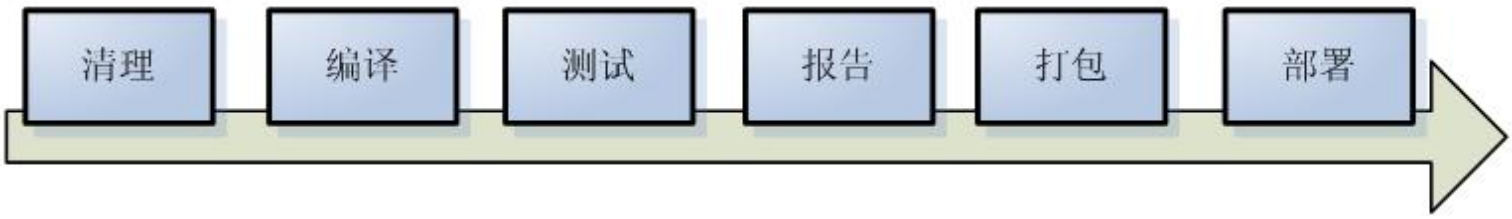
Maven学习总结(一)——Maven入门 - 孤傲苍狼

一、Maven的基本概念

Maven(翻译为"专家"，"内行")是跨平台的项目管理工具。主要服务于基于Java平台的项目构建，依赖管理和项目信息管理。

1.1、项目构建

项目构建过程包括【清理项目】→【编译项目】→【测试项目】→【生成测试报告】→【打包项目】→【部署项目】这几个步骤，这六个步骤就是一个项目的完整构建过程。



理想的项目构建是高度自动化，跨平台，可重用的组件，标准化的，使用maven就可以帮我们完成上述所说的项目构建过程。

1.2、依赖管理

依赖指的是jar包之间的相互依赖，比如我们搭建一个Struts2的开发框架时，光光有struts2-core-2.3.16.3.jar这个jar包是不行的，struts2-core-2.3.16.3.jar还依赖其它的jar包，依赖管理指的就是使用Maven来管理项目中使用到的jar包，Maven管理的方式就是“自动下载项目所需要的jar包，统一管理jar包之间的依赖关系”。

1.3、使用Maven的好处

Maven中使用约定，约定java源代码代码必须放在哪个目录下，编译好的java代码又必须放到哪个目录下，这些目录都有明确的约定。

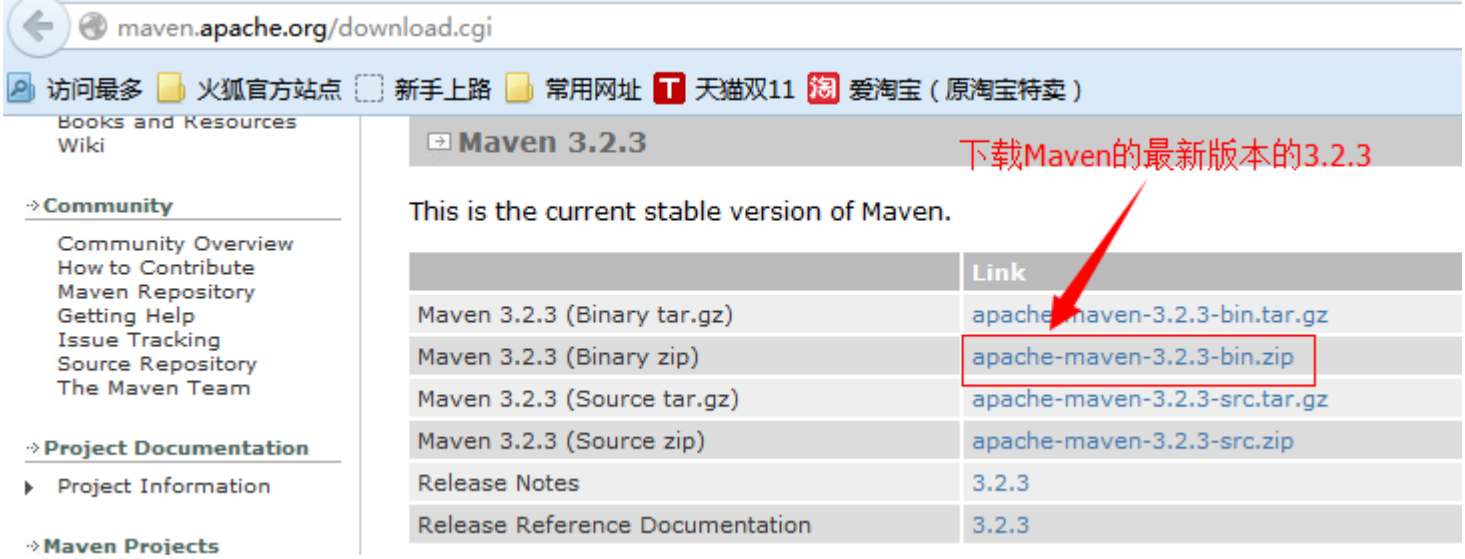
Maven的每一个动作都拥有一个生命周期，例如执行 mvn install 就可以自动执行编译，测试，打包等构建过程

只需要定义一个pom.xml,然后把源码放到默认的目录，Maven帮我们处理其他事情

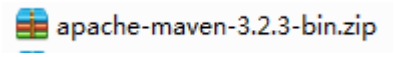
使用Maven可以进行项目高度自动化构建，依赖管理(这是使用Maven最大的好处)，仓库管理。

