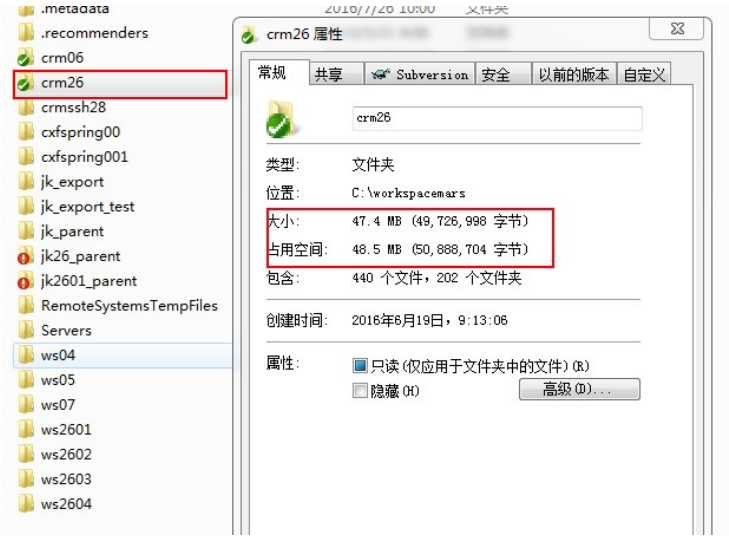
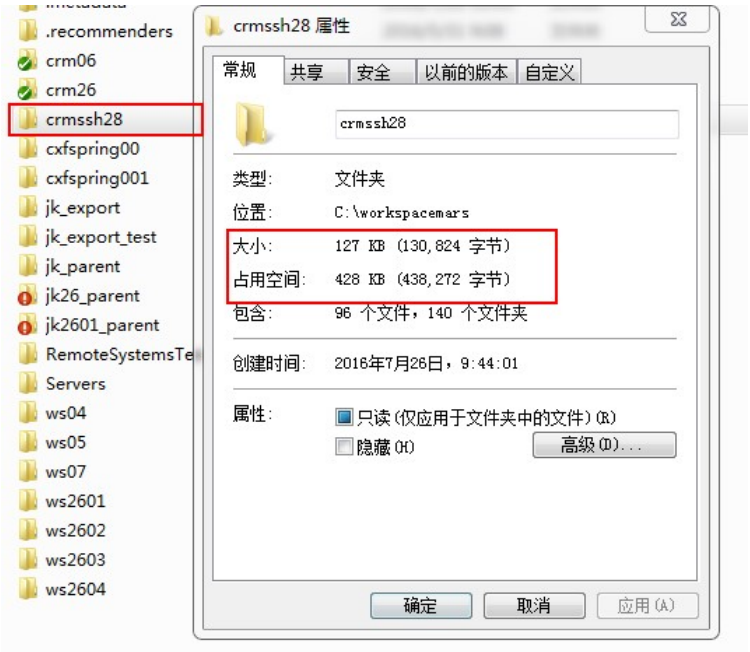
Maven 第一天

