引入依赖

要添加依赖项，我们一般是先在 src 文件夹下添加 lib 文件夹，然后将你工程需要的 jar 文件复制到src/lib 文件夹下。然后在pom.xml中：

```
<dependencies>
    <!-- 在这里添加你的依赖 -->
    <dependency>
        <groupId>ldapjdk</groupId>  <!-- 库名称，也可以自定义 -->
        <artifactId>ldapjdk</artifactId>    <!--库名称，也可以自定义-->
        <version>1.0</version> <!--版本号-->
        <scope>system</scope> <!--作用域-->
        <systemPath>${basedir}\src\lib\ldapjdk.jar</systemPath> <!--项目根目录下的lib文件夹下-->
    </dependency>
</dependencies>
```

完整代码如下：

```
<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.companyname.bank</groupId>
   <artifactId>consumerBanking</artifactId>
   <packaging>jar</packaging>
   <version>1.0-SNAPSHOT</version>
   <name>consumerBanking</name>
   <url>http://maven.apache.org</url>

   <dependencies>
      <dependency>
         <groupId>junit</groupId>
         <artifactId>junit</artifactId>
         <version>3.8.1</version>
         <scope>test</scope>
      </dependency>
```

```
        <dependency>
            <groupId>ldapjdk</groupId>
            <artifactId>ldapjdk</artifactId>
            <scope>system</scope>
            <version>1.0</version>
            <systemPath>${basedir}\src\lib\ldapjdk.jar</systemPath>
        </dependency>
    </dependencies>
</project>
```

快照

快照是一种特殊的版本，指定了某个当前的开发进度的副本。不同于常规的版本，Maven 每次构建都会在远程仓库中检查新的快照。两个工程都添加

```
    <version>1.0-SNAPSHOT</version>
```

强制 maven 现在最新的快照构建。

```
mvn clean package -U
```

自动化构建

app-web-ui 和 app-desktop-ui 项目的团队要求不管依赖的 bus-core-api 项目何时变化，他们的构建过程都应当可以启动。两种方式：

- 在 bus-core-api 项目的 pom 文件中添加一个 post-build 目标操作来启动 app-web-ui 和 app-desktop-ui 项目的构建。
- 使用持续集成（CI）服务器，比如 Hudson，来自行管理构建自动化。（略）

修改 bus-core-api 项目的 pom.xml 文件。

```
<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/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <groupId>bus-core-api</groupId>
   <artifactId>bus-core-api</artifactId>
   <version>1.0-SNAPSHOT</version>
   <packaging>jar</packaging>
   <build>
   <plugins>
   <plugin>
      <artifactId>maven-invoker-plugin</artifactId>
      <version>1.6</version>
      <configuration>
         <debug>true</debug>
         <pomIncludes>
            <pomInclude>app-web-ui/pom.xml</pomInclude>
            <pomInclude>app-desktop-ui/pom.xml</pomInclude>
         </pomIncludes>
      </configuration>
      <executions>
         <execution>
            <id>build</id>
            <goals>
               <goal>run</goal>
            </goals>
         </execution>
      </executions>
   </plugin>
   </plugins>
   <build>
</project>
```

执行以下命令：

```
C:\MVN\bus-core-api>mvn clean package -U
```

Maven 将开始构建项目 bus-core-api，然后app-web-ui ， app-desktop-ui。

依赖管理

功能	功能描述
依赖调节	决定当多个手动创建的版本同时出现时，哪个依赖版本将会被使用。 如果两个依赖版本在依 时候，第一个被声明的依赖将会被使用。
依赖管理	直接的指定手动创建的某个版本被使用。例如当一个工程 C 在自己的依赖管理模块包含工程 么 A 即可指定在 B 被引用时所使用的版本。
依赖范围	包含在构建过程每个阶段的依赖。
依赖排除	任何可传递的依赖都可以通过 "exclusion" 元素被排除在外。举例说明，A 依赖 B， B 依赖 (为 "被排除的"。
依赖可选	任何可传递的依赖可以被标记为可选的，通过使用 "optional" 元素。例如：A 依赖 B， B 依 记 C 为可选的， 这样 A 就可以不再使用 C。

依赖范围

范围	描述
编译阶段	该范围表明相关依赖是只在项目的类路径下有效。默认取值。
供应阶段	该范围表明相关依赖是由运行时的 JDK 或者 网络服务器提供的。
运行阶段	该范围表明相关依赖在编译阶段不是必须的，但是在执行阶段是必须的。
测试阶段	该范围表明相关依赖只在测试编译阶段和执行阶段。
系统阶段	该范围表明你需要提供一个系统路径。
导入阶段	该范围只在依赖是一个 pom 里定义的依赖时使用。同时，当前项目的POM 文件的 部分定 代某特定的 POM。

- 公共的依赖可以使用 pom 父的概念被统一放在一起。
- Maven 通过使用可传递的依赖机制来实现。

常用的几种命令

mvn compile 编译项目源代码

mvn clean 删除 target 目录

mvn test 运行测试（运行src/test/java中的测试代码）

mvn clean package（组合命令） Maven 自动帮我们完成项目的编译、测试、打包

mvn install 效果跟mvn clean package命令一样，且项目被打包发布到了 maven 的仓库，以后其他项目需要依赖到这个项目，就可以通过在 pom.xml 文件中添加依赖来引用。

mvn archetype:generate 自动创建Maven目录结构