二、Maven下载

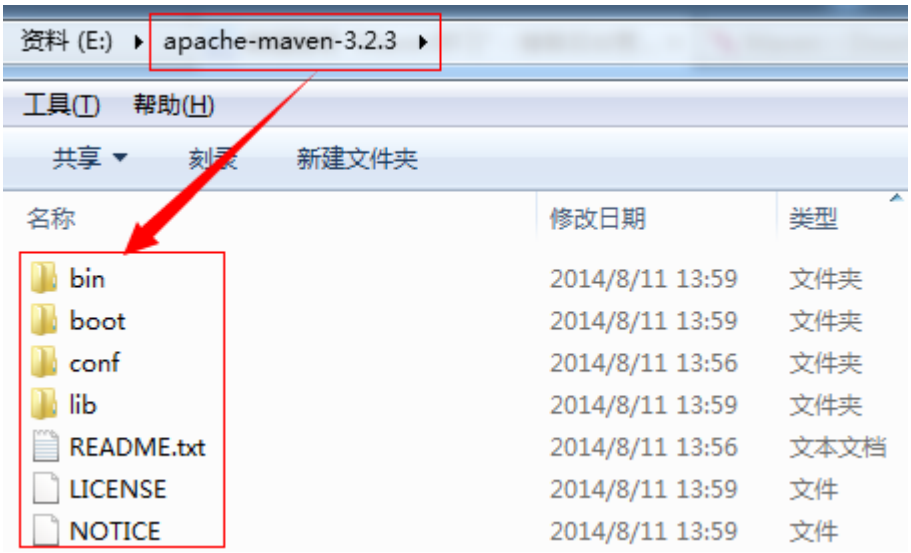
下载地址：http://maven.apache.org/download.cgi



下载完成后，得到一个压缩包



，解压，可以看到maven的组成目录



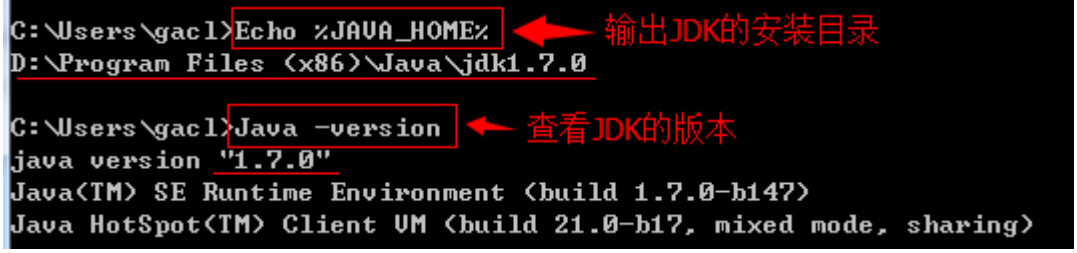
Maven目录分析

- bin：含有mvn运行的脚本
- boot：含有plexus-classworlds类加载器框架
- conf：含有settings.xml配置文件
- lib：含有Maven运行时所需要的java类库
- LICENSE.txt，NOTICE.txt，README.txt针对Maven版本，第三方软件等简要介绍

三、Maven安装

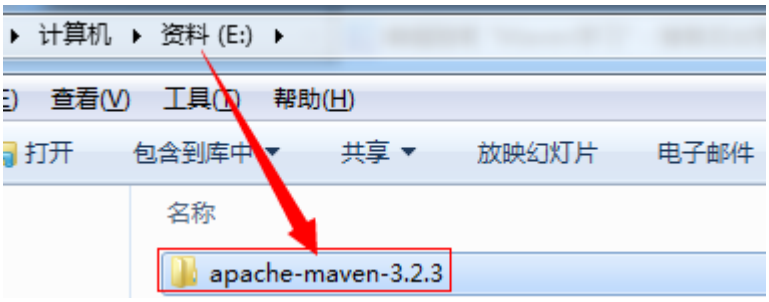
1、首先要确保电脑上已经安装了JDK(要jdk 1.6+的版本)，配置好JDK的环境变量，使用如下的两个命令检查检查JDK安装的情况。

```
1 Echo %JAVA_HOME%
2 Java -version
```

