Maven 实验手册

第一部分 基本概念和操作

一、mvn compile

1. 创建一个 user-dao 项目，从空白目录开始，按照 maven 要求创建 main, test 等目录
包名：edu\zju\cst\myproject\model

2. 打开 eclipse，创建一个 Java 项目（这个例子里 eclipse 仅用来编辑 java 文件），在 eclipse 里编辑 java 代码

3. 创建一个类 User, 加上若干属性: id, name, address，并额外提供一个方法

```
public String sayHello(String guestName){
    return "Hello "+ guestName;
}
```

4. 在项目的根目录里创建一个 pom.xml

```
<?xml version="1.0" encoding="utf-8"?>
<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>edu.zju.cst.myproject</groupId>
    <artifactId>user-dao</artifactId>
    <version>0.0.1-SNAPSHOT</version>
</project>
```

掌握 pom.xml 文件的基本格式

5.了解版本命令的方法
0.0.1-SNAPSHOT
SNAPSHOT
Alpha：是内部测试版,一般不向外部发布,会有很多 Bug.一般只有测试人员使用。
Beta：也是测试版，这个阶段的版本会一直加入新的功能。在 Alpha 版之后推出。
RC：(Release Candidate) 顾名思义么！用在软件上就是候选版本。系统平台上就是发行候选版本。RC 版不会再加入新的功能了，主要着重于除错。
GA: General Availability,正式发布的版本，在国外都是用 GA 来说明 release 版本的。

6. 在命令行界面里进入到项目的根目录，执行 mvn compile
观察界面输出，同时根目录下多了一个 target 目录

二、mvn test 及其他几个命令

1. 在 test 目录的对应位置下，创建 UserTest.java

```
package edu.zju.cst.myproject.model;

import org.junit.*;
import static org.junit.Assert.*;

public class UserTest{

  @Test
  public void testSayHello(){
        User user = new User();
        String expected = "Hello world";;
        String actual = user.sayHello("world");
        assertEquals(expected, actual);
  }
}
```

这个时候由于，没有引入 JUnit 包，仍然是有编译错误的。

2. 打开中央仓库网站 http://search.maven.org/#browse
搜索 JUnit，随便找一个版本，比如 4.3
获得 dependency 写法，加入到 pom 中

```
    <dependencies>
```

```
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.3</version>
        </dependency>
    </dependencies>
```

此时 eclipse 里仍然是有编译错误的，因为不是一个 maven 工程。

3. 在命令行界面里执行任意 mvn 命令，会自动下载 junit 的 jar 包
执行 mvn test 后
观察 target 目录下会生成多个文件，包括编译出来的 class 和测试报告。
如果测试不通过，可以查看问题细节。

4. 执行其他几个 maven 命令
mvn compile
mvn test
mvn clean
mvn package
mvn install

三、安装、配置

1. 掌握 maven 里仓库的概念，并知道默认的仓库位置，以及如何配置自定义的位置。
默认的位置在 C:\Users\用户\.m2\repository
可以在 conf\settings.xml 里配置
<localRepository>D:/Program Files/apache-maven-3.0.5/repo</localRepository>

2. 执行完 install 后，查看仓库里多了相关的 jar 包。

3. 了解中央仓库的概念，以及是在哪里设置了中央仓库
lib/maven-model-builder-3.0.5.jar
这个 jar 包里有一个 pom-4.0.0.xml 设置了中央仓库

```
<repositories>
  <repository>
    <id>central</id>
    <name>Central Repository</name>
    <url>http://repo.maven.apache.org/maven2</url>
    <layout>default</layout>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
  </repository>
</repositories>
```

4 每次要自己创建目录比较烦，maven 里提供了自动创建骨架功能
随便创建一个目录，执行 mvn archetype:generate
查看生成的目录和文件

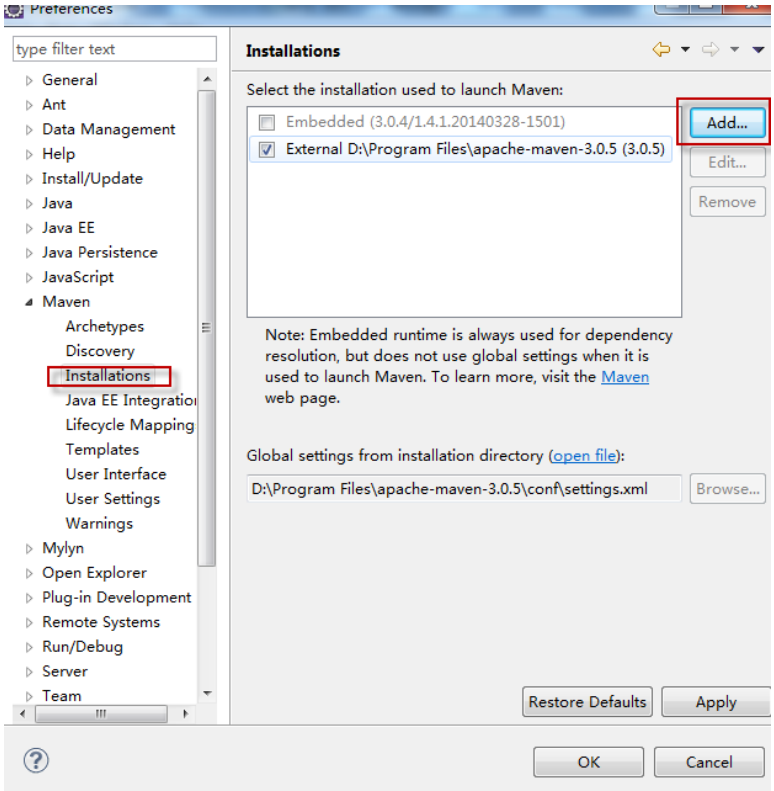
5. 也可以在命令里加上参数
mvn archetype:generate -DgroupId=edu.zju.cst.myproject -DartifactId=module2 -Dversion=0.0.1-SNAPSHOT

