[INFO] Compiling 1 source file to E:\Genuitec\ws\MavenDemo\target\classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 2.044s
[INFO] Finished at: Sat May 28 16:23:07 CST 2011
[INFO] Final Memory: 5M/15M
[INFO] ------------------------------------------------------------------------
```

执行后造成的结果。就是项目的根路径下出现了一个 taget 文件夹



里面就是编译后的 class 类。经理过来说，你需要进行单元测试才能发布出来给大家用。

Ok，我们在源码包——src/test/java 下面开始编写 junit 单元测试类。

单元测试代码如下

```
package com.liuyan.maven.helloword;

import org.junit.Test;
import org.junit.Assert;

public class TestHelloWorld {

    @Test
    public void testSayHello() {
        HelloWorld helloWorld = new HelloWorld();
        Assert.assertEquals(helloWorld.sayHello(), "hello maven");
    }

}
```

之后我们因为使用了 junit 单元测试，那么就是说我们这个项目依赖了它。修改一下

pom.xml 文件内容，如下

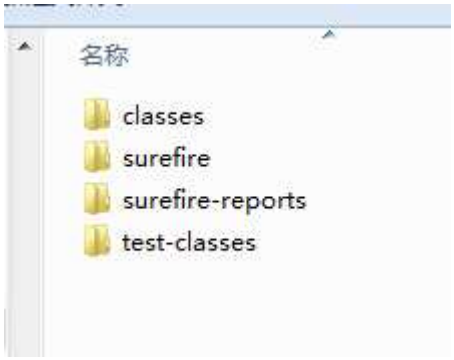
```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.liuyan.maven</groupId>
    <artifactId>MavenDemo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <dependencies>
        <dependency>
```

```
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.7</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

进入命令行，执行命令

```
mvn clean test
```

执行后观察一下 taget 文件夹如下图 ,多出了 test-classes 文件夹和另外 2 个咱们暂时不用去管的文件夹。



之后在观察一下本地的临时仓库 C:\Users\liuyan\.m2\repository ，会多出文件夹 junit ，下载的版本是 4.7。

如果我们想把项目打成 jar 的形式输出出去呢？在项目根目录下执行

```
mvn clean package
```

执行后效果如下

生成了一个 jar 包 ,至于 SNAPSHOT 是快照的意思 ,快照就是项目暂时还不稳定的意思。

打包测试后没问题了，想把此项目当做 Maven 的本地资源仓库，为其他的项目也能提供服务，可以这么做。

执行命令

```
mvn clean install
```

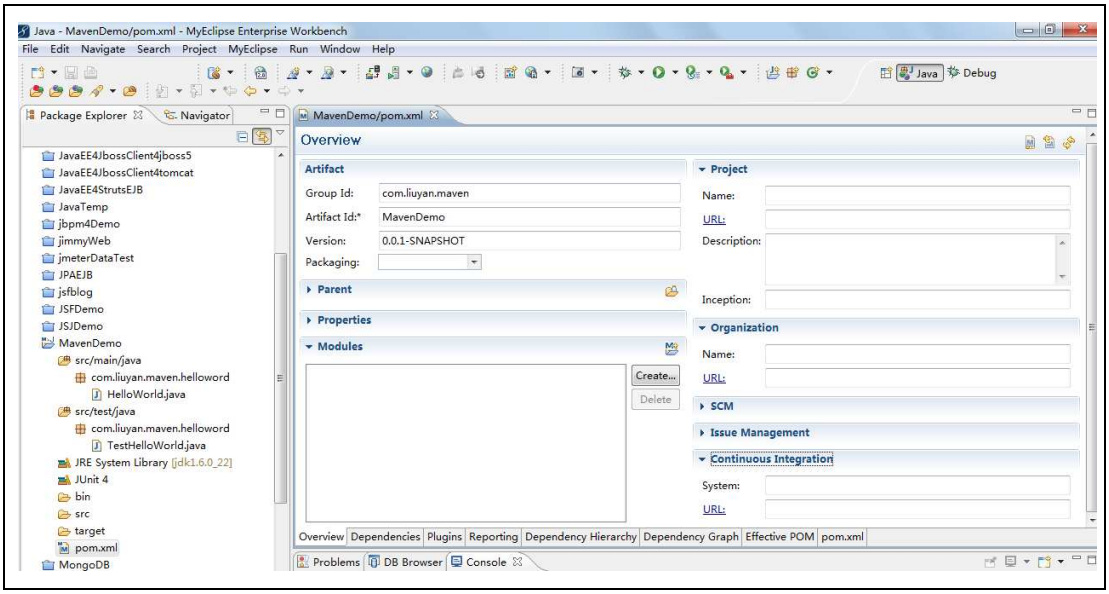
执行后本地的临时库文件多了你自己的这个项目。这样别的项目需要你这个项目提供服务的时候就可以从本地 Mavne 库中提取相应的 jar 了。