一、 maven 的好处？

1. 使用传统的 web 项目开发的 crm 大小

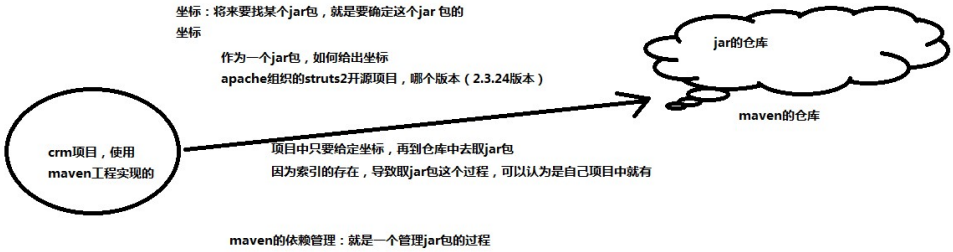


2. 同样的项目使用 maven，它的大小



小结：同样的项目使用 maven 工程来实现，它的项目源码很小，可以初步推断它里面一定没有 jar 包，继续思考，没有 jar 包的项目怎么可能运行呢？

分析 jar 包查找的过程



二、 分析出 maven 优点是如何实现的？

1. 依赖管理

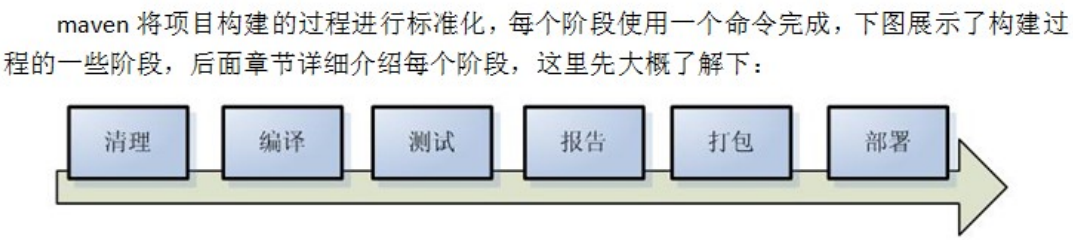
就是对 jar 包管理的过程，管理的过程参考上面的图分析过程

2.一键构建

（编译-----测试------打包-----安装-----部署 ）

什么是项目构建？

指的是项目从编译-----测试------打包-----安装-----部署 整个过程都交给 maven 进行管理，这个过程称为构建

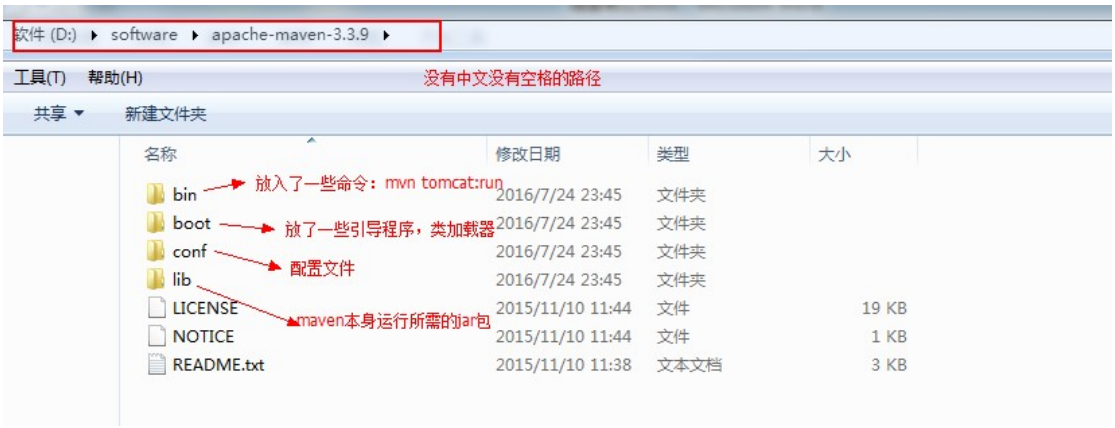


三、 Maven 的安装

- 下载

从该网站 http://maven.apache.org/download.cgi 下载 maven3.3.9 版本

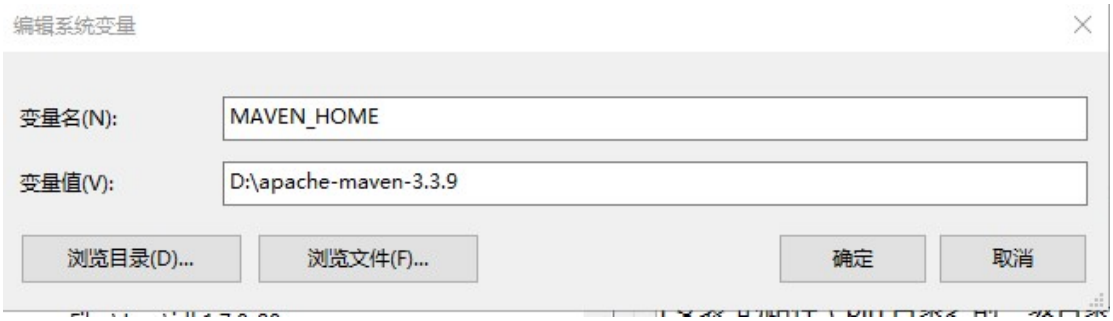
下载之后，解压它，得到一个如下的目录结构：



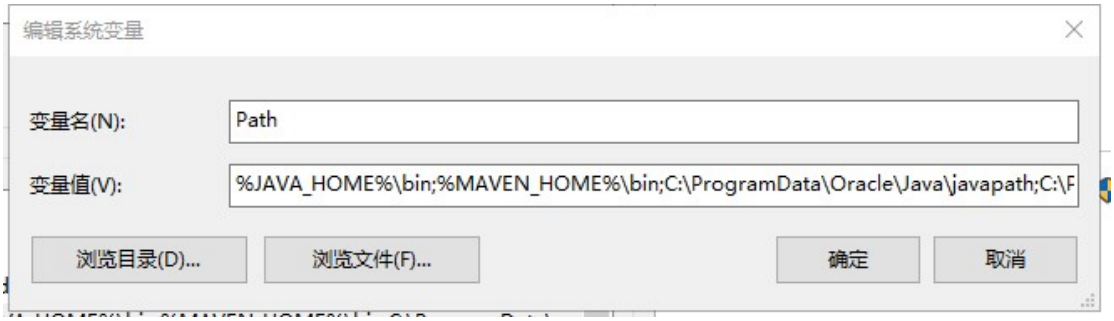
1.配置它的环境变量

电脑上需安装 java 环境，安装 JDK1.7 + 版本 （将 JAVA_HOME/bin 配置环境变量 path ）

配置 MAVEN_HOME ，变量值就是你的 maven 安装 的路径（bin 目录之前一级目录）



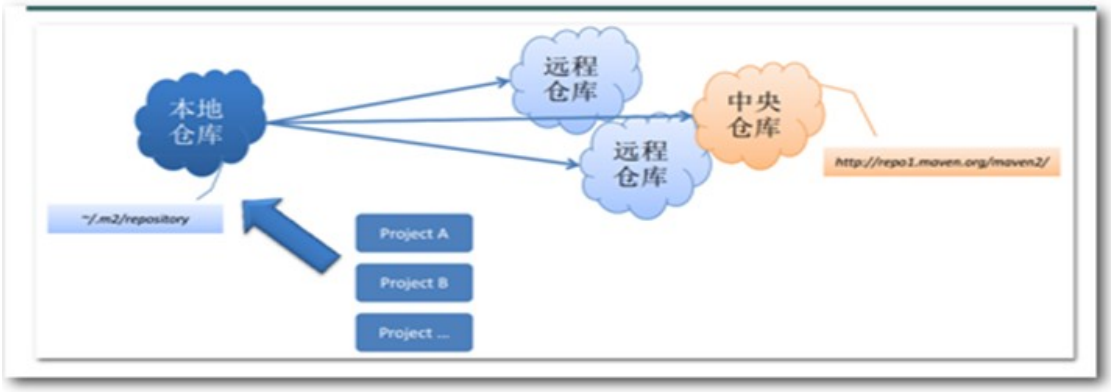
将 %MAVEN_HOME%\bin 加入环境变量 path



通过 mvn -v 命令检查 maven 是否安装成功，看到 maven 的版本为 3.3.9 及 java 版本为 1.7 即为安装成功。

2.仓库的配置

2.1 仓库的分类



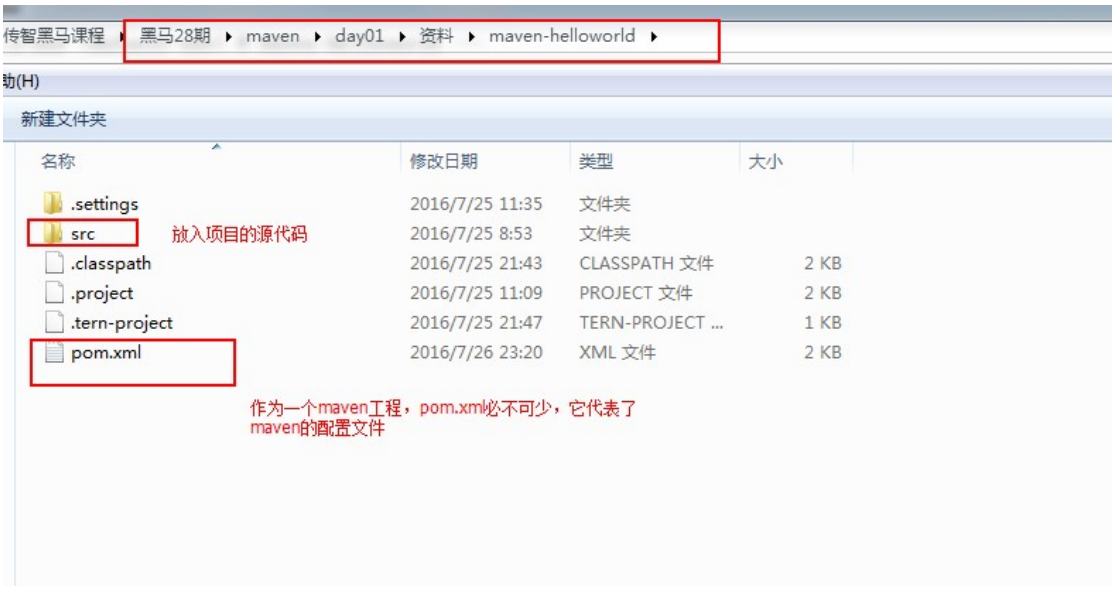
2.2 本地仓库的配置

这个配置文件：maven 安装路径下\conf\settings.xml 文件中

```
    -->
  <localRepository>D:\repository</localRepository>
  <!-- interactiveMode
```

本地仓库的路径

四、 分析出 maven 工程的标准目录结构



Maven 目录结构的规范

五、 Maven 命令？

compile

compile 是 maven 工程的编译命令，作用是将 src/main/java 下的文件编译为 class 文件输出到 target 目录下。

cmd 进入命令状态，执行 mvn compile，如下图提示成功：

```
F:\develop\maven\project\maven-helloworld>mvn compile
[INFO] Scanning for projects...
[WARNING]
[WARNING] Some problems were encountered while building the effective model for cn.itcast.maven:maven-helloworld:war:0.0.1-SNAPSHOT
[WARNING] 'build.plugins.plugin.version' for org.apache.maven.plugins:maven-compiler-plugin is missing. @ line 28, column 12
[WARNING]
[WARNING] It is highly recommended to fix these problems because they threaten the stability of your build.
[WARNING]
[WARNING] For this reason, future Maven versions might no longer support building such malformed projects.
[WARNING]
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building 第一个maven工程 0.0.1-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ maven-helloworld ---
[WARNING] Using platform encoding (GBK actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ maven-helloworld ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to F:\develop\maven\project\maven-helloworld\target\classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 0.985 s
[INFO] Finished at: 2016-07-23T16:57:49+08:00
[INFO] Final Memory: 9M/22M
[INFO] ------------------------------------------------------------------------
```

