



云计算上机实验报告 3

所在学院：_____ 计算机学院

所在专业：_____ 计算机科学与技术

实验题目：_____ 实验 3：CloudSim 练习

学生学号：_____ 16041321

学生姓名：_____ 黄继升

指导老师：_____ 黄杰

实验日期：_____ 2019 年 6 月 10 日

1、实验目的

CloudSim 是一个云计算仿真平台软件，提供给研究和设计人员做仿真实验。通过本次实验，初步了解 CloudSim，通过学习 CloudSim 内置的仿真实例，理解云计算应用设计的方法。

2、实验内容

(1) CloudSim 平台的安装

CloudSim 是用 java 开发的跨平台软件，在个人电脑上安装部署 CloudSim。

(2) 运行 CloudSim 的 Examples

CloudSim 带有 Basic Examples、Network Examples、Power Examples、Container Examples 等例子。尝试正确运行其中的一个，并理解过程和运行结果。

参考资料（不限于）：

<https://blog.csdn.net/lhakuma/article/details/78754957>

<http://www.cloudbus.org/cloudsim/>

<http://www.cloudbus.org/cloudsim/examples.html>

3、实验报告

3.1 安装环境准备

- (1) 系统环境: Windows 10
- (2) 配置工具: jdk1.7.0_80、apache-maven-3.2.5-bin、
cloudsim-cloudsim-4.0
- (3) 编程平台: eclipse-jee-2019-03-R-win32-x86_64

3.2 配置流程

3.2.1 安装配置 JDK 环境

我自己的电脑之前安装过 3 个 jdk, 版本依次 1.6、1.7 和 1.8。安装这么多版本 jdk, 主要是因为之前做的一些项目对 jdk 的版本有要求。3 个 jdk 的路径依次为:

jdk1.6: C:\Program Files\Java\jdk1.6.0_45

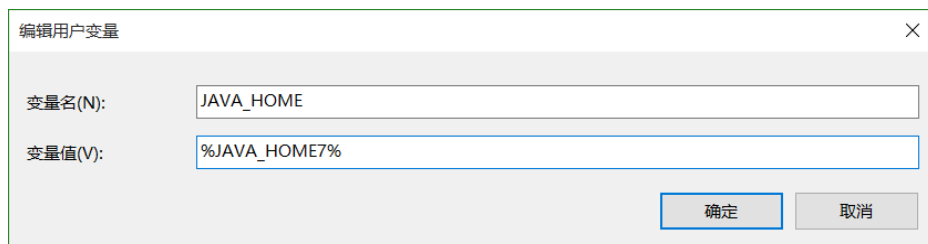
jdk1.7: C:\Program Files\Java3\jdk1.7.0_80

jdk1.8: C:\Program Files\Java2\jdk1.8.0_191

创建环境变量时, 我是这样设置的:

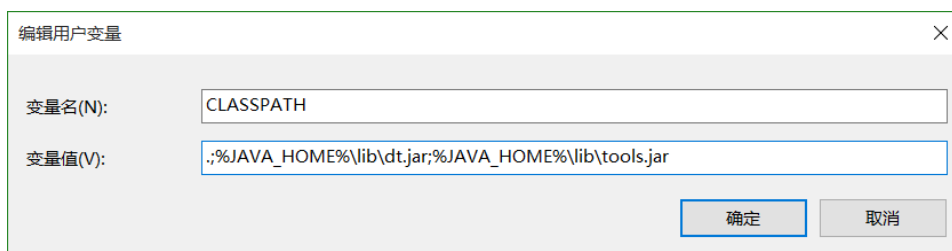
JAVA_HOME	%JAVA_HOME7%
JAVA_HOME6	C:\Program Files\Java\jdk1.6.0_45
JAVA_HOME7	C:\Program Files\Java3\jdk1.7.0_80
JAVA_HOME8	C:\Program Files\Java2\jdk1.8.0_191

先分别为每个 jdk 创建对应的“JAVA_HOME+编号”环境变量, 然后创建一个总的环境变量“JAVA_HOME”, 其变量值为 %JAVA_HOME“编号”%, 如果要切换系统的 jdk, 只要修改“JAVA_HOME”对应的变量值的“编号”就可以了



