Maven的生命周期

就是maven构建项目的过程

--Maven把项目构建的各个过程对应成其生命周期的各个阶段，每一个阶段都提供相应的命令
--Maven独立使用开发项目用到的常用命令如下：
mvn clean：清理，删除原来编译和测试的目录，target目录。但是已经install到仓库中的包不会被清除
mvn compile：编译主程序（只编译src/main/java文件下的java文件），在当前目录下（项目目录）生成一个target目录，里面存放编译生成的字节码文件
mvn test-compile：编译测试程序（只编译src/test/java文件下的java文件），结果和上面一样
mvn test：测试，生成一个surefire-reports目录，保存测试结果
mvn package：打包主程序，编译、编译测试、测试、按照pom文件中的配置把主程序打包生成jar文件或war包
mvn install：安装主程序，把本工程打包，按照工程的坐标存放保存到本地仓库中
mvn deploy：部署主程序

--maven插件：
maven这些命令执行时，真正完成这些功能的是插件，即一些jar包，一些类

--maven插件类型
1）单元测试：用junit，是一个专门测试的框架（工具）
测试的内容：测试类中的方法，每一个方法都是独立测试的。方法是测试的基本单位（单元）
maven就借助单元测试批量检测类中大量的方法是否符合预期
使用步骤：
1、加入junit依赖代码到pom文件中，在中央仓库搜索junit，点击使用最多的，选择版本，其中4.11和4.12使用最多

```
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>4.12</version>
<scope>test</scope>
</dependency>
```

2、在maven项目中的src/test/java目录下，创建测试程序
推荐的创建类和方法提示：
1）、测试类的名称是Test+要测试的类名
2）、测试的方法名称，test+方法名称
--如测试之前HelloMaven的程序：
创建测试类：TestHelloMaven

```
@Test
public void testAdd(){
    --测试HelloMaven的add方法是否正确的代码
}
```

其中testAdd叫做测试方法，定义规则如下：
1）、方法必须是public修饰
2）、方法必须是没有返回值的
3）、方法名称自定义，推荐test+测试方法名（遵从驼峰式命名规则）
4）、在方法上面加上注解@Test