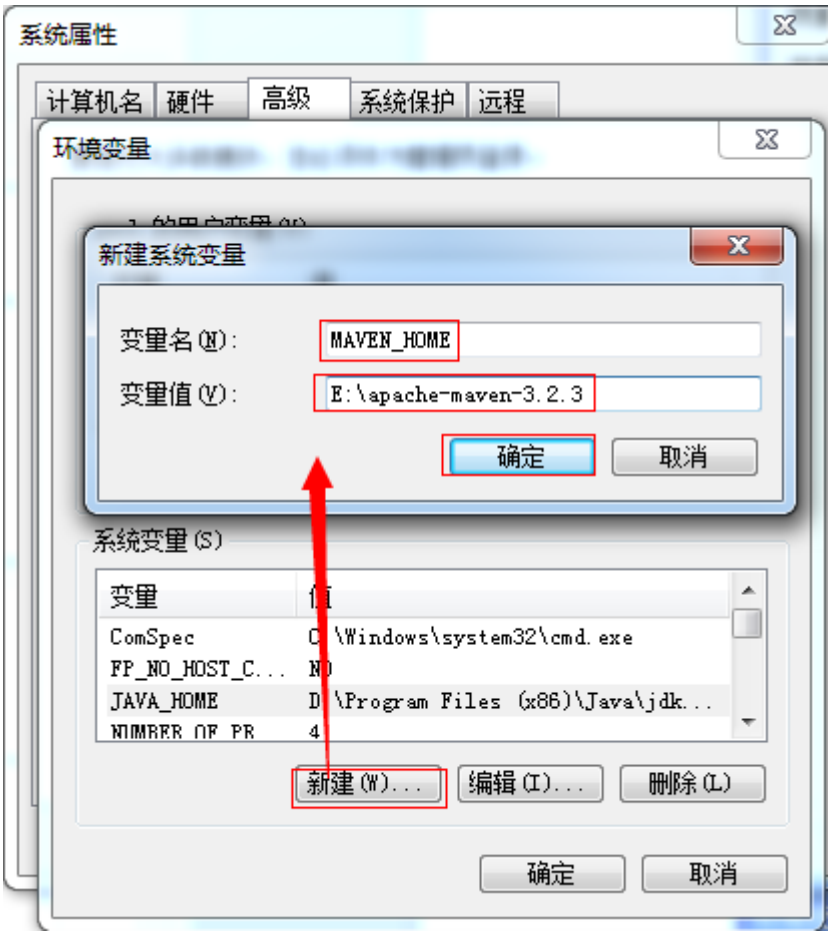


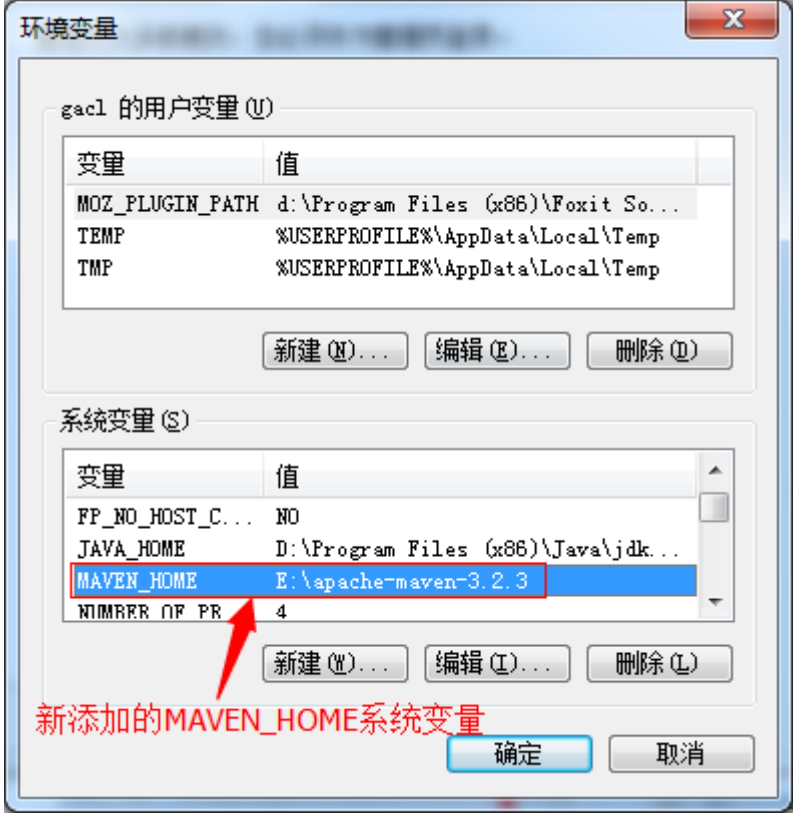
2、对apache-maven-3.2.3-bin.zip进行解压缩

对apache-maven-3.2.3-bin.zip进行解压缩，例如解压到如下目录(解压目录最好不要有中文)