查看 target 目录，class 文件已生成，编译完成。

test

test 是 maven 工程的测试命令，会执行 src/test/java 下的单元测试类。

cmd 执行 mvn test 执行 src/test/java 下单元测试类，下图为测试结果，运行 1 个测试用例，全部成功。

clean

clean 是 maven 工程的清理命令，执行 clean 会删除 target 目录的内容。

package

package 是 maven 工程的打包命令，对于 java 工程执行 package 打成 jar 包，对于 web 工程打成 war 包。

install

install 是 maven 工程的安装命令，执行 install 将 maven 打成 jar 包或 war 包发布到本地仓库。

从运行结果中，可以看出：

当后面的命令执行时，前面的操作过程也都会自动执行，

5.5.1 三套生命周期

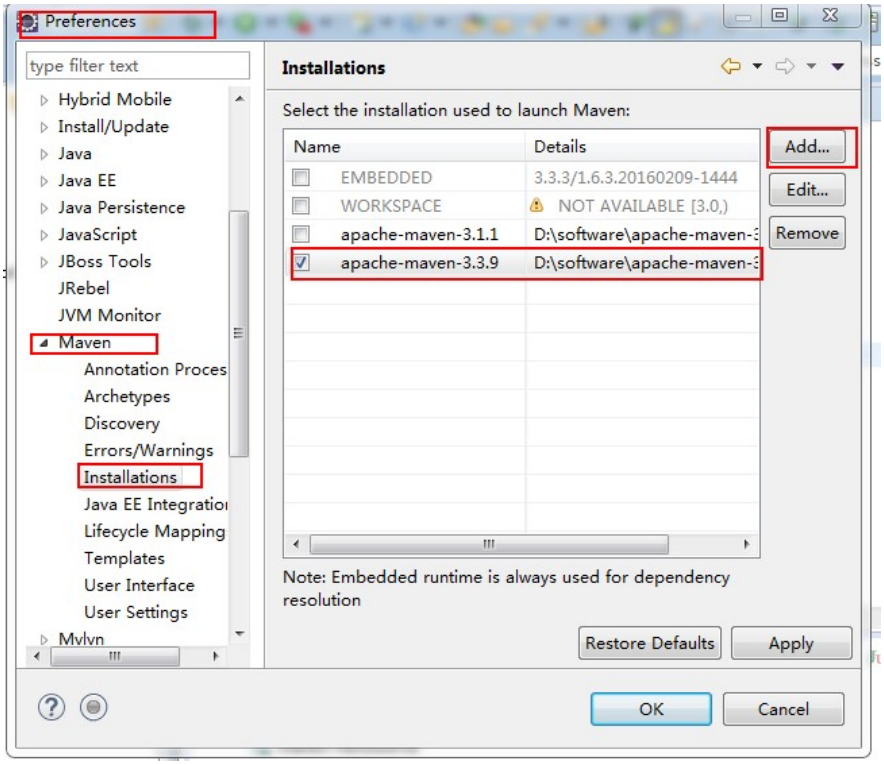
maven 对项目构建过程分为三套相互独立的生命周期，请注意这里说的是“三套”，而且“相互独立”，这三套生命周期分别是：

Clean Lifecycle 在进行真正的构建之前进行一些清理工作。
Default Lifecycle 构建的核心部分，编译，测试，打包，部署等等。
Site Lifecycle 生成项目报告，站点，发布站点。

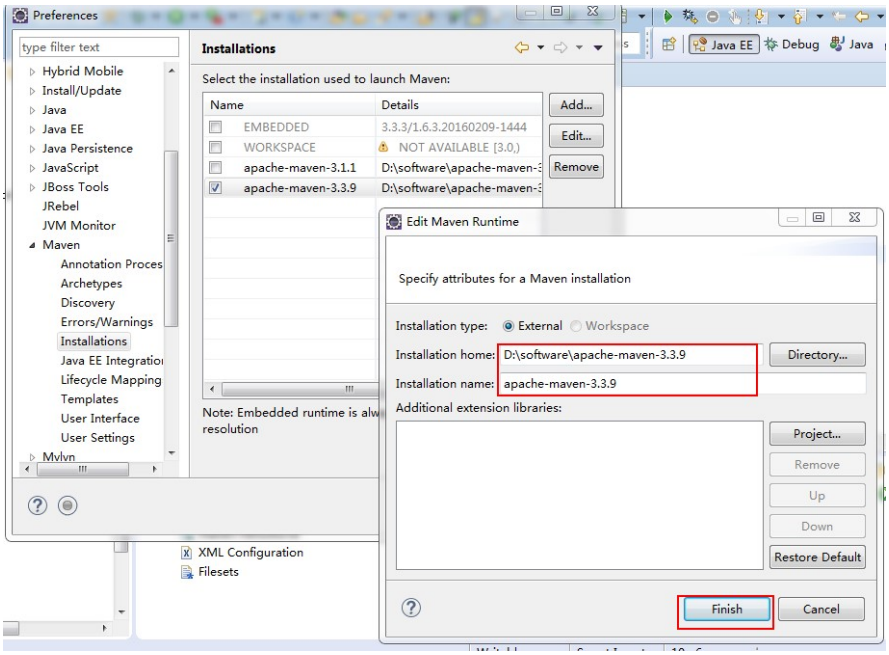
六、 如何使用 eclipse 开发 maven 项目

1. m2e 的插件

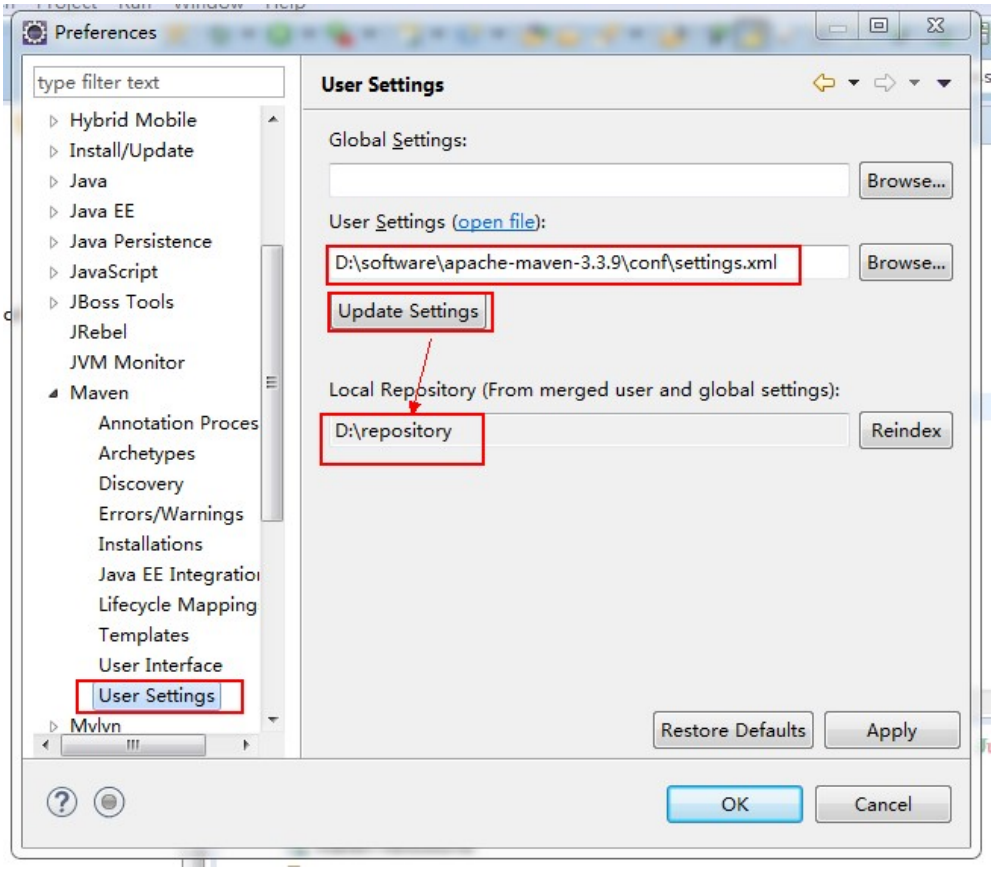
   因为使用的 eclipse 版本比较高，所以它自带了有 maven 插件。但是需要配置 Maven 工具



