b0f70d2f675.png](https://p1-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/21ae3510ae7c49a4af811b875cccfe21~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=958&h=564&s=53462&e=png&b=fefefe)
我们使用解压工具打开`rt.jar`可以清晰的看到，里面包括的内容很全，基本上包括了Java的方方面面，都给你加载进去了。就算你只写一个HelloWord，它也给你加载了Applet、awt等你根本不会用到的东西。现在的服务器和个人电脑随便都是8G、16G以上的内存，对于`rt.jar`占用的这点内存可能没什么感觉，但在一些对于内存很敏感的领域，Java这种方式就显得不太合理。Java官方可能也意识到了这个问题，所以在Java9的时候推出了Java平台模块系统（Java Platform Module System，JPMS）

Java平台模块系统（Java Platform Module System，JPMS）

JPMS是在Java9发布的，其实从Java7开始官方就在准备JPMS，本来准备在Java8中引入JPMS的，实在是改动太大，到Java9中才正式发布。其实做过开发的都会想的到，模块化相当于要从整体上重构整个系统，结构调整巨大，对于整个系统考验是很大的，这可能也是JPMS从Java7开始准备，直到Java9才发布的原因。

前面我们说过，在Java8及之前的版本中，其实是包括JDK和JRE两个部分的。但从Java9之后，就没有专门的JRE了，因为模块化之后，自己可以选择JVM加载哪些模块，相当于按需加载就行了，你写HelloWord可能只需要一个加载`java.base`就行了。
下面是JDK11的结构

![7940167be7f54191a8e5143b5f1 的包

```
    public static void main(String[] args) {
        UserService userService = new UserService();
        userService.register();
    }
```

如上所示，在OrderModule中就可以直接引入UserService了

如果你只想将UserModule开放给GoodsModule，可以这样写

```
module UserModule {
    exports com.user to GoodsModule;
}
```

这样就算OrderModule引用了UserModel，也没办法使用UserService，因为UserModule只开放给GoodsModule了

以上就是一个使用模块化来实现模块之间相关调用的简单例子，当然实际的模块化系统不可能这么简单，此处只起到一个抛转引玉的作用，如果对模块化系统比较感兴趣可以去JDK官网了解详细的信息

模块化的好处

1、强封装性

类和包可以被模块化，只有模块之间声明的接口是对外可见的，提高了代码的封装性，减少了不必要的耦合

2、明确的依赖管理

块之间通过requires声明依赖，明确指出了哪些模块需要哪些其他模块，避免了隐式依赖和类路径冲突

3、运行时性

JVM可以仅加载运行应用程序所需的模块，减少了内存占用，提高了启动速度和运行时性能

4、安全性和隔离

模块边界强化了安全性，限制了代码的访问权限，降低了攻击面

通过模块化，Java开发者可以构建更加健壮、高效且易于维护的大型应用程序

你的项目有用到模块化吗？欢迎给我留言讨论！

原文链接: https://juejin.cn/post/7363828811362238515