1、使用maven快速创建一个工程

为了加快速度，在settings.xml中加一段配置，用国内阿里云的镜像仓库去下载各种东西

<mirror>

<id>nexus-aliyun</id>

<mirrorOf>\*</mirrorOf>

<name>Nexus aliyun</name>

<url>http://maven.aliyun.com/nexus/content/groups/public</url>

</mirror>

随便找一个地方创建一个目录（F:\development\workspace），然后在那个目录中执行下面的命令，基于maven创建一个工程：

mvn archetype:generate -DgroupId=com.zhss.maven -DartifactId=maven-first-app -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

此时maven会在当前目录下，新建了一个目录，名称就是-DartifactId指定的名称。进入这个maven-first-demo目录，就可以看到maven自动给我们初始化好了一个工程对应的目录结构。

* DgroupId：src/main/java目录下后续的目录结构
* DartifactId：工程名字（总文件夹名）

2、maven的约定

这就是基本的maven工程目录结构，其中src/main/java目录包含了这个项目的java源码，src/test/java目录包含了测试代码，pom.xml文件是maven的核心配置文件，是这个项目的Project Object Model。

3、pom.xml初步介绍

pom.xml文件是一个项目最核心的maven配置文件，包含了大量的信息，maven正是基于这里的配置信息来对工程进行构建等管理工作。一个最最基本的pom.xml文件如maven生成的pom.xml所示。

<project>：pom.xml中的顶层元素

<modelVersion>：POM本身的版本号，一般很少变化

<groupId>：创建这个项目的公司或者组织，一般用公司网站后缀，比如com.company，或者cn.company，或者org.zhonghuashishan

<artifactId>：这个项目的唯一标识，一般生成的jar包名称，会是<artifactId>-<version>.<extension>这个格式，比如说myapp-1.0.jar

<packaging>：要用的打包类型，比如jar，war，等等。

<version>：这个项目的版本号

<name>：这个项目用于展示的名称，一般在生成文档的时候使用

<url>：这是这个项目的文档能下载的站点url，一般用于生成文档

<description>：用于项目的描述

3、对项目进行打包

可以认为源代码和测试代码都写好了，然后我们就要自动化运行测试用例+编译+打包，另外的话呢，这个项目有一个junit的依赖，我们期望的时maven可以自动给我们下载和管理这些依赖包

使用mvn clear package命令，对一个工程进行构建，构建出来一个可以执行的java jar包。

第一个发现，我们依赖了一个junit包，然后我们不需要自己手工去网上下载对应的jar包了，直接maven自动给我们下载了jar包，不再需要我们自己去管理依赖了，maven自动化管理依赖

第二个发现，运行单元测试、编译、打包，自动化运行了单元测试的用例，自动化把你的java源代码编译成了.class文件，自动化把我们的代码打包成了一个jar包

4、执行打好的jar包

java -cp target/maven-first-app-1.0-SNAPSHOT.jar com.zhss.maven.App

853769620