没有maven的原始社会中，我们程序员的日常工作流程如下。。。。。。853769620

1、每天都要编写自己负责的功能模块的代码，然后编写对应的单元测试，接着运行单元测试，生成单元测试覆盖率的报告，发送给自己的leader

2、注意，如果你的项目遇到了需要使用的第三方jar包，你需要自己去寻找对应版本的jar包，然后添加到你的工程里面去

我们的每一个工程，都是要依赖大量的其他第三方的项目的，比如最经典的，ssh框架的整合，spring mvc spring mybatis。这个时候就要把对应框架的jar包给从网上下载下来，然后copy到我们自己的工程的lib目录下面去。

很麻烦，一个工程，依赖包太多了，可能多达数十个，搞一大堆

然后就要开始尝试跑，依赖能不能放在一起整合在一起，跑通，这个时候还会出现各种缺少依赖jar包。你依赖了spring，spring可能又依赖了个什么什么包，此时直接运行还会报错，class not found exception，可能又要去下载更多的包放进来，直到不再报错。

还有的时候，可能还会出现一些依赖冲突，比如你依赖了个什么mybatis，还依赖了spring，spring和mybatis都依赖了相同的一个项目，打个比方log4j，但是版本还不一样。这个时候你还得自己手动选择一个较高的版本，解决一些依赖的冲突问题。

3、接着可能每隔几天都需要将自己开发好的代码进行编译、打包，然后在测试环境进行部署，等待对应的测试人员来测试

每天都会写一些代码，还会写一些测试，但是每隔几天，可能一块功能开发好了，这个时候就要部署到集成测试环境里面，跟其他人开发的代码集成在一起，一起测试一下，能不能跑通。

集成测试完了以后，先要编译代码，然后打包，手工部署

如果测试过程中，出现了各种bug，再次修改代码，再次手工编译，手工打包，手工部署

以此类推，直到集成测试通过

接着呢，又是QA测试，测试工程师，来测试java工程师开发出来的系统。又是手工编译，手工打包，手工部署到QA测试环境。测试的过程中，有任何的bug，又是返工，重新修改代码，测试，然后手工编译，手工打包，手工部署到QA测试环境。

部署也很坑，tomcat，每次部署，先停一下，然后弄个war包，放进去，把老的war包给干掉，新的war包放进去，tomcat启动

每次要部署到某个环境的tomcat里面的时候，还得弄个终端软件，ssh连接到那个环境的服务器上面去，在那个服务器上的tomcat里面部署。

4、再次注意，可能你一个大型的系统有很多个工程，每个工程的众多依赖包都需要你去手工管理，如果升级某个依赖，可能需要对几十个工程的jar包进行手工替换

实际上来说，对于一些大型的erp系统，一些大型的oa系统，或者大型的银行系统，电信系统

2010年以前，说实话，整个软件行业，技术还是比较low一点的

2007，2006，2008

一般都是一个单块应用，但是得拆分啊，可能是一个系统，拆分成了多个模块，每个模块是一个工程，最后会将多个工程的jar包放到一个统一的web工程里面去，让web工程去运行

每个模块对应的工程，就是一个工程师，几个工程师在玩儿，在开发代码

erp，电信系统，几十个工程，甚至上百个工程

那个年代，不堪入目，很麻烦，纯手工，效率很低下，经常出问题

5、同时注意，可能整个团队经常要对几十个工程进行繁琐的打包和部署

6、日复一日，年复一年，项目做好了，接着又是将项目中所有工程，都进行代码编译，然后构建成一个一个的发布包，然后依次将几十个工程部署到生产环境

程序员的痛苦在于。。。。。。

1、你可能需要手工运行数十个，甚至数百个，上千个的单元测试，还要想办法弄出来一份完整的单元测试覆盖率的报告

2、你可能需要手工管理多个工程中复杂的依赖jar包，包括后期的可能依赖冲突调解以及jar包版本升级

3、你可能需要手工对多个工程进行繁琐的编译、构建发布包

4、你可能需要频繁的手工部署发布包到各种环境下

大量的手工，手工，手工，各种繁琐而且重复的操作，浪费了我们大量的时间，而且很容易出错

这，就是没有maven这种工程管理工具的痛苦原始社会！