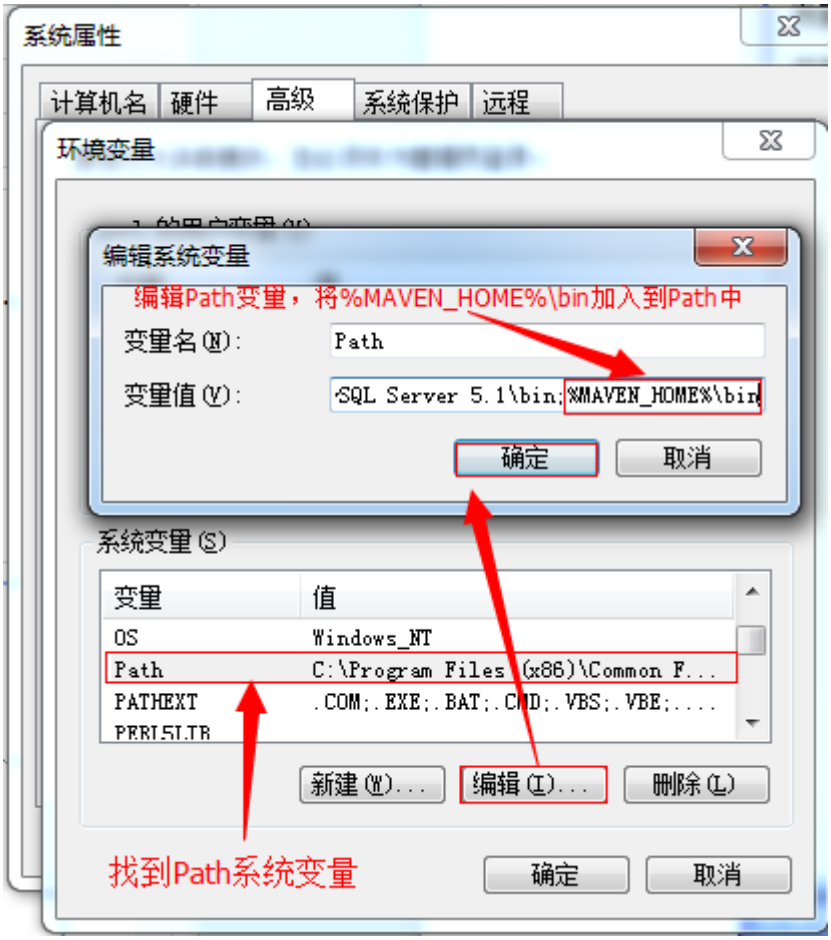


3、设置系统环境变量：MAVEN_HOME





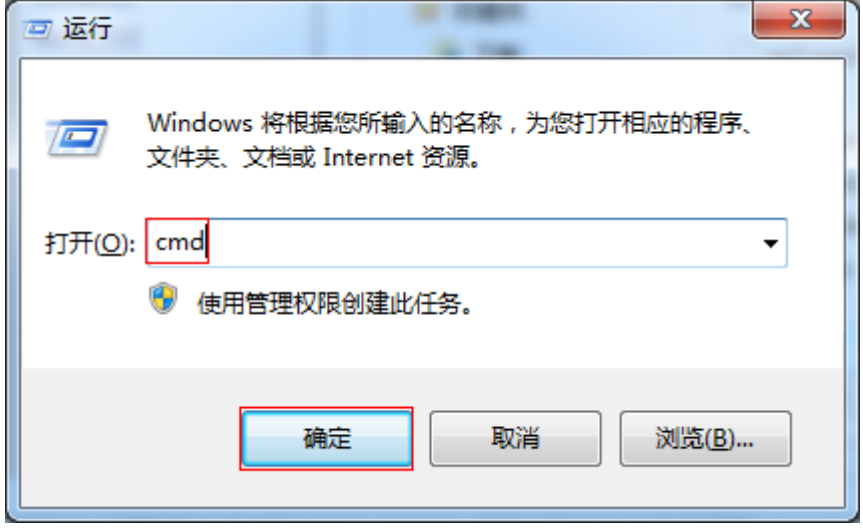
4、设置环境变量Path，将%MAVEN_HOME%\bin加入Path中，一定要注意要用分号；与其他值隔开，如下图所示：



%MAVEN_HOME%\bin代表的就是"E:\apache-maven-3.2.3\bin"目录

5、验证Maven安装是否成功

打开cmd窗口



输入"mvn　-v"命令 查看Maven的相关信息，如下图所示：

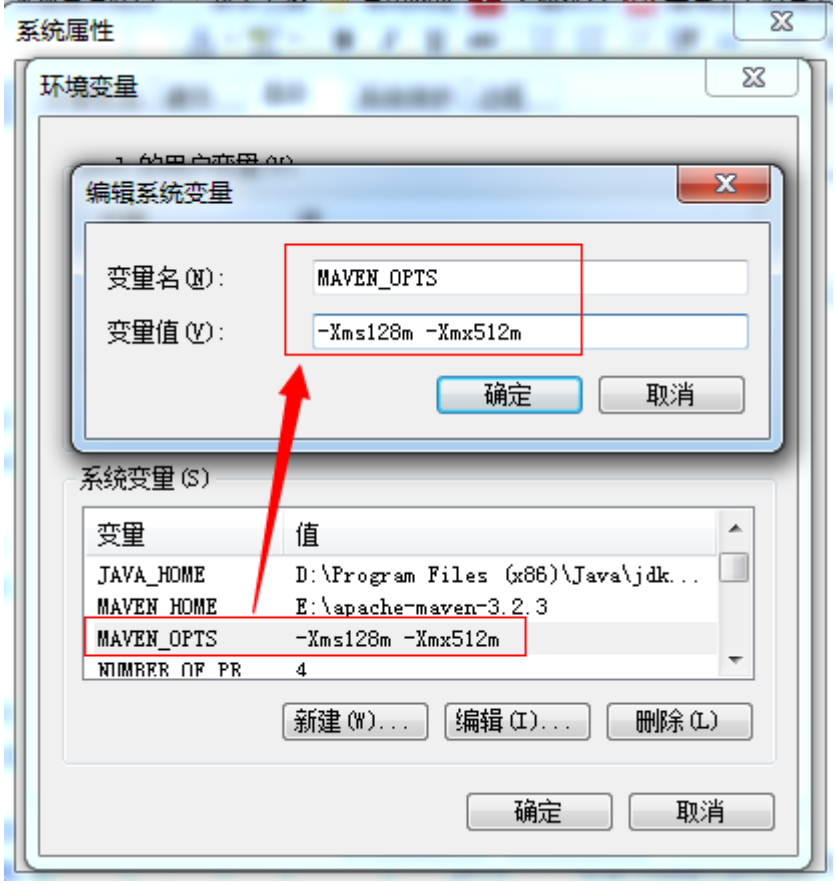
能够出现这样的信息就说明Maven的安装已经成功了。

6、设置MAVEN_OPTS环境变量(可选配置)

由于Maven命令实际上是执行了Java命令，所以可以通过JAVA命令参数的方式来设置MAVEN运行参数。MAVEN_OPTS环境变量正是用于此用途

MAVEN_OPTS

-Xms128m -Xmx512m，分别设置JVM的最小和最大内存，如下图所示：



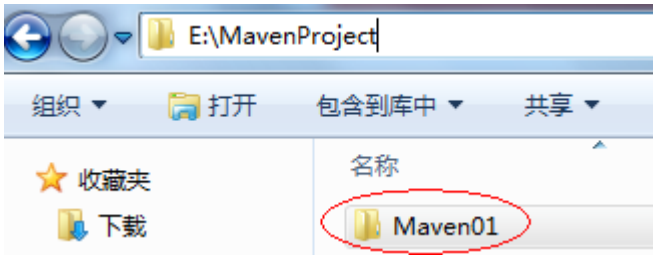
四、Maven的简单使用

4.1、Maven项目的目录约定

```
MavenProjectRoot(项目根目录)
    |----src
    |         |----main
    |         |              |----java ——存放项目的.java文件
    |         |              |----resources ——存放项目资源文件，如spring, hibernate配置文件
    |         |----test
    |         |              |----java ——存放所有测试.java文件，如JUnit测试类
    |         |              |----resources ——存放项目资源文件，如spring, hibernate配置文件
    |----target ——项目输出位置
    |----pom.xml ----用于标识该项目是一个Maven项目
```