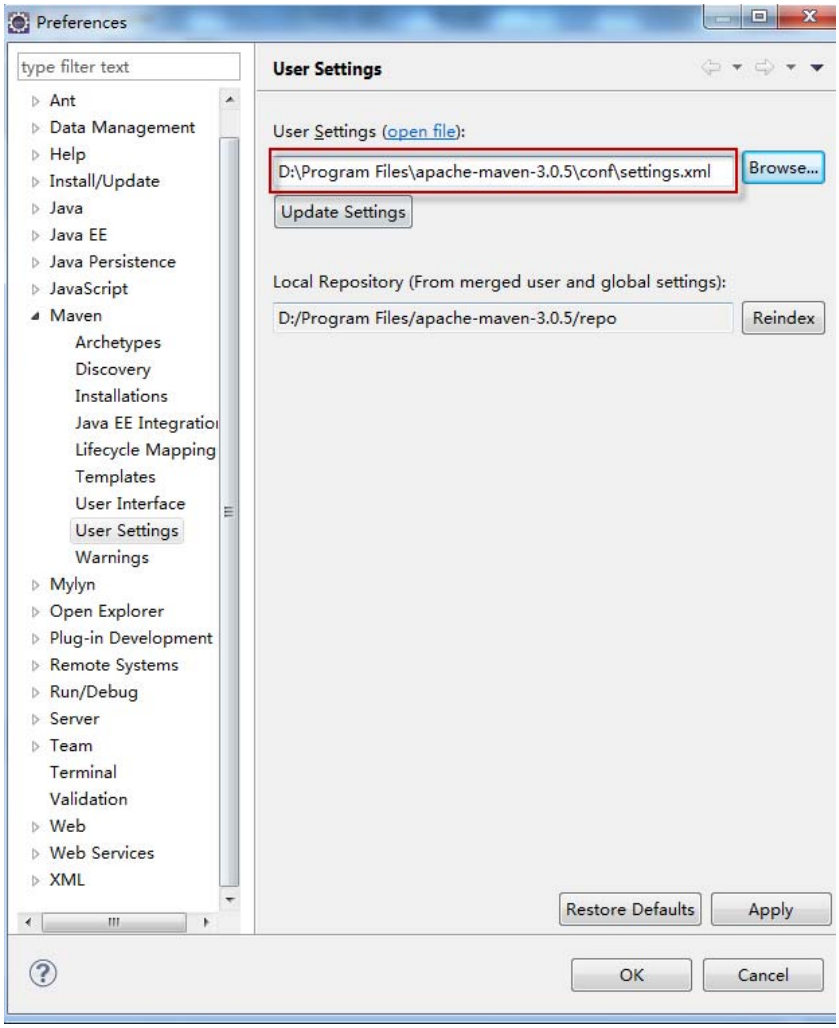
可能会出现问题，检查是否有拼写错误，或者多了空格之类的

```
[ERROR] The goal you specified requires a project to execute but there is no POM in this directory (R:\jeffy\programming\sandbox\xjmaven). Please verify you invoked Maven from the correct directory. -> [Help 1]
```