点击"Add",进入下面的页面

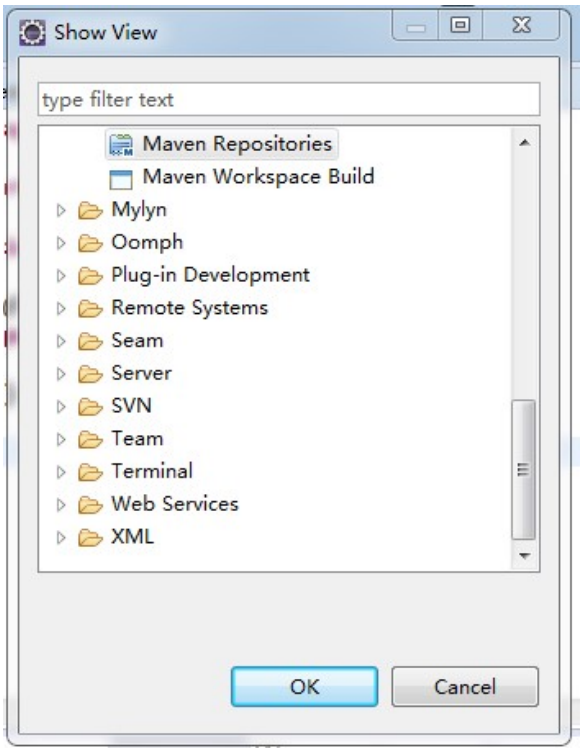


2. 在 eclipse 中配置仓库的位置，设置好本地仓库后需要重新启动 Eclipse

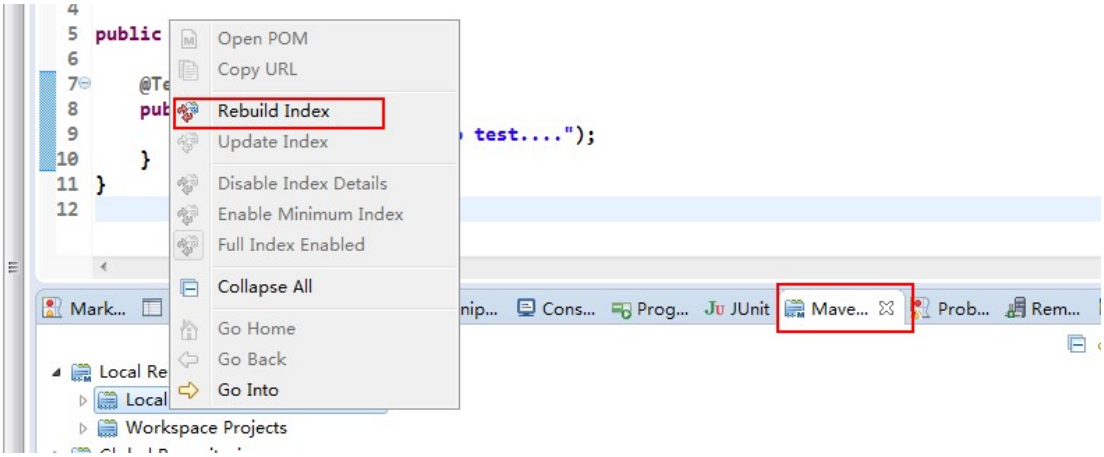


3. 构建索引

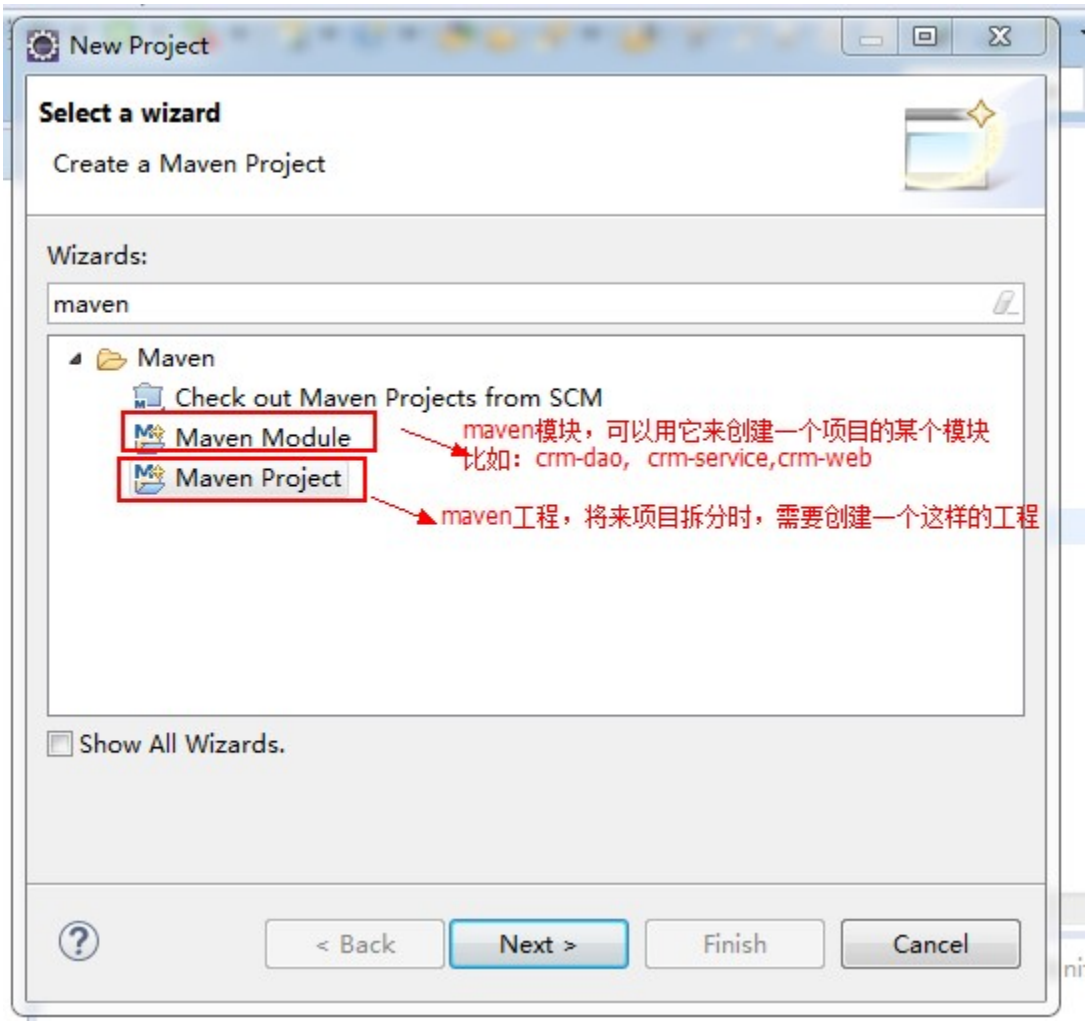
   Window---show view ------other----------maven Repositories



