说的简单点，这玩意就是方便开发人员管理项目。

maven的目录结构是src/main/java以及src/test/java，嗯其实就是src文件夹下又放了一个main文件夹与test文件夹。

########################**pom.xml**##############################

pom.xml是maven的必备文件，用于描述项目，与src文件夹同级目录。

Group Id 一般代表组织或是公司 com.jojo

Artifact Id 一般代表项目 HelloWorld

// 为什么要说是一般呢，因为maven项目是存在父子依赖关系的，一个maven项目又由多个maven项目组成。那末，子maven项目pom.xml的Group Id就要描述清楚是哪个公司的哪个项目，Artifact Id说清楚是项目下的哪个模块。

// 可惜，这些都是我猜的。。。。

Version 就是版本号，RELEASE代表稳定版本，其他都不稳定。

* 仓库

这些呢，是maven的必备属性，相当于坐标，能在“中央仓库”唯一标识一个项目。

哦(⊙o⊙)哦，那什么是“中央仓库”呢？存放maven项目的仓库，好像什么都没说啊，那么换个套路。如果一个项目是用maven写的，而且是开源的，而且还想发布出去给别人用，那么这个项目就必须放到“中央仓库”里去。

与之相对，自然也有“本地仓库”这个概念。本地仓库默认位于C盘，在用户文件夹的“.m2/repository”文件夹下，这里头存的都是jar包。用本地仓库是一种很自然的想法，毕竟不可能每次写项目都从远程中央仓库找需要的jar包，速度不允许，中央仓库也支撑不住。每次项目开始编译（mvn compile）时会先从本地仓库找jar包，没有才从中央仓库里去下。

编译完毕后会在创建target目录，该目录与src目录同级，target里放的就是编译完成后的“.class”文件。运行mvn clean命令会清除“.class”文件。

留个问题，如何修改本地仓库的路径？（LocalRepository）

* Maven 常见命令（右击pom.xml）

// 说是这么说，不过现在ecplise都自动集成了，谁还闲着没事开cmd敲命令。

Mvn compile 编译

Mvn clean 清空

Mvn test 测试

Mvn package 打包

Mvn install 把项目安装到本地仓库

* <properties>

用来定义一些变量，因为是xml，所以键值对是标签格式

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<source.jdk.version>1.6</source.jdk.version>

<target.jdk.version>1.6</target.jdk.version>

<spring.version>4.2.5.RELEASE</spring.version>

<swagger.version>1.5.8</swagger.version>

<io.springfox.version>2.5.0</io.springfox.version>

</properties>

中间加个点的只是为了看着清晰一点。

* <scope>

<dependency>的子标签

范围，列举一些常用的

test 比如junit，编译、打包的时候都不会放进去。

设为test，得保证这个jar包有或没有一个样。

provided 比如servlet，编译的时候会算进去，但打包不会，因为这东西服务 器会提供。

设为provided，得保证有东西能提供这个jar包。

compile 默认就是compile，编译、打包都用得着的玩意。

########################**settings.xml**############################

这东西的默认路径说是在 .m2下（C:\Users\administrator\.m2），可尴尬的是我没找到。

ecplise的替换方式：window -> preferences -> Maven -> user settings。

然后自己手动替换一个settings.xml文件

一般公司都会搞一个本地库，要用的jar包统统都从那个本地服务器上下，那么自己折腾项目时，如果还用这个 settings.xml 某些jar包就down不下来了。最好是换一个，不换也行，但要加几个镜像，写三个保险.