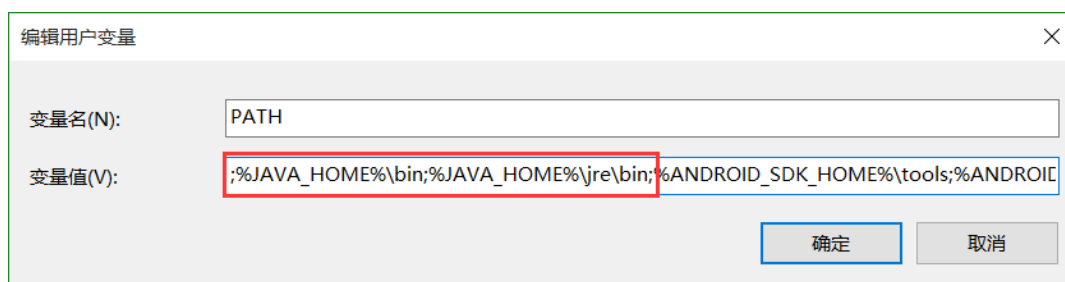
接下来创建 “CLASSPATH” 环境变量，变量值为：

`.;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar`



最后在“PATH”变量中增加内容：

`;%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin;`



最后为了测试是否安装(切换)jdk 成功，快捷键 win+R 打开运行，输入 cmd 命令调出 windows 命令窗口。我之前切换到了 jdk1.7 版本，依次输入命令 `java -version`、`javac`，检验结果如下：

```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\asus>java -version
java version "1.7.0_80"
Java(TM) SE Runtime Environment (build 1.7.0_80-b15)
Java HotSpot(TM) 64-Bit Server VM (build 24.80-b11, mixed mode)

C:\Users\asus>
```

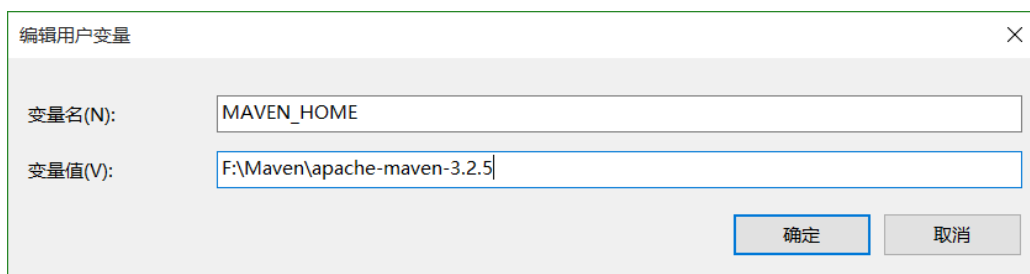
```
C:\Users\asus>javac
用法: javac <options> <source files>
其中，可能的选项包括:
-g          生成所有调试信息
-g:none     不生成任何调试信息
-g:{lines,vars,source} 只生成某些调试信息
-nowarn     不生成任何警告
-verbose    输出有关编译器正在执行的操作的消息
-deprecation 输出使用已过时的 API 的源位置
-classpath <路径> 指定查找用户类文件和注释处理程序的位置
-cp <路径> 指定查找用户类文件和注释处理程序的位置
```

此时可证明安装(切换)jdk1.7 成功。

3.2.2 安装配置 Maven 环境

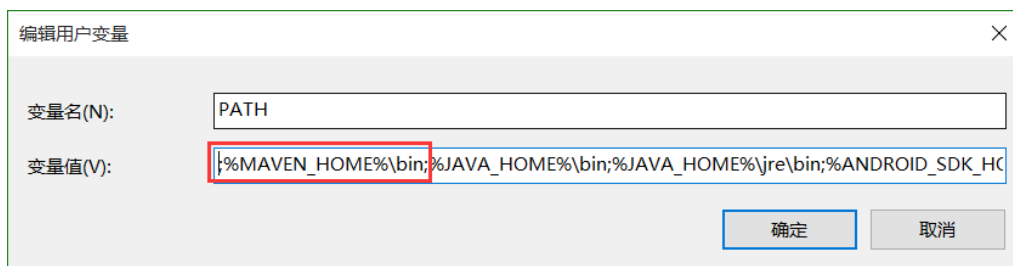
将 apache-maven-3.2.5-bin.zip 解压到 F:\Maven 目录下，并增加“MAVEN_HOME”环境变量，变量值为：

F:\Maven\apache-maven-3.2.5



“Path” 环境变量的变量值添加内容：

; %MAVEN_HOME%\bin;