4.2、手动创建Maven项目，使用Maven编译

1、创建项目根文件夹，例如Maven01

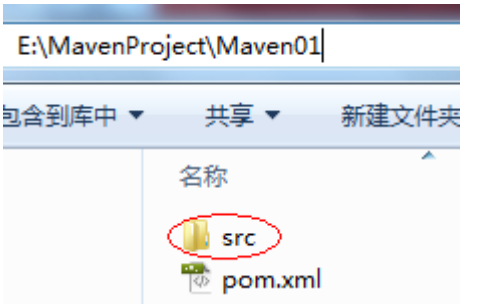


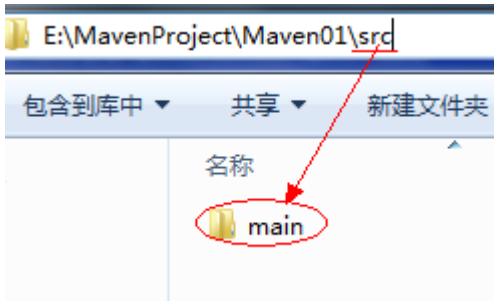
2、在Maven01文件夹中创建“pom.xml”文件，如下图所示：

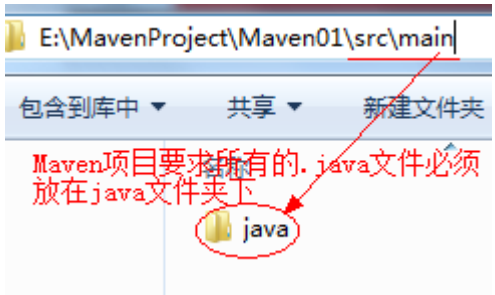
pom.xml文件中的内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <!--所有的Maven项目都必须配置这四个配置项-->
    <modelVersion>4.0.0</modelVersion>
    <!--groupId指的是项目名的项目组，默认就是包名-->
    <groupId>cn.gacl.maven.hello</groupId>
    <!--artifactId指的是项目中的某一个模块，默认命名方式是"项目名-模块名"-->
    <artifactId>hello-first</artifactId>
    <!--version指的是版本，这里使用的是Maven的快照版本-->
    <version>SNAPSHOT-0.0.1</version>
</project>
```

3.编写Java类文件，Maven项目规定，所有的*.java文件必须放在src目录下的main目录下的java目录中，在Maven01项目根目录中创建一个src目录，然后在src目录中创建main目录，在main目录中再创建java文件夹，如下图所示：





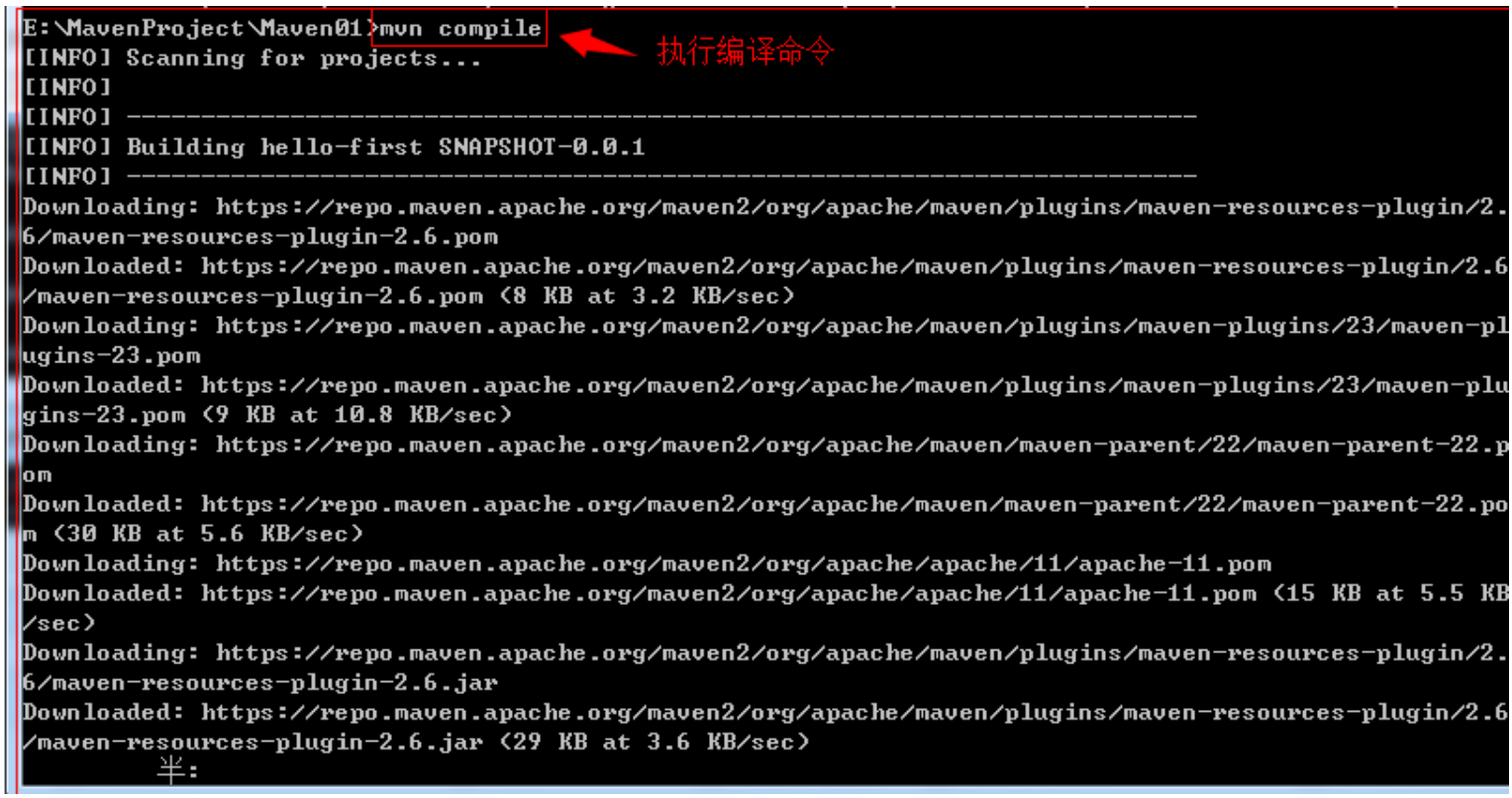


在java文件夹下创建一个Hello.java文件，如下图所示

在Hello.java文件中编写如下代码：

