Maven学习笔记：

maven-dependency-plugin最大的用途是帮助分析项目依赖，

dependency:list能够列出项目最终解析到的依赖列表，

dependency:tree能进一步的描绘项目依赖树，

dependency:analyze可以告诉你项目依赖潜在的问题，如果你有直接使用到的却未声明的依赖，该目标就会发出警告。

maven-dependency-plugin还有很多目标帮助你操作依赖文件，例如dependency:copy-dependencies能将项目依赖从本地Maven仓库复制到某个特定的文件夹下面。

你会发现简单地用dependency:tree往往并不能查看到所有的传递依赖。不过如果你真的想要看所有的，必须得加一个-Dverbose参数，这时就必定是最全的了。  
全是全了，但显示出来的东西太多，头晕目眩，有没有好法呢？当然有了，加上Dincludes或者Dexcludes说出你喜欢或讨厌，dependency:tree就会帮你过滤出来：  
引用  
Dincludes=org.springframework:spring-tx  
过滤串使用groupId:artifactId:version的方式进行过滤，可以不写全啦，如：  
  
mvn dependency:tree -Dverbose -Dincludes=asm:asm    
  
就会出来asm依赖包的分析信息：

omitted for duplicate

递归依赖的关系列的算是比较清楚了，每行都是一个jar包，根据缩进可以看到依赖的关系。

* 最后写着compile的就是编译成功的。
* 最后写着omitted for duplicate的就是有jar包被重复依赖了，但是jar包的版本是一样的。
* 最后写着omitted for conflict with xxxx的，说明和别的jar包版本冲突了，而该行的jar包不会被引入。比如上面有一行最后写着omitted for conflict with 1.8.3，那么该行的commons-beanutils:jar:1.8.0不会被引入，只有1.8.3版本的会被引入。

Spring

权限修饰符default private protected

* Private 作用于属性和方法时就只能在同一个类中能访问他们，同其任何地方都访问不到，比如子类，同包下子类，不同包下子类，都不能访问。
* Default 当不加权限修饰符时，默认就是default，作用于方法和属性时，可以在同一类中，同一包下类中（包括子类和任意其他类），注意：其他包中的类（person的子类或或者其他任意类）中访问不到default修饰的方法和属性

只有Default和public能修饰类和接口，defaul修饰类或接口时，只能在同一包中的类或接口中访问到，其他包中类访问不到包中的default类。Public 修饰类或接口，能被任意位置类中访问到。

*Spring boot*

*@Configuration使用，主要用于替换xml配置spring的方式*

功能描述: 被注解的类内部包含有一个或多个被@Bean注解的方法，这些方法将会被AnnotationConfigApplicationContextAnnotationConfigWebApplicationContext类进行扫描，并用于构建bean定义，初始化Spring容器。

链接: <https://www.cnblogs.com/duanxz/p/7493276.html>

在写工具类，静态方法时：返回值尽量为booolean，并且在返回值时，无论成功或失败都打印语句，出现异常·· 便于维护/。排除出现特殊情况比如，该工具类包含http请求需要接受返回值，对返回值有操作

创建文件是，如果路径是全部由拼接的可以用Fil.separator拼接，或者写public final static String separator = “/”;用这种方式去拼接路径，

如果路径一部分是已经拼接好的，另一部分需要拼接，则使用public final static String separator = “/”;去拼接 这样可以避免出现在linux和windows下路径格式问题

也可以使用File的成员方法getCanonicalPath();去生成规范路径名字符串

If 和if else 的用法

① if ,true 执行代码块，false跳过代码块

② else ,true 执行代码块一，false执行代码块二，

③ if – else if ,if 为true执行代码块一，else – if为true执行代码块二 ，false

Spring boot

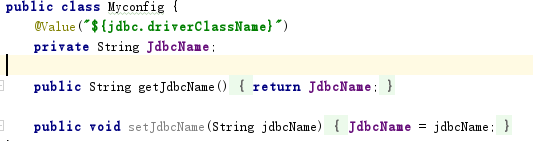
@propertySource(“”)

@Configuration

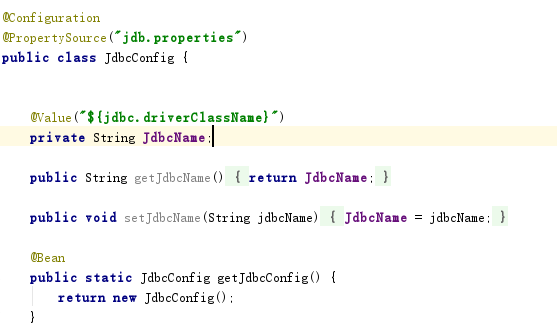
加上@Value(“${}”)和@Bean注解读取\*.properties配置文件可以完成一个配置类的配置

使用方法主要有两种

方式一：除了配置类，需要单独协议类Bean来存配置文件信息，Bean不用任何注解。除此之外配置类需@Bean返回一个Bean对象，在Controller中需要注入改bean



方式二：只需要写一个配置类,用@Bean注解并将创建一个配置类对象，需要在Bontroller中注入改配置类



@import 配合上面的配置可以导入配置类

@Bean 只能写在方法上，表明使用此方法创建一个对象，并放入spri

Java中的基类和派生类

派生类继承自基类（必须是extends）

从外部来看，派生类是一个与基类具有相同接口的新类，或许还有一些额外的方法和域。但继承并不仅仅是类的复用。当创建了一个派生类的对象时，该类包含（依赖）一个基类对象。这个子对象和用基类直接创建的对象没有什么不同。两者的区别在于，前者包含的这个子对象来自于派生类内部，后者来自于外部。对于派生类的子对象初始化时其至关重要的作用是，派生类的构造器调用了基类的构造器，而基类的构造器具有执行基类初始化所需的所有能力和知识。

需要注意的是基类必须有无参构造器（一般类中隐含了一个无参构造器，如果自定义了有参构造器则无参构造器失效，如需使用无参构造器，需要单独声明），否则在派生类的构造器中单独声明有参构造器

Java8 语法解析

<https://blog.csdn.net/io_field/article/details/54971761>

今天按了ctrl+win+c 发现电脑边黑白色的了，在按一次就会变回来