最后通过 cmd 命令窗口运行命令：mvn -v 检查 Maven 是否安装成功

```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

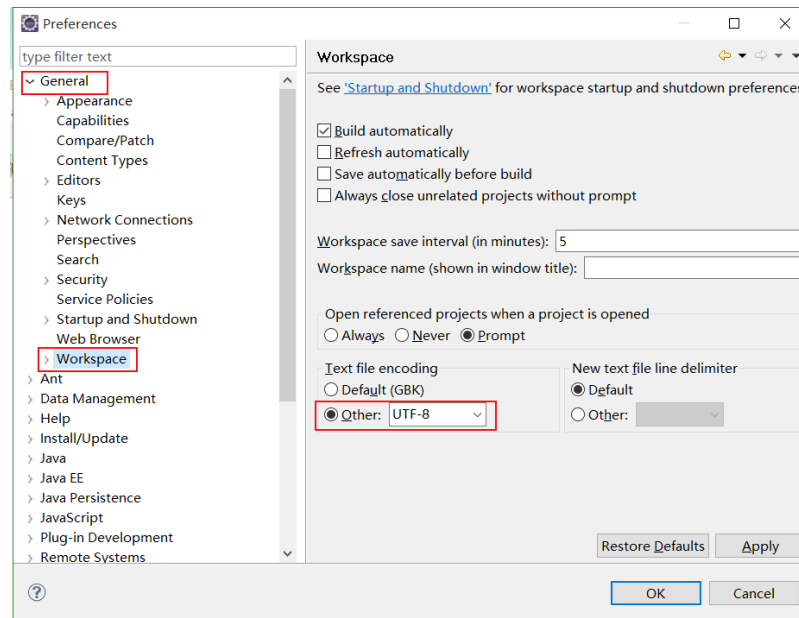
C:\Users\asus>mvn -v
Apache Maven 3.2.5 (12a6b3acb947671f09b81f49094c53f426d8cea1; 2014-12-15T01:29:23+08:00)
Maven home: F:\Maven\apache-maven-3.2.5\bin\..
Java version: 1.7.0_80, vendor: Oracle Corporation
Java home: C:\Program Files\Java3\jdk1.7.0_80\jre
Default locale: zh_CN, platform encoding: GBK
OS name: "windows 8.1", version: "6.3", arch: "amd64", family: "windows"
C:\Users\asus>
```

此时可证明安装 Maven 3.2.5 成功。

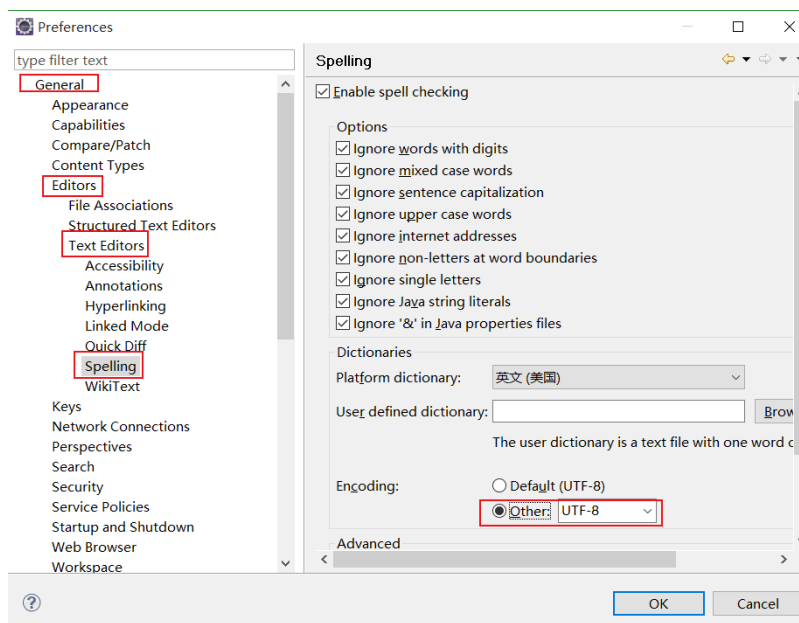
3.2.3 安装和配置 Eclipse-jee

由于我电脑之前安装过 Eclipse-jee，所以这里不展开如何安装的教程。实际上直接下载，解压后运行里面的 eclipse.exe 文件就可以了，不过要注意的是有些版本的 eclipse 是需要自己手动导入 jdk 的。另外还要对 Eclipse 设置编码格式为 utf-8，防止乱码输出。