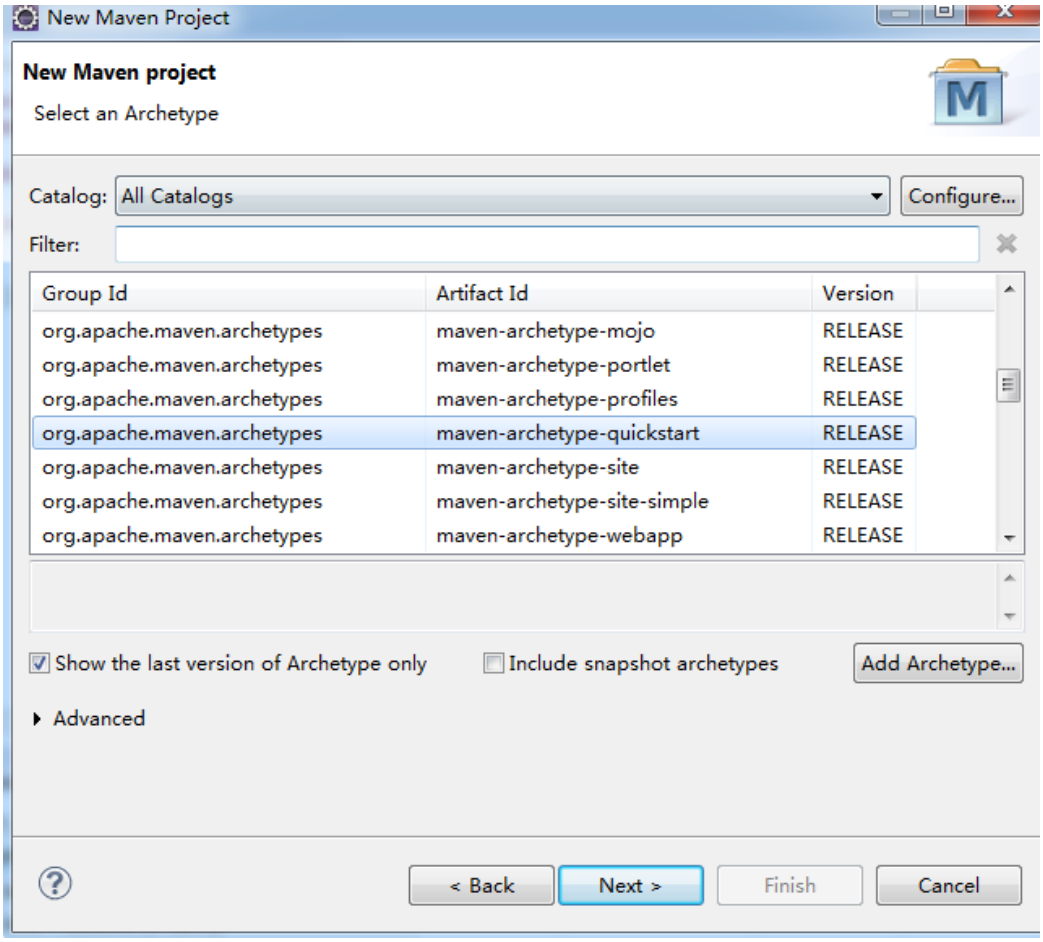
四、m2eclipse

1. Eclipse 插件：m2eclipse
安装好之后在 Eclipse 里配置
使用自己安装的 maven，不要使用 eclipse 自带的





2. 在 eclipse 里创建 maven 项目



资源包的位置
src/main/java/resources

3. 将原来的 Java 项目升级成 maven 项目
Configure -> Configure to Maven Projects

第二部分 Eclipse 中开发简单项目

一、开发 DAO 层

在上面介绍的 user-dao 的基础上继续。

1. 在 model/user 类加上必要的注解，并确保时生成 get/set，及构造方法

```
@Entity
@Table(name = "user")
public class User {

    @Id
```

```
    @GeneratedValue
    public int getId() {
        return id;
    }
```

2. 此时的 User.java 文件是存在编译错误的（缺少相应的包）。
需要通过配置 pom.xml 获取相应的依赖包。

去中央仓库找 hibernate 的依赖包（搜索 hibernate-core），并写到 pom.xml 里

```
<dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
        <version>4.3.6.Final</version>
</dependency>
```

3. 在 dao 目录下开发 IUserDao 和 UserDao 类

```
public interface IUserDao {
    public void add(User user);
    public User getUserByName(String userName);
    public void delete(User user);
}

public class UserDao implements IUserDao {
    public void add(User user) {
    }
    public User getUserByName(String userName) {
        return null;
    }
    public void delete(User user) {
    }
}
```

4. 引入辅助类
util\HibernateUtil.java

在 main/resources 下面添加
hibernate.cfg.xml
log4j.properties

5. 补充完整 UserDao 类的开发

6. 添加 UserDao 的单元测试类
（1）创建 UserDaoTest.java 的框架

```
public class UserDaoTest {
```

```
    private IUserDao userDao;

    @Before
    public void setUp()  throws Exception {
        userDao = new UserDao();
    }
    @Test
    public void testLoad()  throws Exception {
    }
    @Test
    public void testAdd() throws Exception {
    }
    @After
    public void tearDown()  throws Exception{
    }
}
```

（2）引入 EntitiesHelper.java
（3）编写完善 UserDaoTest.java 类

（4）创建数据库表，注意 Id 字段要设置成 AI
（5）在配置文件中设置正确的数据库连接信息
（6）执行 mvn clean test
（7）此时会抛出异常

```
Tests in error:
  edu.zju.cst.myproject.dao.UserDaoTest: java.sql.SQLException: No suitable
driver found for jdbc:mysql://10.82.59.88:3306/maven
```