```
1 public class Hello{
2     public static void main(String[] args){
3         System.out.println("Hello Maven");
4     }
5 }
```

4.使用Maven编译Hello.java，首先进入到项目根目录，然后使用命令”mvn compile”进行编译，如下图所示：

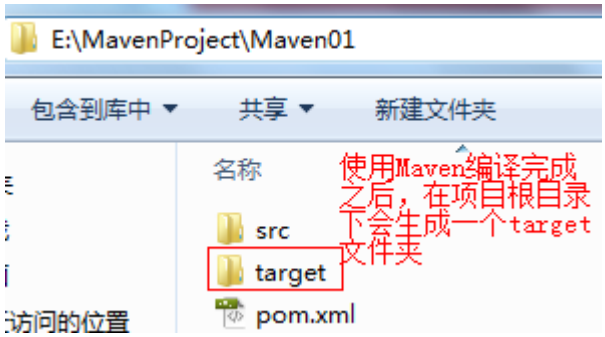


```
管理员: C:\Windows\system32\cmd.exe
s-compiler-api-1.9.1.jar
Downloading: http://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-javac/1.9.1/ple
xus-compiler-javac-1.9.1.jar
Downloading: http://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.0/plexus-utils-3
.0.jar
Downloading: http://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-manager/1.9.1/p
lexus-compiler-manager-1.9.1.jar
Downloaded: http://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-manager/1.9.1/pl
exus-compiler-manager-1.9.1.jar (5 KB at 3.0 KB/sec)
Downloaded: http://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-api/1.9.1/plexus
-compiler-api-1.9.1.jar (21 KB at 11.5 KB/sec)
Downloaded: http://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-javac/1.9.1/plex
us-compiler-javac-1.9.1.jar (14 KB at 3.4 KB/sec)
Downloaded: http://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.0/plexus-utils-3.
0.jar (221 KB at 23.0 KB/sec)
[WARNING] File encoding has not been set, using platform encoding GBK, i.e. build is platform depend
ent!
[INFO] Compiling 1 source file to E:\MavenProject\Maven01\target\classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 2:10.740s
[INFO] Finished at: Tue Sep 03 22:59:05 CST 2013
[INFO] Final Memory: 6M/15M
[INFO] ------------------------------------------------------------------------
E:\MavenProject\Maven01>
```

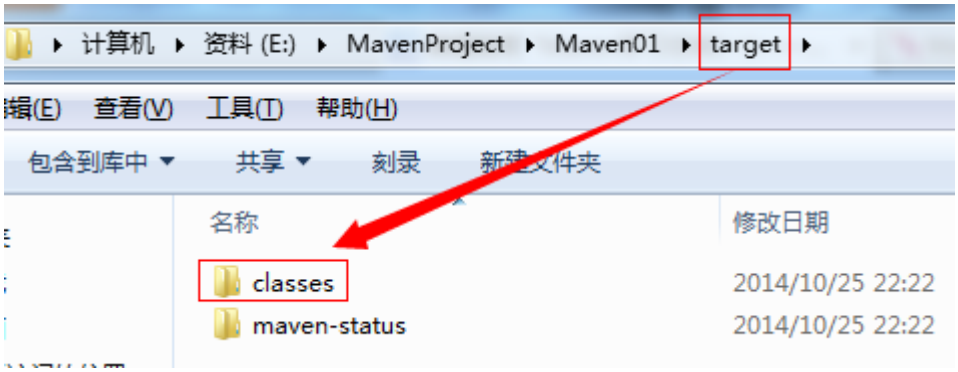
编译成功

Maven在编译时，会自动去Maven的中心仓库把相关连的jar包自动下载到本地的jar文件仓库中

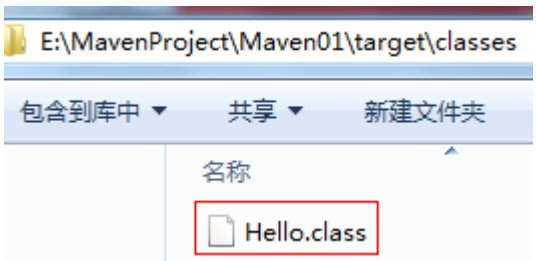
使用Maven编译完成之后，在项目根目录下会生成一个target文件夹，如下图所示：



打开target文件夹，可以看到里面有一个classes文件夹，如下图所示：



classes文件夹存放的就是编译成功后生成的.class文件，如下图所示：



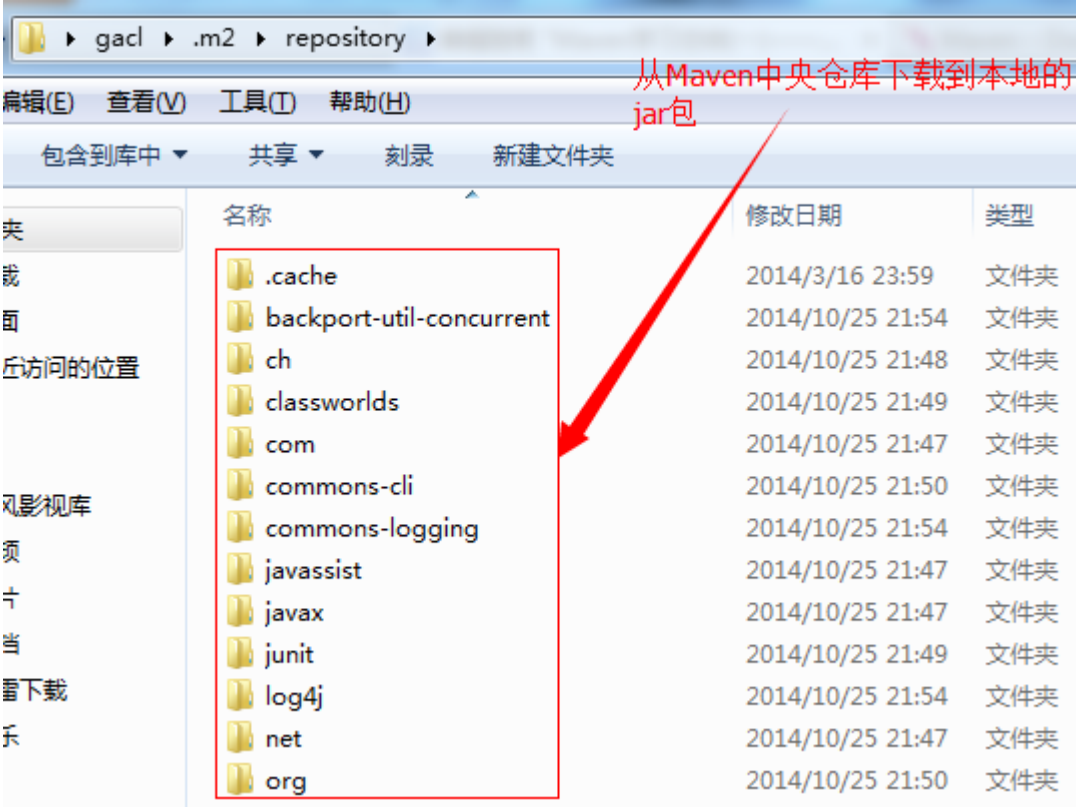
5.使用"mvn clean"命令清除编译结果，也就是把编译生成的target文件夹删掉，如下图所示：