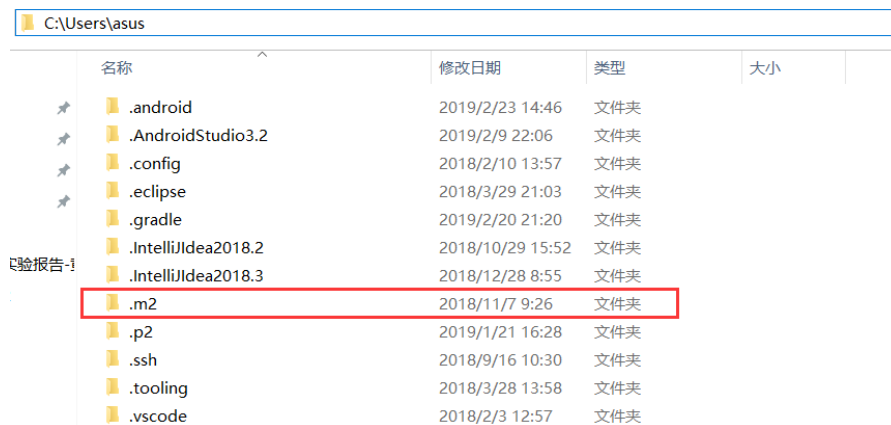
Windows->General->Workspace 下更改编码如下：



windows->preferences->general->Editors->TextEditors->spelling 下更改编码如下：

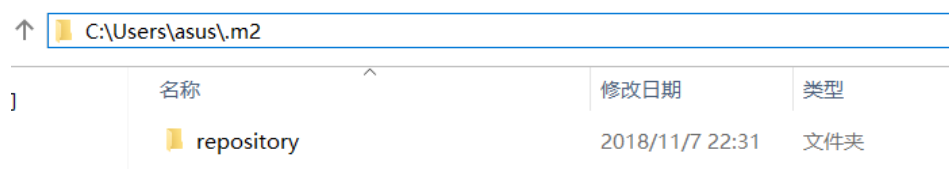


3.2.4 修改 Maven 的默认本地仓库路径，并更新 Eclipse 的 Maven 配置



	名称	修改日期	类型	大小
	.android	2019/2/23 14:46	文件夹	
	.AndroidStudio3.2	2019/2/9 22:06	文件夹	
	.config	2018/2/10 13:57	文件夹	
	.eclipse	2018/3/29 21:03	文件夹	
	.gradle	2019/2/20 21:20	文件夹	
	.IntelliJdea2018.2	2018/10/29 15:52	文件夹	
	.IntelliJdea2018.3	2018/12/28 8:55	文件夹	
	.m2	2018/11/7 9:26	文件夹	
	.p2	2019/1/21 16:28	文件夹	
	.ssh	2018/9/16 10:30	文件夹	
	.tooling	2018/3/28 13:58	文件夹	
	.vscode	2018/2/3 12:57	文件夹	

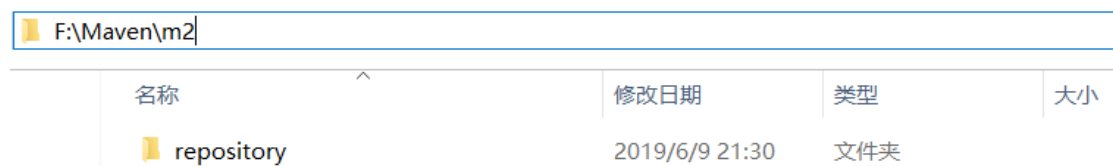
安装 Maven 后会默认在 C 盘的 User 目录下生成 .m2 文件夹，用于存放 Maven 本地仓库，Maven 能够根据 pom.xml 配置文件从自动下载的所有 jar 包到仓库里，因此可能会造成 C 盘空间不足，所以需要更改 jar 包默认保存路径。



	名称	修改日期	类型
	repository	2018/11/7 22:31	文件夹

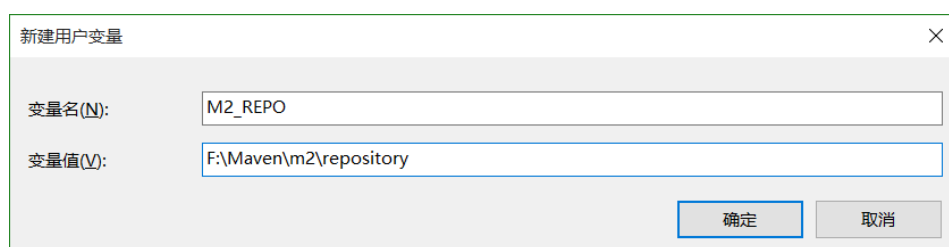
修改方法如下：

(1) 创建新的本地仓库路径文件夹：F:\Maven\m2\repository



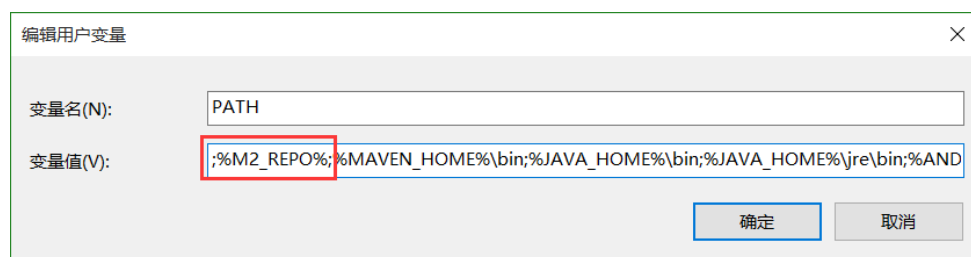
	名称	修改日期	类型	大小
	repository	2019/6/9 21:30	文件夹	

