Day53笔记

一、Maven

1.1 现有问题

- 随着学习的深入，项目的增多，我们使用的jar包越来越多
  - 需要自己手动查找
  - 需要手动导入
- 每个项目都需要一份jar包
- jar难找
- 重复过多

1.2 解决问题

- 使用maven能解决这些问题
  - jar管理工具，能执行jar包的增删
- 还能进行一些项目生命周期的操作

二、下载和安装（掌握）

2.1 下载

> https://maven.apache.org/download.cgi

2.2 安装

- 解压到合适的文件夹即可

2.3 程序目录

```
1、bin
2、boot
3、conf
4、lib
```

2.4 配置环境变量

- MAVEN_HOME
  - 在系统变量中定义
  - 值是到maven的安装根目录
- 配置到系统path中

三、maven配置（掌握）

3.1 配置maven默认的jdk版本

> 1、maven安装目录=》conf==》settings.xml
>
> 2、在profiles中添加profile节点
>
> 3、激活节点

```
<profiles>
    <!-- 在已有的profiles标签中添加profile标签 -->
    <profile>
        <id>myjdk</id>
        <activation>
            <activeByDefault>true</activeByDefault>
            <jdk>1.8</jdk>
        </activation>
        <properties>
            <maven.compiler.source>1.8</maven.compiler.source>
            <maven.compiler.target>1.8</maven.compiler.target>

<maven.compiler.compilerVersion>1.8</maven.compiler.compilerVersion>
        </properties>
    </profile>
</profiles>
<!-- 让增加的 profile生效 -->
<activeProfiles>
    <activeProfile>myjdk</activeProfile>
</activeProfiles>
```