需要在 pom 里配置数据库连接依赖包：

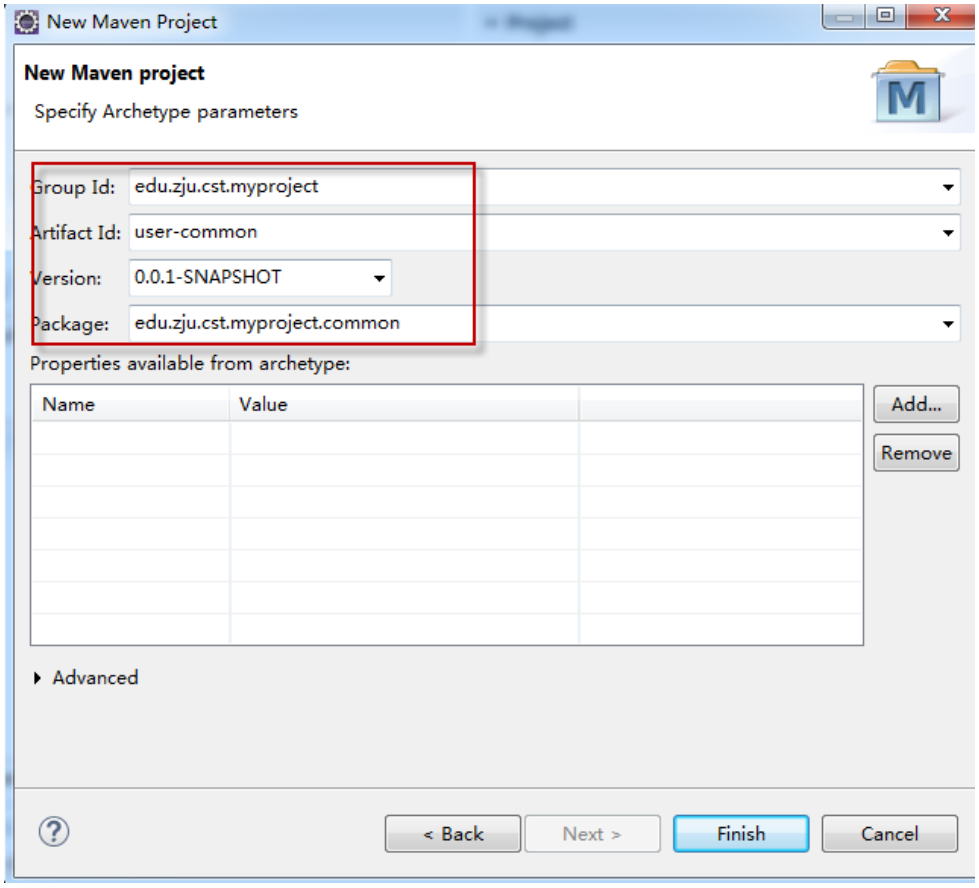
```
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.24</version>
        </dependency>
```