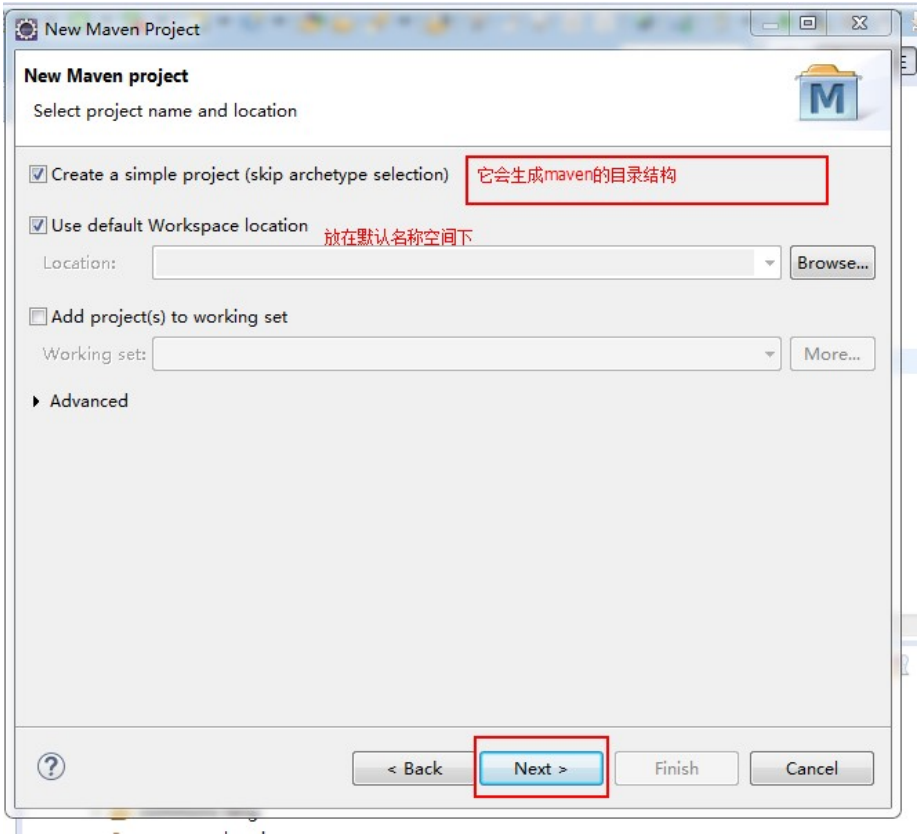
4. 重构索引



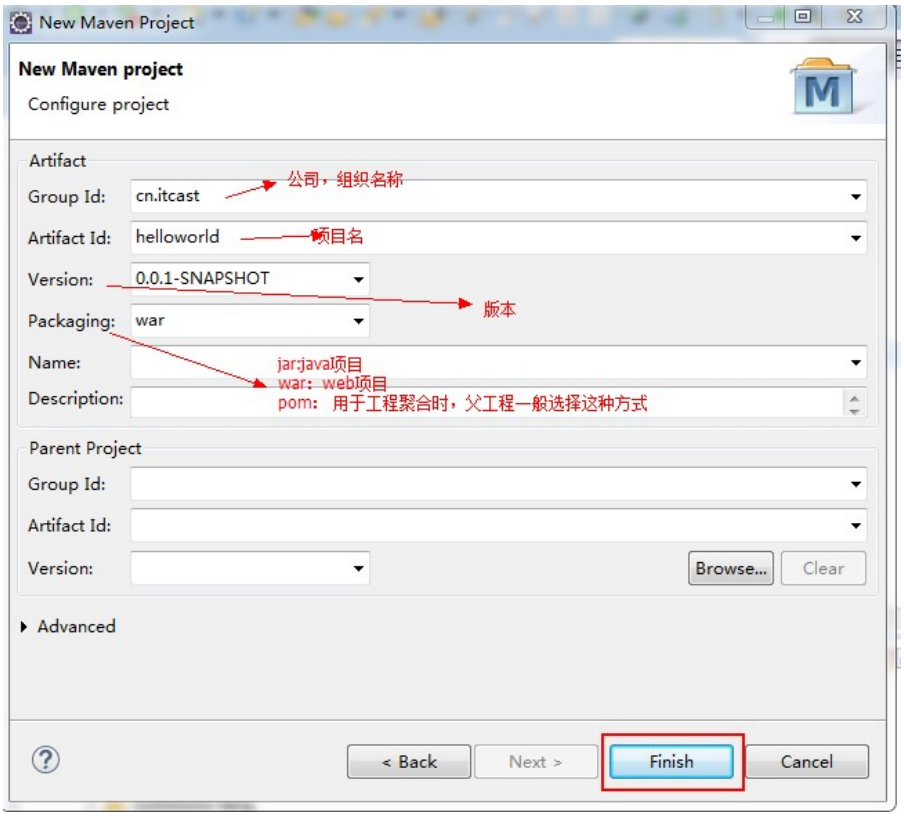
5. 在 eclipse 中创建一个 maven 工程



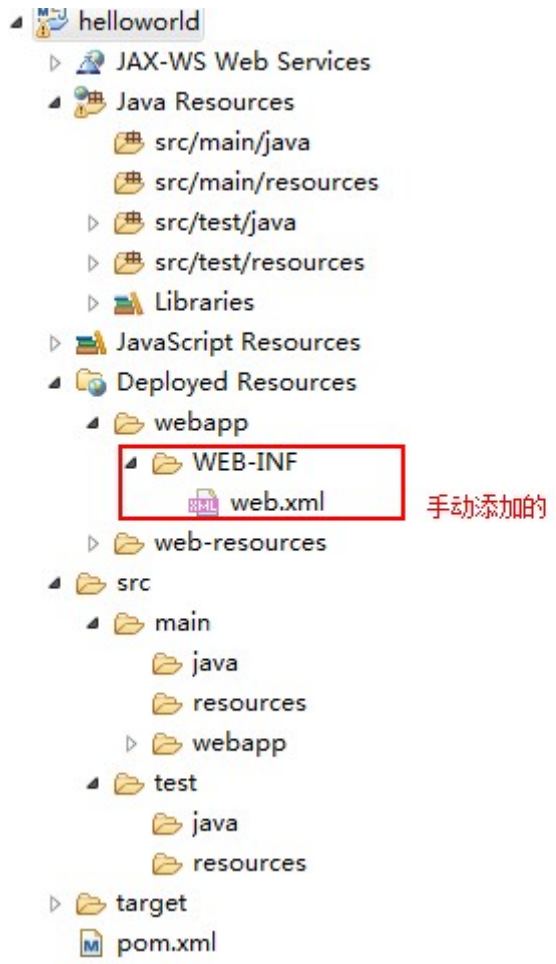