```
管理员: C:\Windows\system32\cmd.exe
E:\MavenProject\Maven01>mvn clean
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building hello-first SNAPSHOT-0.0.1
[INFO] ------------------------------------------------------------------------
Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/2.5/ma
ven-clean-plugin-2.5.pom
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/2.5/mav
en-clean-plugin-2.5.pom (4 KB at 1.4 KB/sec)
Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins/22/maven-pl
ugins-22.pom
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins/22/maven-plu
gins-22.pom (13 KB at 15.1 KB/sec)
Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/2.5/ma
ven-clean-plugin-2.5.jar
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/2.5/mav
en-clean-plugin-2.5.jar (25 KB at 16.0 KB/sec)
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ hello-first ---
Downloading: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.0/plexus-utils-
3.0.pom
Downloaded: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.0/plexus-utils-3
.0.pom (4 KB at 5.1 KB/sec)
Downloading: https://repo.maven.apache.org/maven2/org/sonatype/spice/spice-parent/16/spice-parent-16
.pom
Downloaded: https://repo.maven.apache.org/maven2/org/sonatype/spice/spice-parent/16/spice-parent-16.
pom (9 KB at 10.5 KB/sec)
Downloading: https://repo.maven.apache.org/maven2/org/sonatype/forge/forge-parent/5/forge-parent-5.p
om
Downloaded: https://repo.maven.apache.org/maven2/org/sonatype/forge/forge-parent/5/forge-parent-5.po
m (9 KB at 10.2 KB/sec)
Downloading: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.0/plexus-utils-
3.0.jar
Downloaded: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.0/plexus-utils-3
.0.jar (221 KB at 14.8 KB/sec)
[INFO] Deleting E:\MavenProject\Maven01\target
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 23.210 s
[INFO] Finished at: 2014-10-25T21:58:15+08:00
        半:
```

执行完"mvn clean"命令后，target文件夹就会被删除了。

五、修改从Maven中心仓库下载到本地的jar包的默认存储位置

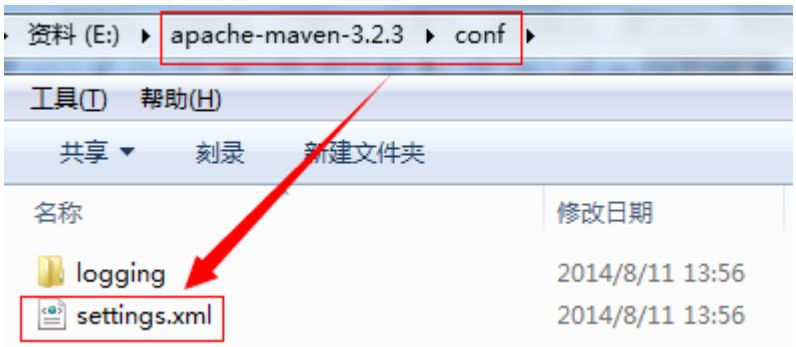
从Maven中心仓库下载到本地的jar包的默认存放在”${user.home}/.m2/repository”中，${user.home}表示当前登录系统的用户目录(如"C:\Users\gacl")，如下图所示



Jar包存放在这个位置不太好，我们希望能够自己定义下载下来的jar包的存放位置，因此我们可以自己设置下载到本地时的jar包的存放目录。

在“E:\”目录下创建一个“repository”文件夹

找到apache-maven-3.2.3\conf目录下的settings.xml文件，如下图所示：



编辑setting.xml文件，如下图所示：