（8）再运行 mvn clean test 就可以成功了
如果单元测试方法出现问题，可以查看 target 下面的 log

二、开发 Service 层

User-service 项目里

1. 新建 user-common 项目

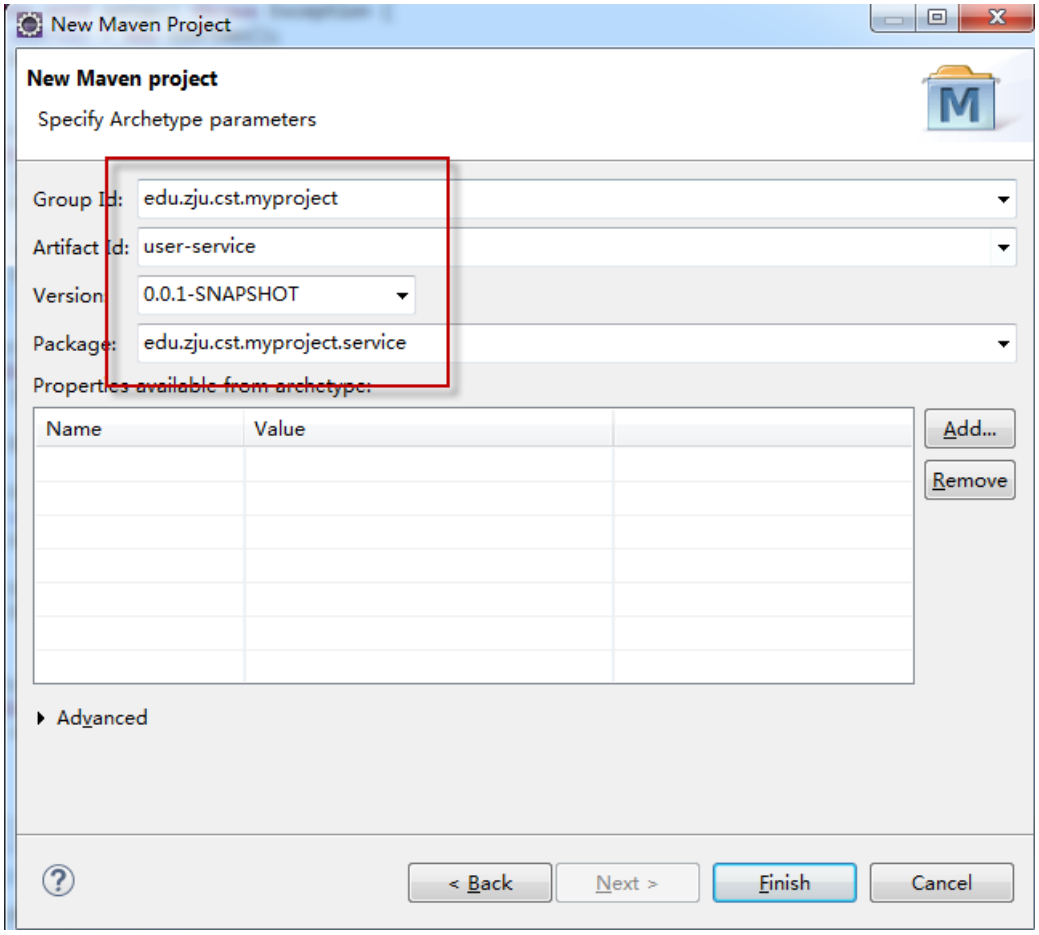


2. 在 pom.xml 里加上必要的依赖，比如：

```
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.10</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>commons-logging</groupId>
        <artifactId>commons-logging</artifactId>
        <version>1.1.1</version>
    </dependency>

    <dependency>
        <groupId>log4j</groupId>
        <artifactId>log4j</artifactId>
        <version>1.2.8</version>
    </dependency>
</dependencies>
```

3. 新建 user-service 项目



4. 在 pom 文件里添加对 user-dao、user-common

```
<dependency>
    <groupId>${project.groupId}</groupId>
    <artifactId>user-common</artifactId>
    <version>0.0.1-SNAPSHOT</version>
</dependency>

<dependency>
    <groupId>${project.groupId}</groupId>
    <artifactId>user-dao</artifactId>
    <version>0.0.1-SNAPSHOT</version>
</dependency>
```

5. 介绍 maven 隐式变量

- ${basedir} 项目根目录
- ${project.build.directory} 构建目录，缺省为 target
- ${project.build.outputDirectory} 构建过程输出目录，缺省为 target/classes

- ${project.build.finalName} 产出物名称，缺省为 ${project.artifactId}-${project.version}
- ${project.packaging} 打包类型，缺省为 jar
- ${project.xxx} 当前 pom 文件的任意节点的内容

6. 观察 Maven Dependencies，发现 dao、common 两个 project 已被加进来，并且这两个 project 的依赖包也被带过来了。

7. 开发 IUserService, UserService

```
public interface IUserService {

    public void add(User user);
    public User getUserByName(String userName);
}

public class UserService implements IUserService {
    private IUserDao userDao;
    public UserService(IUserDao userDao) {
        super();
        this.userDao = userDao;
    }
    public void add(User user) {
        userDao.add(user);
    }
    public User getUserByName(String userName) {
        return userDao.getUserByName(userName);
    }
}
```

8. 开发 UserServiceTest.java
   开发一个 UserDAOMap.java，用于测试。

9. 执行 mvn clean test

```
[WARNING] The POM for edu.zju.cst.myproject:user-common:jar:0.0.1-SNAPSHOT is missing, no dependency information available
[WARNING] The POM for edu.zju.cst.myproject:user-dao:jar:0.0.1-SNAPSHOT is missing, no dependency information available
```

需要先对 user-common, user-dao 执行 clean install 操作

11. (optional)再执行 mvn clean test，又出现错误

```
[ERROR] COMPILATION ERROR :
[INFO] -------------------------------------------------------------
[ERROR]
```

```
\workspace_maven\user-service\src\test\java\edu\zju\cst\myproject\service\U
serServiceTest.java:[11,33] 错误: 找不到符号
[ERROR]
\workspace_maven\user-service\src\test\java\edu\zju\cst\myproject\service\U
serServiceTest.java:[44,2] 错误: 找不到符号
[INFO] 2 errors
```

原因是：user-dao 中 EntitiesHelper.java 放在 test 下面，打包的时候不会打进来。
最简单的办法是把它移动到 main 目录下

12. (optional)重新 install user-dao，并对 user-service 执行 clean test
这回就可以成功了。

三、Maven 的依赖特性

1. 常见的几种 **scope**

compile(默认：编译、打包),

provided（编译和测试会加进去，打包时不会加进去），比如对于 servlet-api.jar 由 tomcat 提供，只需要在编译的时候要自己提供。

test: 测试范围有效，编译、打包都不会使用

runtime：编译的时候不依赖，运行时才依赖，比如 mysql-connector-java-XXX

依赖范围 Scope	对于compile classpath有效	对于test classpath 有效	对于runtime classpath有效	例子
compile	Y	Y	Y	spring-core
test		Y		JUnit
provided	Y	Y		servlet-api
runtime		Y	Y	JDBC驱动实现
system	Y	Y		本地的Maven仓库职位的类库文件