<!-- 阿里云仓库 -->

<mirror>

<id>alimaven</id>

<name>aliyun maven</name>

<url>http://maven.aliyun.com/nexus/content/groups/public/</url>

<mirrorOf>central</mirrorOf>

</mirror>

<!-- 中央仓库1 -->

<mirror>

<id>repo1</id>

<mirrorOf>central</mirrorOf>

<name>Human Readable Name for this Mirror.</name>

<url>http://repo1.maven.org/maven2/</url>

</mirror>

<!-- 中央仓库2 -->

<mirror>

<id>repo2</id>

<mirrorOf>central</mirrorOf>

<name>Human Readable Name for this Mirror.</name>

<url>http://repo2.maven.org/maven2/</url>

</mirror>

#####################**ecplise**###################################

**新建Maven项目**

1. 普通Maven项目

-> new maven project

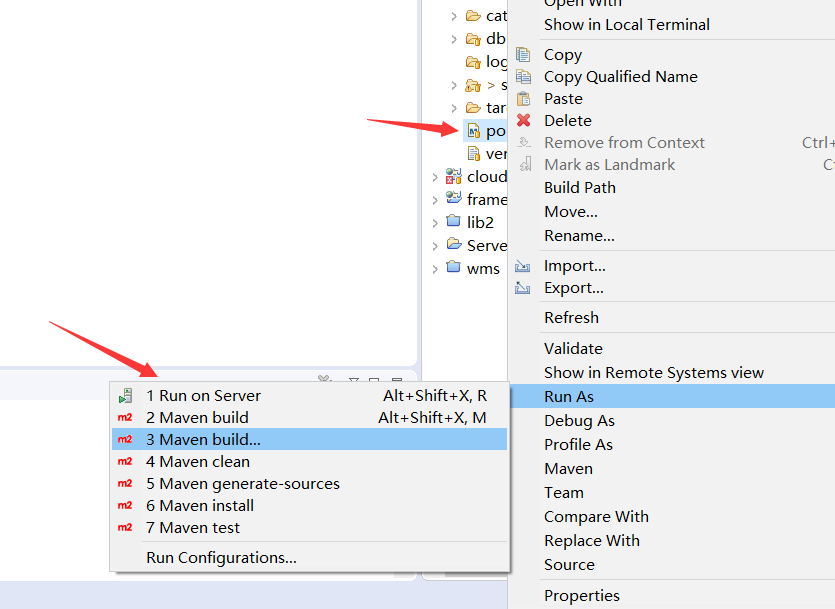
-> 选骨架（所谓骨架就是自动帮你生成想要的目录，比如src/main/java，生成特定文件，比如pom.xml。目前，嗯，我只接触过quickstart）

-> 组织名，项目名，版本号，包名（包名自己懒得填的话，默认是groupId加artifactId）

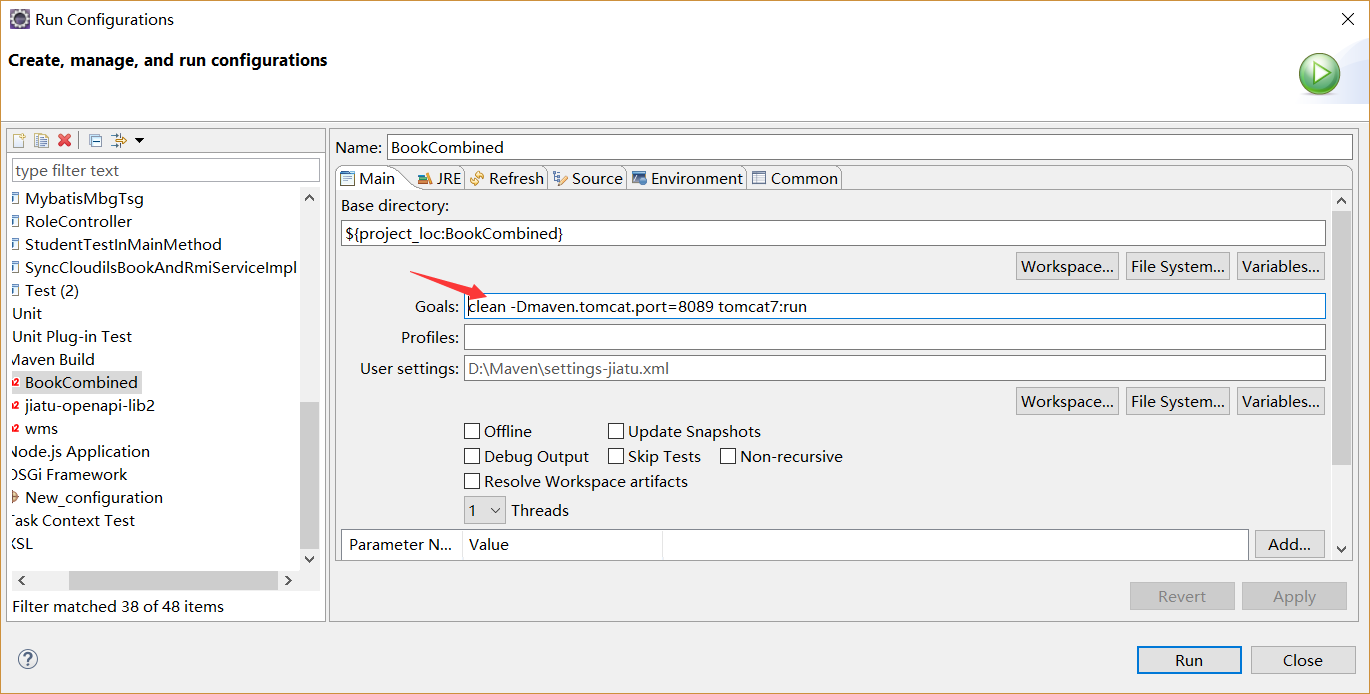
-> 点finish。

src/test/java这个目录还是有用的，不要小看人家。写上一个测试类（XXXTest）然后在这个测试类里写方法，写完后在加上@Test注解。运行mvn test命令会直接运行标了注解的方法，结果显示在控制台里，最屌的是如果测试失败了还会生成一个错误报告。。。。