(2) 新增系统环境变量“M2_REPO”，变量值为：F:\Maven\m2\repository



变量名(N):	M2_REPO
变量值(V):	F:\Maven\m2\repository
<input type="button" value="确定"/> <input type="button" value="取消"/>	

(3) “Path” 环境变量的变量值添加内容:



(4) 修改 Maven 安装目录下的 conf/文件夹内的 setting.xml 文件, 新增一行: <localRepository>F:\Maven\m2\repository</localRepository> (表示本地仓库的地址为: F:\Maven\m2\repository);

```
<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/sett
  <!-- 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>F:\Maven\m2\repository</localRepository>
```

(5) 将上一步修改好的 setting.xml 文件复制到 D:\Java\m2\repositor 目录下; 一份;

F:\Maven\m2\repository				
名称	修改日期	类型	大小	
settings.xml	2019/6/9 21:47	XML 文件	11 KB	

(6) 打开 Eclipse

由于我的 Eclipse 之前没有 Maven 选项, 因此需要自己手动下载, 方法如下:

选择: Help -> Install New Software -> add, 在 name 和 Location 分别添加以下地址, next 即可:

name: m2e

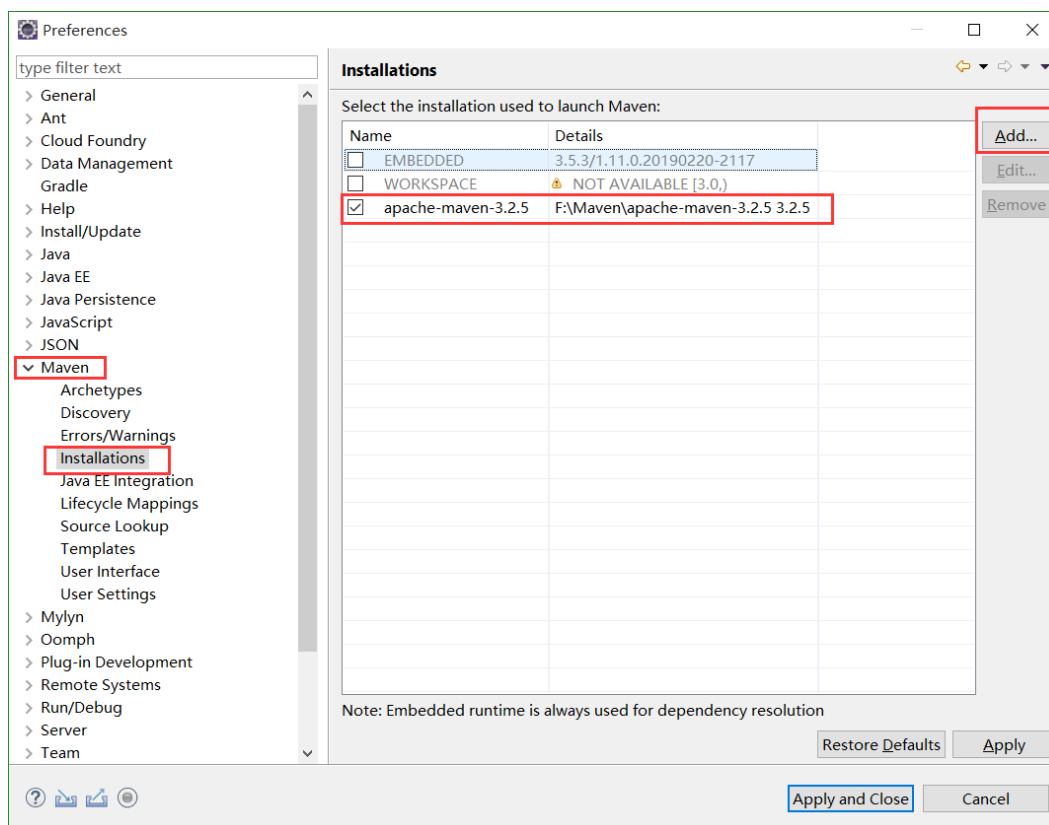
Location: <http://download.eclipse.org/technology/m2e/releases>