2. 举例：JUnit 包的 scope 为 test，要把相关的 java 文件也放到 test 目录下

如果 main 目录下有 java 文件用到了 JUnit 的类，则会编译通不过。

```
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:2.
3.2:compile (default-compile) on project user-service: Compilation failure
[ERROR] \workspace_maven\user-service\src\main\java\edu\zju\cst\myproject\servic
e\UserService.java:[5,16] 错误: 程序包org.junit不存在
```

3. 依赖范围影响传递性依赖

	compile	test	provided	runtime
compile	compile	/	/	runtime
test	test	/	/	test
provided	provided	/	provided	provided
runtime	runtime	/	/	runtime

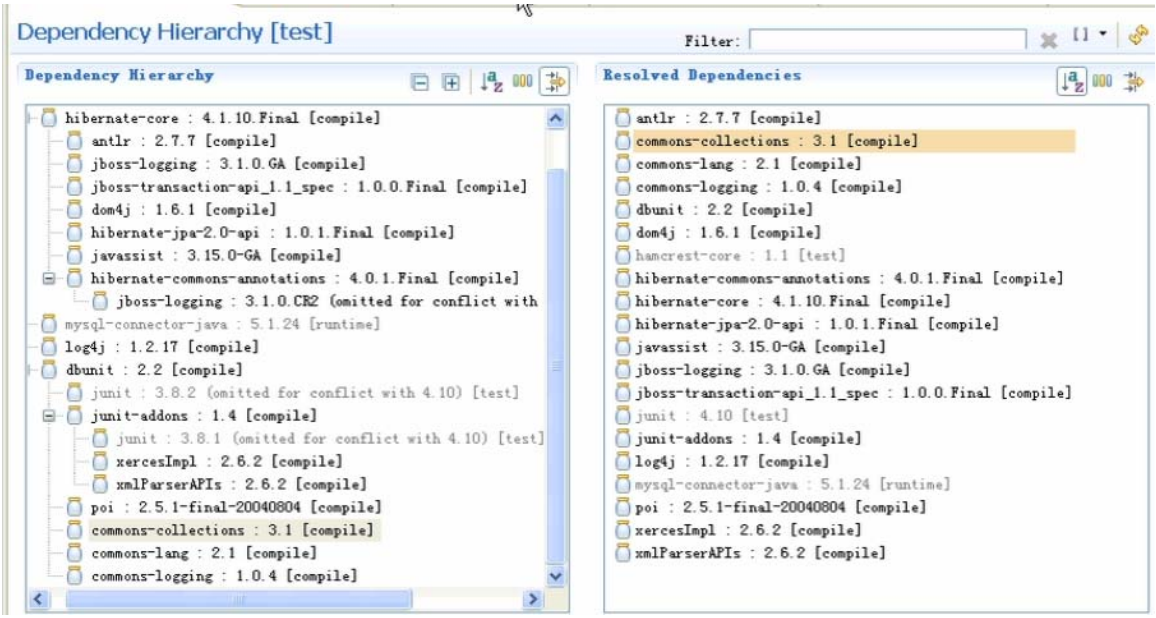
假设 A 依赖于 B，B 依赖于 C，则 A 对于 B 是第一直接依赖，B 对于 C 是第二直接依赖，A 对于 C 是传递性依赖。第一直接依赖和第二直接依赖决定了传递性依赖的范围。

如上表所示，最左边一列表示第一直接依赖范围，最上面一行表示第二直接依赖范围，中间的交叉单元格则表示传递性依赖范围。

举例 1： A 依赖 B，scope 为 test，B 依赖 C，scope 为 compile，则 A 对 C 存在 scope 为 test 的依赖。实际意义：对 A 执行 mvn test 时编译和运行会用到 C。

举例 2：A 依赖 B，scope 为 compile，B 依赖 C，scope 为 test，则 A 对 C 不存在依赖。实际意义：不需要 C，A 就可以编译和运行。