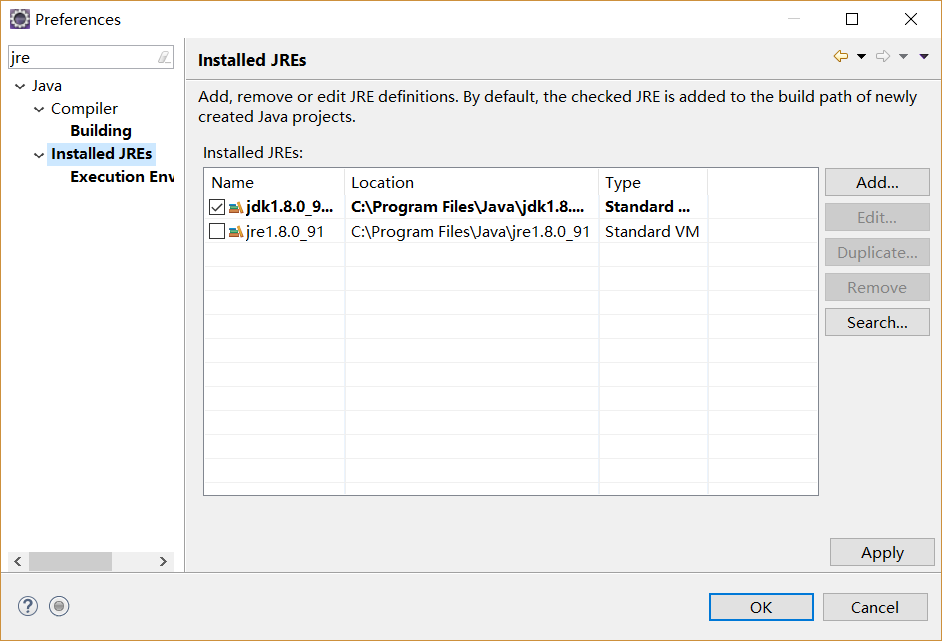
当然ecplise不用这么烦，右键项目 -> run as -> 直接就有maven test选项，鼠标单击就好。但更建议右键pom.xml，可以过滤一些看不懂的东西。



这里面是可以输其他命令的，比如package



maven3开始，必须运行在jdk上，什么意思呢，看图，没有什么比看图来的更实在的了。



2. maven web项目

* 做法一

-> File

-> New

-> Maven Project

-> create a simple project（skip archetype selection）

-> 填基础信息，注意Packaging的值为war -> finish

-> 右键properties -> Project Facets

-> Java 改为1.7，去掉“Dynamic Web Module”的钩，保存关闭

-> 再次打开，Dynamic Web Module选择3.0

-> 左下角出现超链接，点进去，直觉会告诉我该怎么做的，保存关闭

-> 右键properties -> DeployMent Assembly

-> 删掉2个test与WebContent，添加webapp

-> 将WebContent的内容复制到src/main/webapp，再删除WebContent

-> 添加 Tomcat library

* 做法二

-> New Maven Project（骨架选择webapp）

-> 新建dynamic web project，将META-INF与WEB-INF文件夹复制粘贴到maven项目中。

怎么进入还真不是重点，相关参数的配置比较在意，我自己大概列举了一些：

JRE System Library 用最新的

Java Compiler 用最新的

Project Facets 用最新的（这三个版本要一致，同时生成WebContent）

META-INF 置余webapp目录

WEB-INF 置余webapp目录

Deployment Assembly 部署src/main，以及webapp

########################## **error** ###########################

* 明明maven dependencies里有jar包，但没法import

有些jar包依赖了其他的jar包，而这些jar包可能版本冲突了。A 依赖 C的1.2版本，B 依赖C的1.3版本

用IDE 看一下 maven的依赖关系图，然后把冲突的版本都exclude 出去

* missing artifact

jar包down不下来，查看settings.xml里的仓库地址及镜像

jar包down下来了，唉，这个原因多了去了。。。。先clean，没用就close project在open project。没用的话再删.lastUpdated文件。

是在不行 ... ... 手动down下来然后放到本地库。

* Archive for required library : 'D:/Maven/repo/m2/org/springframework/spring-core/4.3.10.RELEASE/spring-core-4.3.10.RELEASE.jar' in project 'demo' cannot be read or is not a valid ZIP file

重点是这个：cannot be read or is not a valid ZIP file

先定位出错的jar包，如果jar包什么问题都没有，那么clean，update，close/open。或者把jar包删了重下。

* File encoding has not been set, using platform encoding GBK, i.e. build is platform dependent!

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

* invalid LOC header (bad signature)

jar包删了重下