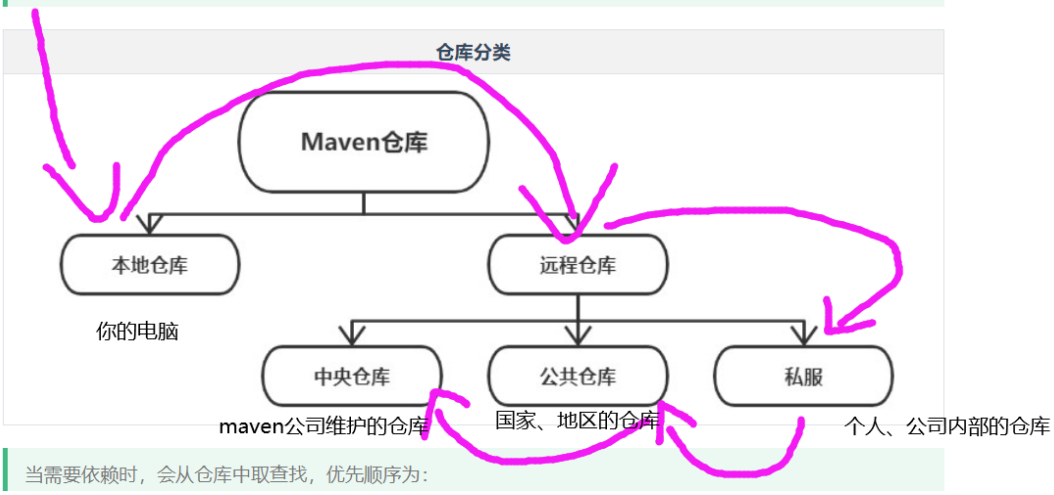
四、Maven仓库

4.1 概述

- 存放jar包的一个文件夹

4.2 仓库分类

> 仓库分类如下：



> 当需要依赖时，会从仓库中取查找，优先顺序为：

4.3 本地仓库

- 你的电脑中存储jar包的一个文件夹
- 优先级最高

4.4 远程仓库--中央仓库

- Maven社区维护的仓库
- 有大部分Java常用的依赖
- 服务器部署在国外，访问速度慢
- https://www.mvnrepository.com/

4.5 远程仓库--公共仓库（重点）

- 除中央仓库之外，还有其他远程仓库。
  - 比如aliyun仓库（http://maven.aliyun.com/nexus/content/groups/public/）
- 中央仓库在国外，下载依赖速度过慢，所以都会配置一个国内的公共仓库替代中央仓库。

```
<!--setting.xml中添加如下配置-->
<mirrors>
    <mirror>
        <id>aliyun</id>
        <!-- 中心仓库的 mirror(镜像) -->
        <mirrorOf>central</mirrorOf>
        <name>Nexus aliyun</name>
        <!-- aliyun仓库地址 以后所有要指向中心仓库的请求，都会指向aliyun仓库-->
        <url>http://maven.aliyun.com/nexus/content/groups/public</url>

    </mirror>
</mirrors>
```

4.6 远程仓库--私服（熟悉）

- 公司内部搭建的maven服务器
- 了解

五、创建Maven项目（重点）

5.1 创建项目

> idea》maven》设置GAV===》选择项目位置

5.2 IDEA关联maven

> settings=》maven=》安装目录=》配置文件=》仓库地址

5.3 导入依赖（掌握）

> 打开prom配置文件
>
> 创建dependencies节点
>
> 在dependencies节点中添加

```
<dependency>
 <groupId>mysql</groupId>
 <artifactId>mysql-connector-java</artifactId>
 <version>5.1.47</version>
</dependency>
坐标查询
https://mvnrepository.com/
```

导入改变

六、maven项目运行web项目（掌握）

6.1 构建web目录

- 创建webapp
  - 位置在main中
- 创建WEB-INF
- 创建配置文件
  - web.xml

七、maven中jar包的生命周期（熟悉）

- 在项目运行的不同状态中jar包是否参与

标识	周期
compile	缺省值，适用于所有阶段（测试运行，编译，运行，打包）
provided	类似compile，期望JDK、容器或使用者会提供这个依赖。如servlet-api.jar；适用于（测试运行，编译）阶段
runtime	只在运行时使用，如 mysql的驱动jar，适用于（运行，测试运行）阶段
test	只在测试时使用，适用于（编译，测试运行）阶段，如 junit.jar
system	Maven不会在仓库中查找对应依赖，在本地磁盘目录中查找；适用于（编译，测试运行，运行）阶段

八、Maven指令（掌握）

8.1 概述

- 通过指令完成对项目进行的一些操作
  - 打包
  - 清空编译的内容
  - 编译
  - 部署
  - ...

8.2 指令使用方式

- 使用命令提示符

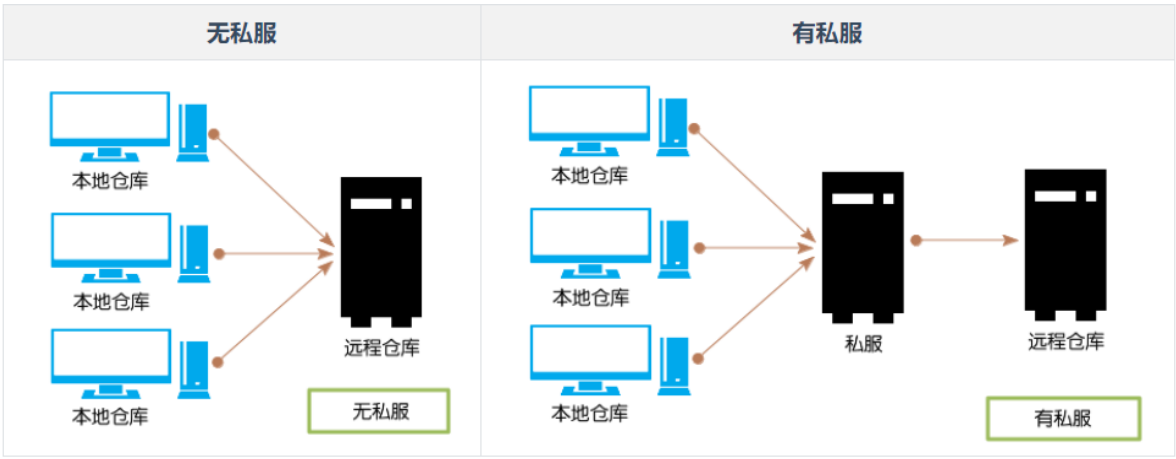
    - Windows自带的命令提示符
    - idea窗口中命令提示符
    - 运行mvn 指令
- 使用maven的可视化工具
    - lIfeCycle中有可以双击运行的一些指令