然后是添加 Maven 路径。

选择: Windows->Preferences->Maven->installations 下

Add maven 的安装目录: F:\Maven\apache-maven-3.2.5, 并勾选。并点击下面的 browse 按钮打开 maven 的全局配置文件:

F:\Maven\apache-maven-3.2.5\conf\setting.xml



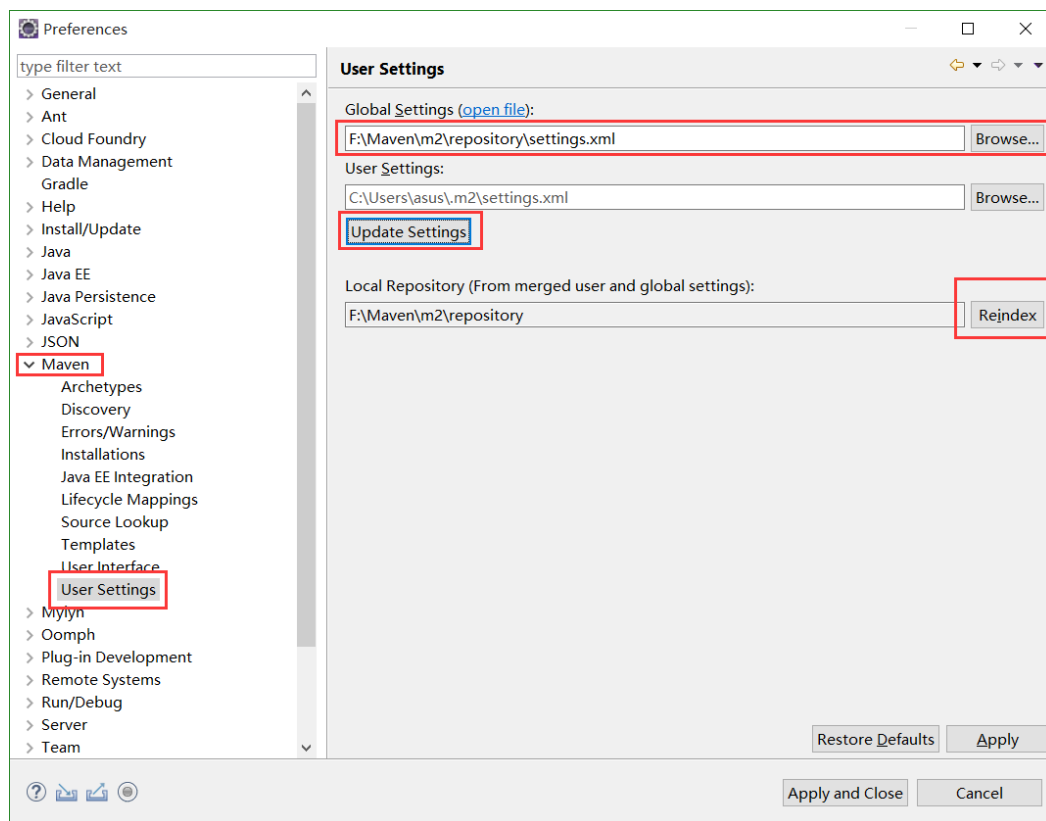
由于之前修改了默认的 Maven 本地仓库路径, 因此还需要进行如下配置:

选择: Preferences->Maven->

User Settings 下, 修改 User Settings 为:

F:\Maven\m2\repository\setting.xml

并点击 Update Settings。并点击下面的 reindex, 最后生效所有设置。



(7) 运行 cmd 命令窗口，执行命令：mvn help:system 进行测试。

```
C:\Windows\system32\cmd.exe - mvn help:system
Microsoft Windows [版本 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\asus>mvn help:system
[INFO] Scanning for projects...
Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/2.5/maven-clean-plugin-2.5.pom
[WARNING] Failed to retrieve plugin descriptor for org.apache.maven.plugins:maven-clean-plugin:2.5: Plugin org.apache.maven.plugins:maven-clean-plugin:2.5 or one of its dependencies could not be resolved: Failed to read artifact descriptor for org.apache.maven.plugins:maven-clean-plugin:jar:2.5
Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-install-plugin/2.4/maven-install-plugin-2.4.pom
```

