你要jar包，不可能每次都要联网去下载吧，多费劲，所以本地仓库就是相当于加了一层jar包缓存，先到这里来查。如果这里查不到，那么就去私服上找，如果私服也找不到，那么去中央仓库去找，找到jar后，会把jar的信息同步到私服和本地仓库中。

私服是中央仓库的缓存还会有本公司独有的包.

在实际开发中，我们经常遇到这样的场景，比如A服务依赖于B服务，A和B同时开发，B在开发中发现了BUG，修改后，将版本由1.0升级为2.0，那么A必须也跟着在POM.XML中进行版本升级。过了几天后，B又发现了问题，进行修改后升级版本发布，然后通知A进行升级...可以说这是开发过程中的版本不稳定导致了这样的问题。

Maven，已经替我们想好了解决方案，就是使用Snapshot版本，在开发过程中B发布的版本标志为Snapshot版本，A进行依赖的时候选择Snapshot版本，那么每次B发布的话，会在私服仓库中，形成带有时间戳的Snapshot版本，而A构建的时候会自动下载B最新时间戳的Snapshot版本！

Mysql的连接引擎就是运行期需要的,编译器用的不是他,用的是jdbc.

Servlet-api就是编译器需要的,运行期用的tomcat的包.

Maven的包的范围

compile：默认的scope，运行期有效，需要打入包中。

Compile是正常的,编译期有效的不是这个

provided：编译期有效，运行期不需要提供，不会打入包中。

runtime：编译不需要，在运行期有效，需要导入包中。（接口与实现分离）

test：测试需要，不会打入包中。

假如有Maven项目A，项目B依赖A，项目C依赖B。那么我们可以说 C依赖A。也就是说，依赖的关系为：C—>B—>A。

    那么我们执行项目C时，会自动把B、A都下载导入到C项目的jar包文件夹中。

    这就是依赖的传递性。

    二：依赖传递的排除

    如上，C—>B—>A。加入现在不想执行C时把A下载进来，那么我们可以用 <exclusions>标签。

[IMG_257](https://www.cnblogs.com/ygj0930/p/javascript:void(0);" \o "复制代码)

<dependencies>

<dependency>

<groupId>B</groupId>

<artifactId>B</artifactId>

<version>0.0.1</version>

<exclusions>

<exclusion>

<!--被排除的依赖包坐标-->

<groupId>A</groupId>

<artifactId>A</artifactId>

<version>0.0.1</version>

</exclusion>

</exclusions>

</dependency></dependencies>

[IMG_257](https://www.cnblogs.com/ygj0930/p/javascript:void(0);" \o "复制代码)

    三：依赖冲突与解决

    依赖冲突：一个项目A，通过不同依赖传递路径依赖于X，若在不同路径下传递过来的X版本不同，那么A应该导入哪个版本的X包呢？

    冲突解决方案：

    1：如果依赖路径的长度不同，则“短路优先”：

         A—>B—>C—>D—>E—>X(version 0.0.1)

         A—>F—>X(version 0.0.2)

         则A依赖于X(version 0.0.2)。

     2：依赖路径长度相同情况下，则“先声明优先”：

         A—>E—>X(version 0.0.1)

         A—>F—>X(version 0.0.2)

         则在项目A的<depencies></depencies>中，E、F那个在先则A依赖哪条路径的X。