九、私服

9.1 概述

- 私服可以理解从在公司内部创建的一个maven仓库，只供本公司内部使用
- 存在的意义
    - 如果私有有就可以直接下载到本地仓，提高下载速度
    - 公司内部的框架

9.2 私服结构



9.3 下载和安装私服（熟悉）

- 下载地址

> https://help.sonatype.com/repomanager2/download

- 安装
    - 解压到合适的文件夹中
    - 安装目录中不要出现非法字符
- 常用指令

> 解压后在bin目录中执行：
>
> - nexus install 在系统中安装nexus服务
> - nexus uninstall 卸载nexus服务
> - nexus start 启动服务
> - nexus stop 停止服务

9.4 登录私服（熟悉）

> http://localhost:8081/nexus/
>
> 管理员账号和密码
>
> admin/admin123

9.5 仓库组（熟悉）

- 把多个仓库管理成一个group形成一个仓库组
- 仓库组中可以使用多个仓库中的内容
- 设置中央仓库在第一位
    - 设置中央仓库的远程地址--更改为阿里云的地址

9.6 私服关联maven（掌握）

- 在maven的设置settings.xml文件中操作
- server
- profile
- 激活

```
<servers>
    <server>
        <id>nexus-public</id> <!-- nexus的认证id -->
        <username>admin</username> <!--nexus中的用户名密码-->
        <password>admin123</password>
    </server>
</servers>
<profiles>
    <profile>
        <id>nexus</id>
        <repositories>
            <repository>
                <id>nexus-public</id> <!--nexus认证id 【此处的repository的id要和 <server>的id保持一致】-->
                <!--name随便-->
                <name>Nexus Release Snapshot Repository</name>
                <!--地址是nexus中仓库组对应的地址-->

<url>http://localhost:8081/nexus/content/groups/public/</url>
                <releases><enabled>true</enabled></releases>
                <snapshots><enabled>true</enabled></snapshots>
            </repository>
        </repositories>
        <pluginRepositories> <!--插件仓库地址，各节点的含义和上面是一样的-->
            <pluginRepository>
                <id>nexus-public</id> <!--nexus认证id 【此处的repository的id要和 <server>的id保持一致】-->
                <!--地址是nexus中仓库组对应的地址-->

<url>http://localhost:8081/nexus/content/groups/public/</url>
                <releases><enabled>true</enabled></releases>
                <snapshots><enabled>true</enabled></snapshots>
            </pluginRepository>
        </pluginRepositories>
    </profile>
</profiles>
<activeProfiles>
    <activeProfile>yourjdk</activeProfile>
    <!-- 使私服配置生效 -->
    <activeProfile>nexus</activeProfile>
</activeProfiles>
```

9.7 把自己的项目发布到私服（熟悉）

- 自己封装的工具类、框架、对框架的二次封装可以部署到公司内部公开的位置
- 在idea中设置

```
    ...
    <dependencies>
        .....
    </dependencies>

    <!-- 在项目的pom.xml中 配置私服的仓库地址，可以将项目打jar包部署到私服 -->
    <distributionManagement>
        <repository>
            <id>nexus-public</id> <!-- nexus认证id -->

<url>http://localhost:8081/nexus/content/repositories/releases</url>
        </repository>
        <snapshotRepository>
            <id>nexus-public</id> <!-- nexus认证id -->

<url>http://localhost:8081/nexus/content/repositories/snapshots</url>
        </snapshotRepository>
    </distributionManagement>
</project>
```

> 注意：
>
> 如上的 repository的 id 依然是要和settings.xml中配置的server中的id 一致，才能通过私服的认证
>
> 地址localhost以后是需要修改的