此后,Maven 从远程库下载的 jar 包都会放到新修改后的仓库路径中

3.2.4 部署 CloudSim4.0

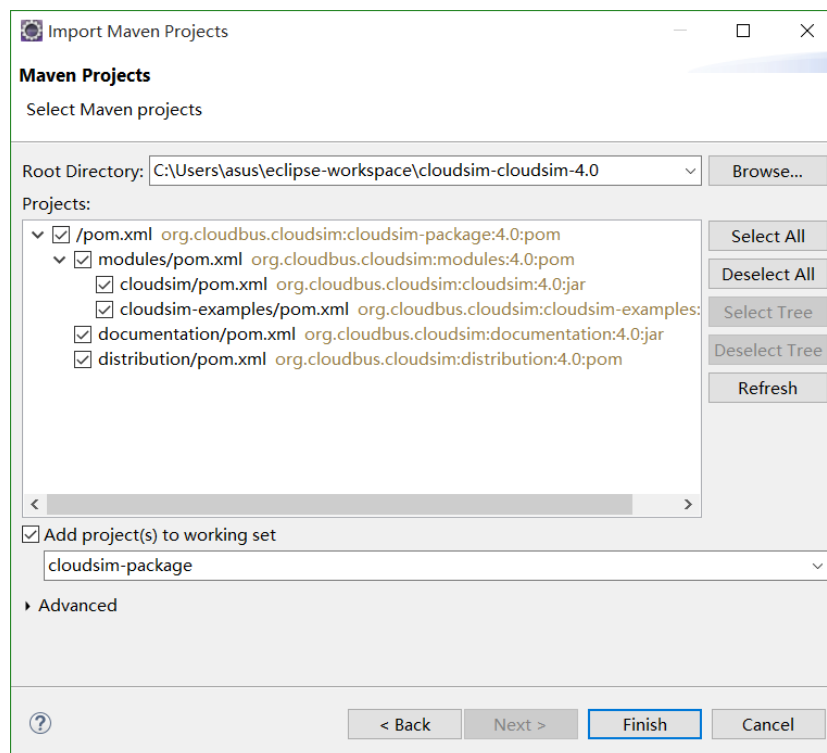
解压 cloudsim-cloudsim-4.0.zip 到当前 Eclipse 的工作空间:

C:\Users\asus\eclipse-workspace;

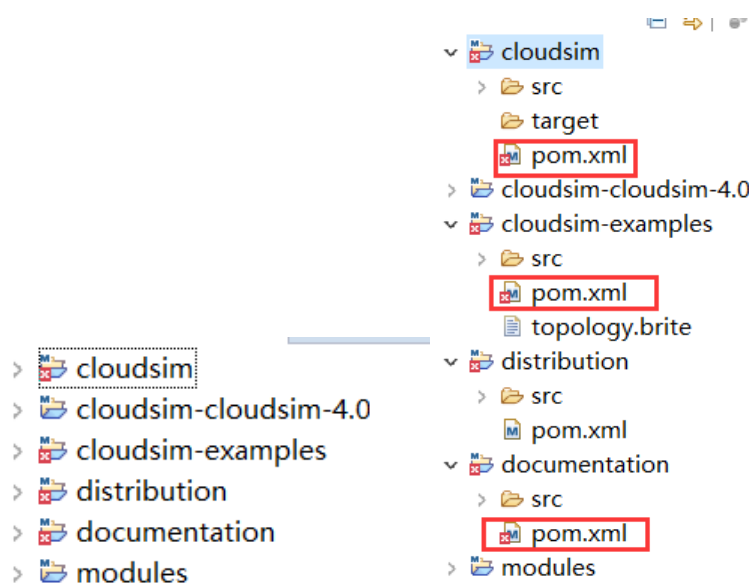
并在 Eclipse 中导入 cloudsim-cloudsim-4.0 项目:

File->Import...->Maven->Existing Maven Projects->next->Browse

最后点击 **Finish** 完成整个项目的加载就可以了，此时 **Maven** 正在根据 **pom.xml** 配置文件从云端下载向 **jar** 包。



最后如果加载成功，就可以从左侧的工作区域中看到产生的几个工程文件，我之前出了一点小意外，几个工程全都报错了...



Description	Resource	Path	Location	Type
❌ Maven Configuration Problem (3 items)				
❌ Failure to transfer org.apache.maven.pl	pom.xml	/cloudsim	line 1	Maven Con...
❌ Failure to transfer org.apache.maven.pl	pom.xml	/cloudsim-exam...	line 1	Maven Con...
❌ Failure to transfer org.apache.maven.pl	pom.xml	/documentation	line 1	Maven Con...
❌ Maven Problems (3 items)				
🔧 Project configuration is not up-to-date	cloudsim-e...		line 1	Maven Con...
🔧 Project configuration is not up-to-date	distribution		line 1	Maven Con...
🔧 Project configuration is not up-to-date	documenta...		line 1	Maven Con...