5. 利用 IDE 构建 POM.xml 骨架

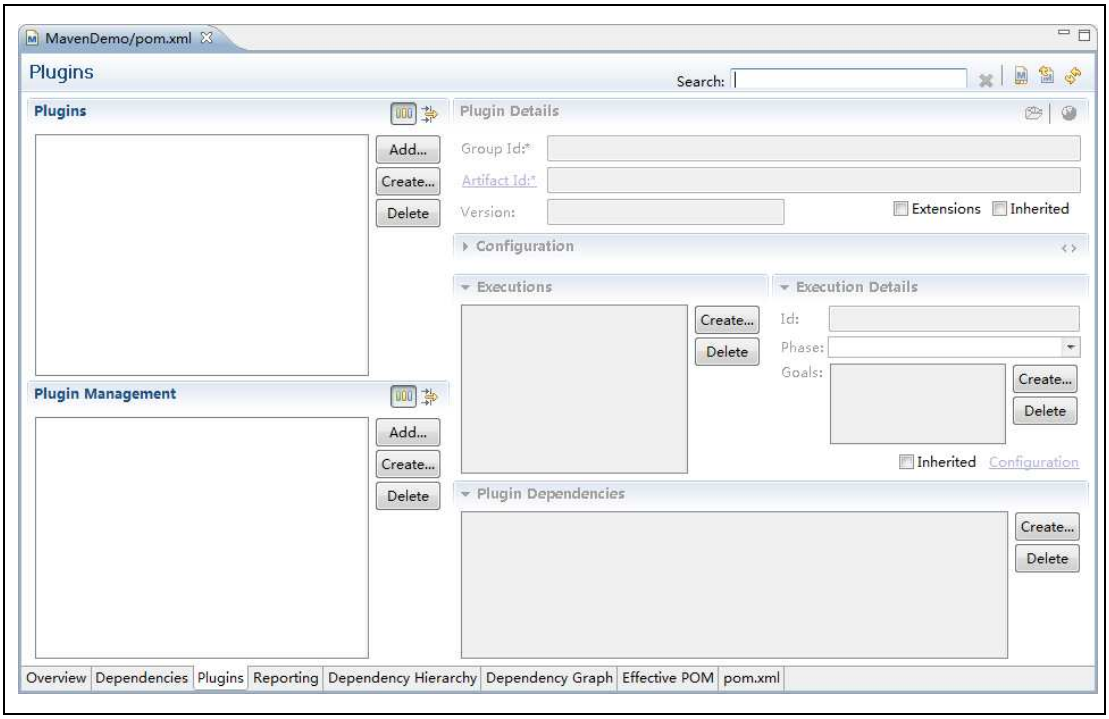
现在的形式是这样，Maven 虽然如火如荼，但是支持 Maven 的各种 IDE 插件确实还有很多不少的缺陷。正所谓名不正则言不顺，总给人感觉 IDE 不支持的开源项目都是“非主流”，呵呵，确实现在 Maven 在 IDE 中存在各种各样的问题，比如就刚才简单的命令在 dos 命令行就可以顺利执行，到了 Myeclipse 怎么就不行了呢。笔者在此仅仅使用 IDE 构建一下 pom.xml 文件，因为个人感觉利用 IDE 执行 Maven 命令其实意义不是很大，因为到了 linux 环境下还是需要手工执行 Maven 命令的。

使用 Myeclipse 工具可以为开发者提供良好的可视化编辑 POM.xml 文件

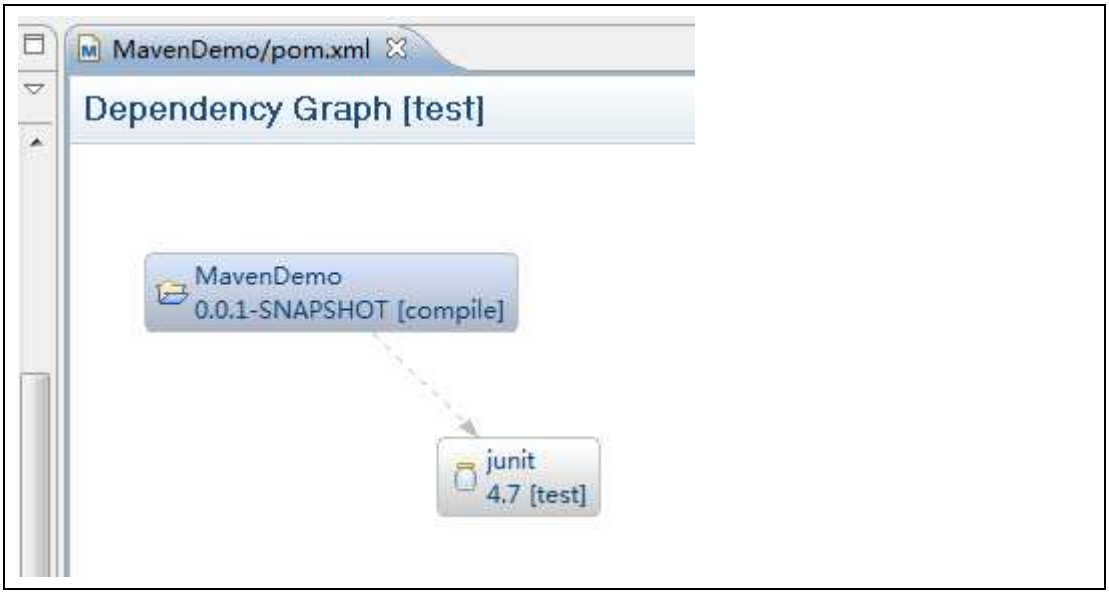
主文件编辑



插件编辑



依赖图形化显示



6. 总结

这算是开张了，本次仅仅介绍了 Maven 的环境搭建和入门命令，还了解了一些 Maven 的思想。之后我们会结合书里面的小案例来一步步的学习 Maven 的特性。