6. 选择 maven project



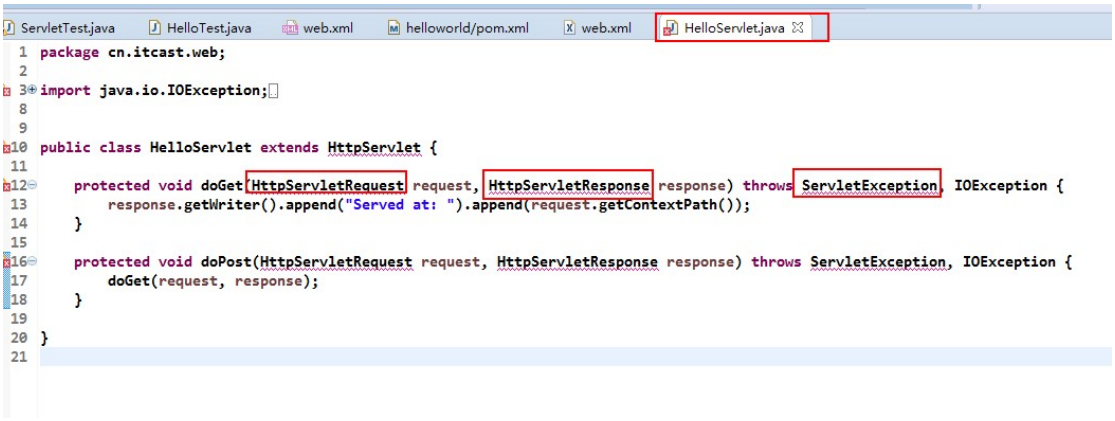
7. 点 next 进入下面的界面



8. 可以看到一个 helloworld 工程，但报错(添加下面的内容就 OK 了)