4. 在 eclipse 打开 pom.xml，可以查看 dependency hierarchy 视图



（1）

A→ L1.0 直接依赖

B-> L2.0

C→ A, B 间接依赖

则 C 会用 L1.0 （级别相同，按照声明的顺序）

举例：使用 commons-logging 的例子练习

（2）直接依赖优先于间接依赖

举例：使用 commons-logging 的例子练习

5. 排除一些依赖包

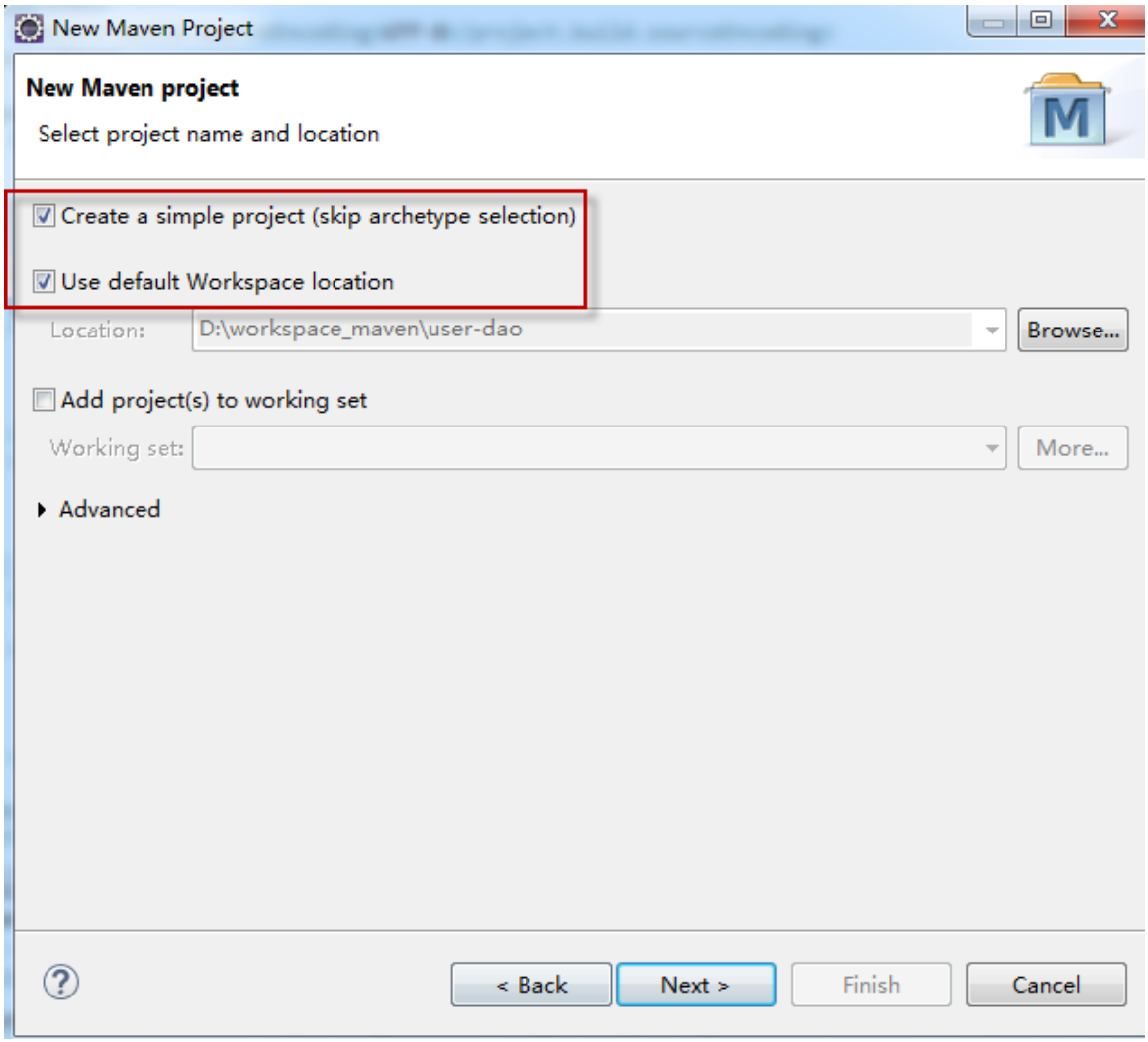
```
<dependency>
    <groupId>${project.groupId}</groupId>
    <artifactId>user-common</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <exclusions>
        <exclusion>
            <groupId>commons-logging</groupId>
            <artifactId>commons-logging</artifactId>
        </exclusion>
    </exclusions>
    </dependency>
```