```
46  <settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
47            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
48            xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd">
49    <!-- localRepository
50     | The path to the local repository maven will use to store artifacts.
51     |
52     | Default: ${user.home}/.m2/repository
53    <localRepository>/path/to/local/repo</localRepository>
54    -->
```

jar包默认的下载路径

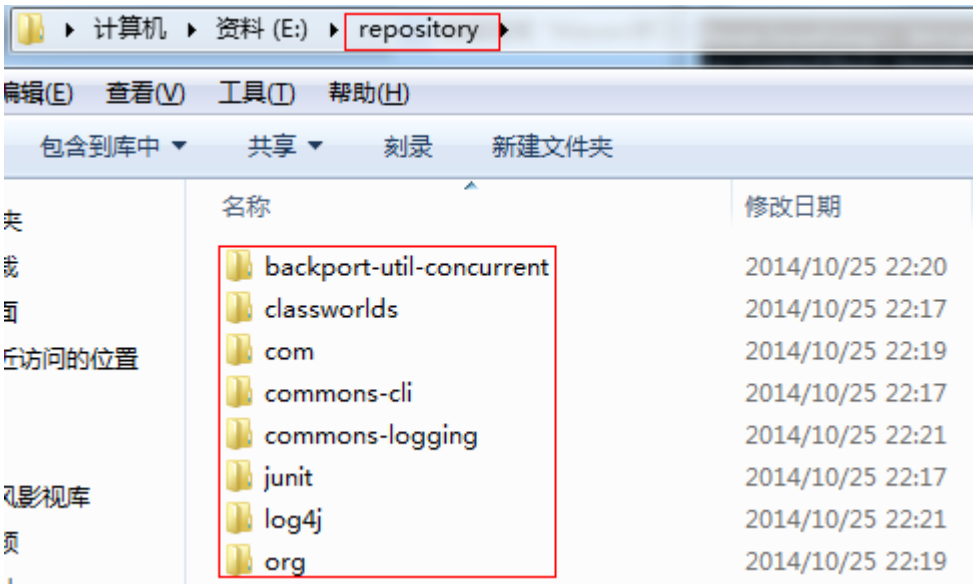
加上下面的代码

```
1 <localRepository>E:/repository</localRepository>
```

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <!-- localRepository
   | The path to the local repository maven will use to store artifacts.
   |
   | Default: ${user.home}/.m2/repository
  <localRepository>/path/to/local/repo</localRepository>
  -->
  <localRepository>E:/repository</localRepository>
```

设置E盘下的repository目录作为maven下载的jar包的默认存储目录

这样就可以把jar包下载到我们指定的E:/repository目录中了，如下图所示：



把jar包下载到本地的好处就是，当编译时，会优先从本地的jar包去找，如果本地存在，就直接拿来用，如果不存在，就从Maven的中心仓库去下载。如下图所示：

```
管理员: C:\Windows\system32\cmd.exe

E:\MavenProject\Maven01>mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building hello-first SNAPSHOT-0.0.1
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ hello-first ---
[WARNING] Using platform encoding (GBK actually) to copy filtered resources, i.e. build is platform
dependent!
[INFO] skip non existing resourceDirectory E:\MavenProject\Maven01\src\main\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ hello-first ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding GBK, i.e. build is platform depend
ent!
[INFO] Compiling 1 source file to E:\MavenProject\Maven01\target\classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 1.594 s
[INFO] Finished at: 2014-10-25T22:27:10+08:00
[INFO] Final Memory: 9M/23M
[INFO] ------------------------------------------------------------------------
E:\MavenProject\Maven01>mvn clean
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building hello-first SNAPSHOT-0.0.1
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ hello-first ---
[INFO] Deleting E:\MavenProject\Maven01\target
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 0.249 s
[INFO] Finished at: 2014-10-25T22:27:20+08:00
[INFO] Final Memory: 4M/15M
[INFO] ------------------------------------------------------------------------
E:\MavenProject\Maven01>
        半:
```

第一次执行"mvn compile"和"mvn clean"这两个命令时，Maven会去中央仓库下载需要的jar包，而第二次执行这两个命令时，由于所需的jar包已经在本地的仓库中存储，所以就可以直接拿来用了，这样就省去了去中央仓库下载jar包的时间。

以上就是Maven的简单入门讲解。

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Stopwatch的使用
{
    class Program
    {

        /// <summary>
```

```
13          /// 使用Stopwatch比较for循环和foreach循环的效率
14          /// </summary>
15          /// <param name="args"></param>
16          static void Main(string[] args)
17          {
18
19              int[] intArr = new int[1000000];
20              for (int i = 1; i <= 1000000; i++)
21              {
22                  intArr[i - 1] = i;
23              }
24
25              //使用Stopwatch统计程序运行的时间
26              /*
27               Stopwatch提供了几个方法用以控制Stopwatch对象。
28               * Start方法开始一个计时操作，Stop方法停止计时。
29               * 此时如果第二次使用 Start方法，将继续计时，最终的计时结果为两次计时的累加。
30               * 为避免这种情况，在第二次计时前用Reset方法将对象归零。这三个方法都不需要参数
31               */
32              System.Diagnostics.Stopwatch sw = new System.Diagnostics.Stopwatch();
33              sw.Start(); //开始计时
34
35              int sum = 0;
36
37              for (int i = 1; i <= intArr.Length; i++)
38              {
39                  sum += i;
40              }
41              Console.WriteLine("sum={0}",sum);
42
43              sw.Stop(); //停止计时
44              long result = sw.ElapsedMilliseconds;//获得程序运行的时间
45              Console.WriteLine("使用for循环计算从1加到1000000的和所需要的时间是：{0}", result);
46
47              Console.WriteLine("**********************************************************************");
48              sw.Reset();
49              sw.Start();
50
51              sum = 0;
52              foreach (var item in intArr)
53              {
54                  sum += item;
55              }
56              Console.WriteLine("sum={0}", sum);
57              Console.WriteLine("使用foreach循环计算从1加到1000000的和所需要的时间是：{0}",
sw.ElapsedMilliseconds);
58              sw.Stop();
59
60              Console.ReadKey();
61          }
```

```
62         }
63 }
```