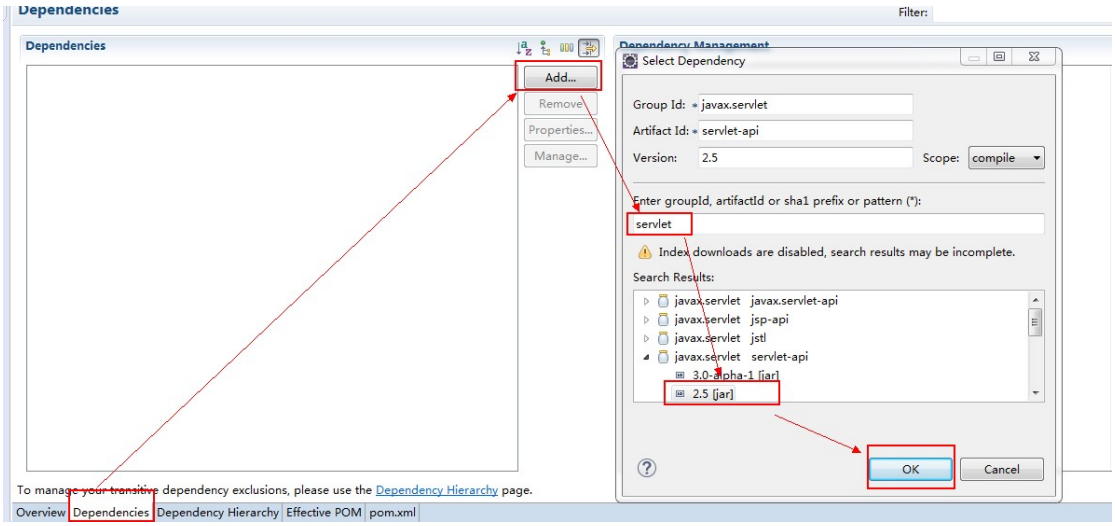


9. src/java/main 创建了一个 Servlet，但报错

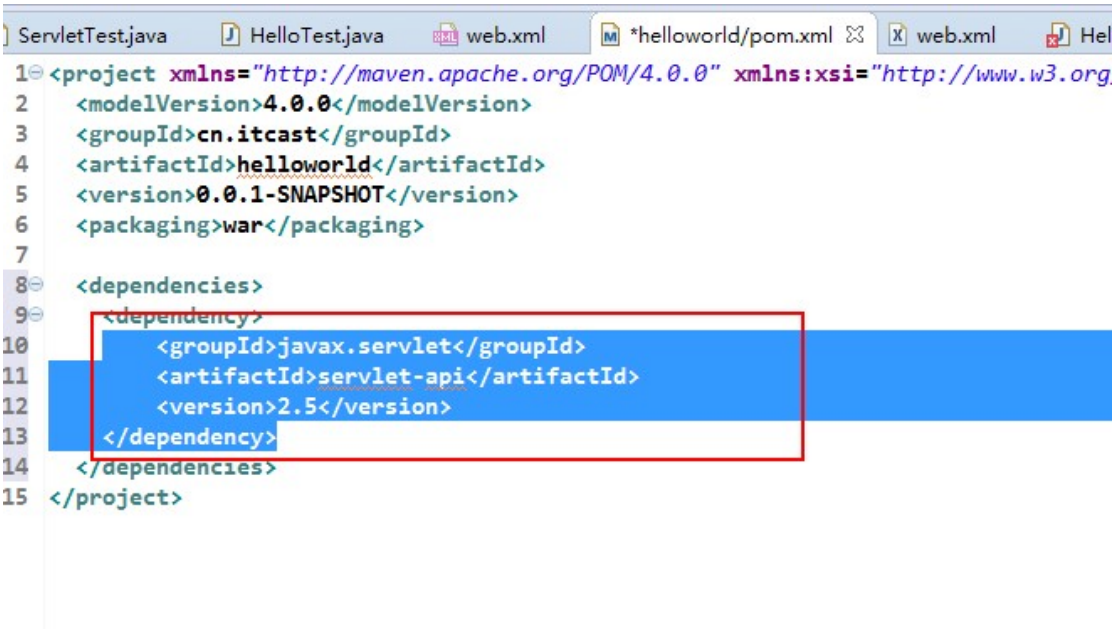


10. 要解决问题，就是要将 servlet-api-xxx.jar 包放进来，作为 maven 工程应当添加 servlet 的坐标，从而导入它的 jar

直接打开 helloworld 工程的 pom.xml 文件，再添加坐标



添加后自动生成的结果如下：



添加 jar 包的坐标时，还可以指定这个 jar 包将来的作用范围

依赖范围包括：

- ✓ compile：编译范围，指 A 在编译时依赖 B，此范围为默认依赖范围。编译范围的依赖会用在编译、测试、运行，由于运行时需要所以编译范围的依赖会被打包。

- ✓ provided: provided 依赖只有在当 JDK 或者一个容器已提供该依赖之后才使用，provided 依赖在编译和测试时需要，在运行时不需要，比如：servlet api 被 tomcat 容器提供。

- ✓ runtime：runtime 依赖在运行和测试系统的时候需要，但在编译的时候不需要。比如：jdbc 的驱动包。由于运行时需要所以 runtime 范围的依赖会被打包。

- ✓ test：test 范围依赖 在编译和运行时都不需要，它们只有在测试编译和测试运行阶段可用，比如：junit。由于运行时不需要所以 test 范围依赖不会被打包。

- ✓ system：system 范围依赖与 provided 类似，但是你必须显式的提供一个对于本地系统中 JAR 文件的路径，需要指定 systemPath 磁盘路径，system 依赖不推荐使用。

依赖范围	对于编译classpath有效	对于测试classpath有效	对于运行时classpath有效	例子
compile	Y	Y	Y	spring-core
test	-	Y	-	Junit
provided	Y	Y	-	servlet-api
runtime	-	Y	Y	JDBC驱动
system	Y	Y	-	本地的，Maven仓库之外的类库

```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <version>2.5</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.servlet.jsp</groupId>
    <artifactId>jsp-api</artifactId>
    <version>2.0</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.9</version>
    <scope>test</scope>
</dependency>
```

11. 调整 jdk,在 pom.xml 文件中添加如下配置：

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <configuration>
                <source>1.7</source>
                <target>1.7</target>
                <encoding>UTF-8</encoding>
            </configuration>
        </plugin>
    </plugins>
</build>
```

如何停掉 tomcat：run

先查：netstat -ano|findstr 8080

再杀：tskill pid