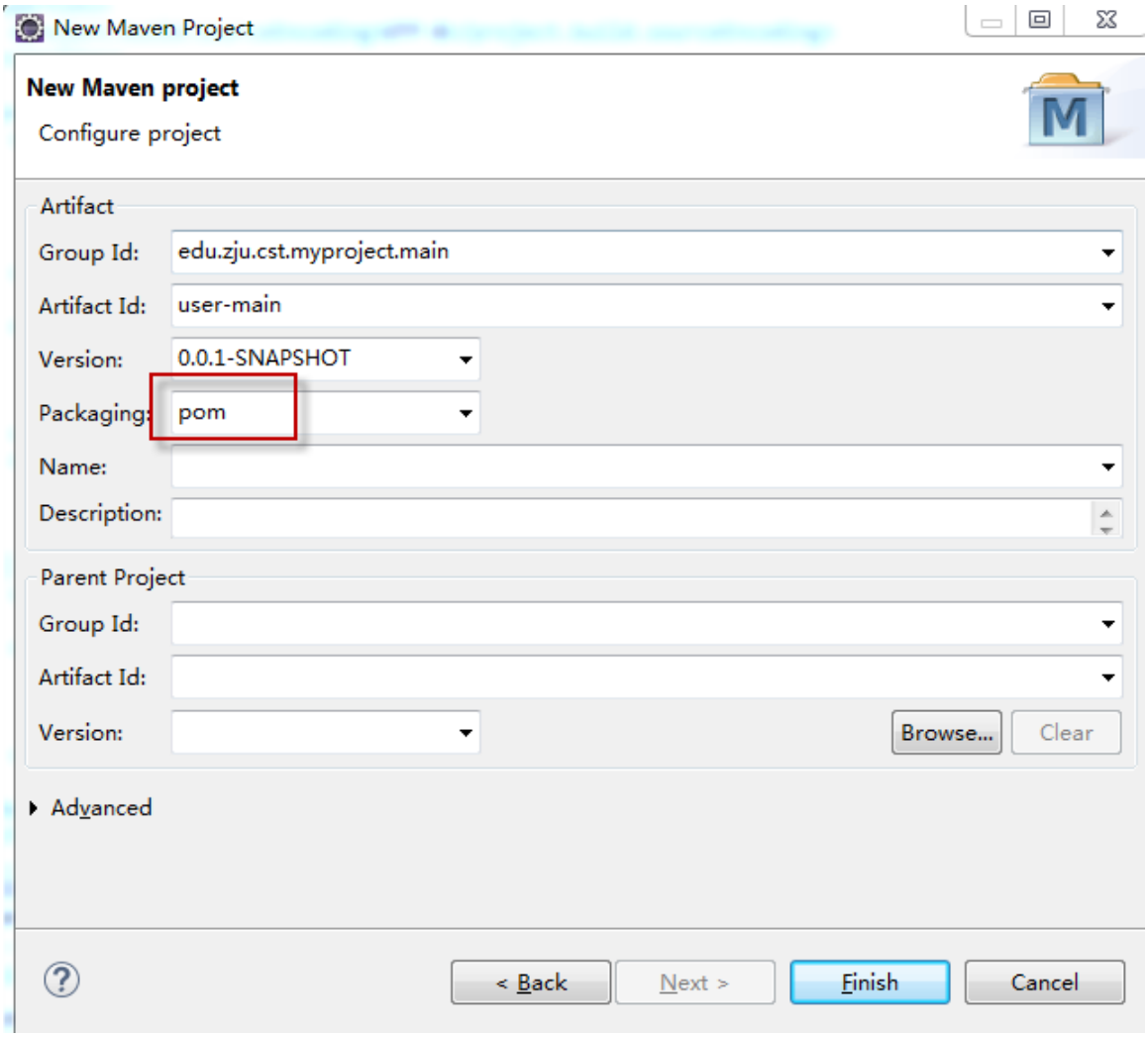
四、Maven 的聚合和继承

1. 问题：项目里有多个模块，编译、运行需要各个模块分别运行，存在重复性工作

一般操作方法:. 在各个项目模块的外层目录放一个 pom.xml，用来**聚合**各个模块的 pom.xml

2. 在 eclipse 中无法看到外层目录的 pom.xml
办法：创建一个单独的 project，命名比如 user-main





3. 编写 pom.xml

```
<modules>
    <module>../user-dao</module>
    <module>../user-common</module>
    <module>../user-service</module>
</modules>
```

4. 在 user-main 里执行 mvn clean 等命令
会发现对包含的 project 都会产生效果

5. 继承：去除各个 pom 文件里的冗余信息
也是新建一个 project，比如 user-parent
把各个 project 共同的信息移到 user-parent 的 pom 文件，比如：

```
<url>http://maven.apache.org</url>
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
```

6. 修改各个 project 的里 pom 文件

```
<modelVersion>4.0.0</modelVersion>

<parent>
    <groupId>edu.zju.cst.myproject</groupId>
    <artifactId>user-parent</artifactId>
    <version>0.0.1-SNAPSHOT</version>
</parent>

<artifactId>user-service</artifactId>
<packaging>jar</packaging>

<name>user-service</name>

<dependencies>
```

7. dependencyManagement: 简化依赖包的写法

在 user-parent 里的 pom 文件里添加：

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.3</version>
        </dependency>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>4.3.6.Final</version>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.24</version>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.8</version>
        </dependency>
    </dependencies>
</dependencyManagement>
```

此时继承了此 pom 的 project，并不会下载这里定义的 jar 包

8. 在各个 project 的 pom 里要写上用到的 jar 包，但是可以省略版本、scope 等信息

9. 继承和聚合的两个 pom.xml 合并起来
继承：pom.xml 文件的位置
聚合：模块的位置

第三部分 小结

一、复习

1. pom 项目的目录结构
2. 常用的 maven 命令
3. …

二、进一步学习资料

1. 如果对课上介绍的 Maven 基本内容还不熟悉的，可以网上找一些视频材料再看看。

2. 更深入系统的内容可以看书：许晓斌. 《Maven 实战》. 机械工业出版社. 2011-01-01