详细错误信息如下, 说是缺少 `maven-resources-plugin-2.6.jar` 这个 jar 包:

❌ Failure to transfer
org.apache.maven.plugins:maven-resources-plugin:pom:2.6 from
https://repo.maven.apache.org/maven2 was
cached in the local repository, resolution will not
be reattempted until the update interval of
central has elapsed or updates are forced. Original
error: Could not transfer artifact
org.apache.maven.plugins:maven-resources-plugin:pom:2.6 from/to central
(https://repo.maven.apache.org/maven2):
Received fatal alert: protocol_version

结果去 Maven 仓库查看是否有这个 jar 包, 发现是有的:

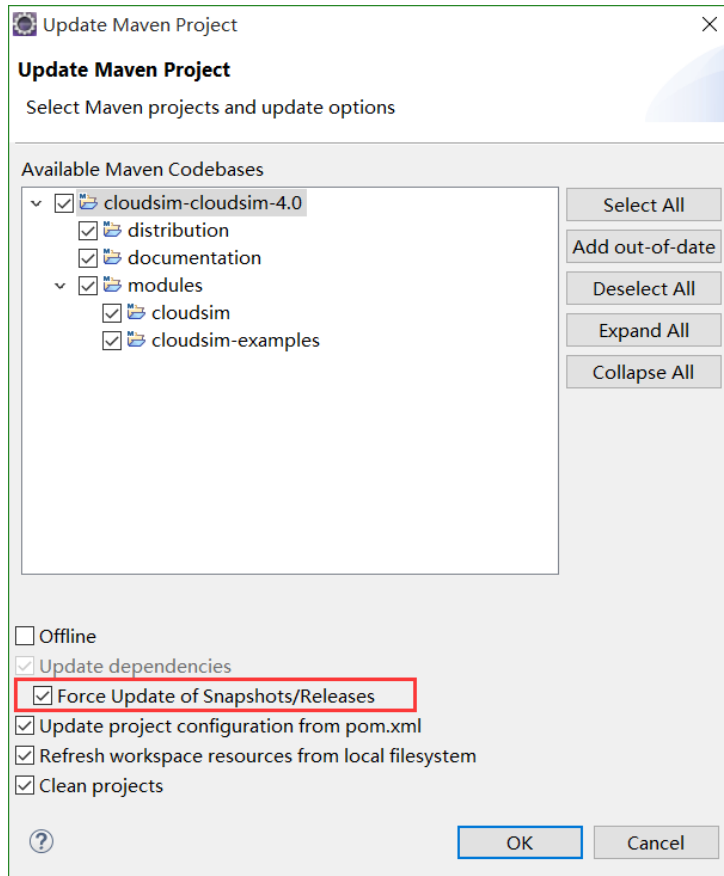
F:\Maven\m2\repository\org\apache\maven\plugins\maven-resources-plugin\2.6				
名称	修改日期	类型	大小	
_remote.repositories	2019/6/10 10:25	REPOSITORIES ...	1 KB	
m2e-lastUpdated.properties	2019/6/10 10:25	Properties 源文件	1 KB	
maven-resources-plugin-2.6.jar	2019/6/10 10:08	Executable Jar File	29 KB	
maven-resources-plugin-2.6.jar.sha1	2019/6/10 10:08	SHA1 文件	1 KB	
maven-resources-plugin-2.6.pom.las...	2019/6/10 0:32	LASTUPDATED ...	1 KB	
maven-resources-plugin-2.6-sources...	2019/6/10 10:25	Executable Jar File	21 KB	
maven-resources-plugin-2.6-sources...	2019/6/10 10:25	SHA1 文件	1 KB	

我自己个人尝试了网上无数种方法, 都没什么用... 诶

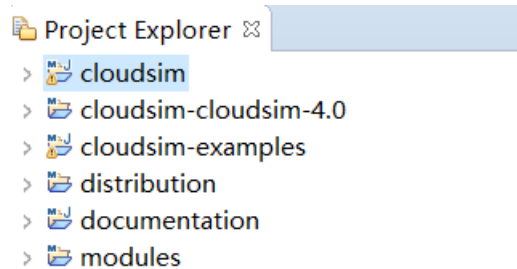
最后我自己才误打误撞地自己找到了一种有效的解决方法, 特此记录:

选中出错的所有项目 -> 右键 -> Maven -> UpdateProject..

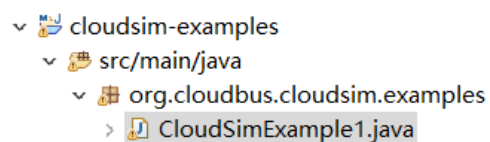
然后在出来的弹窗选项将所有出错项目全选, 然后一定要勾选那个 **Force Update of Snapshots/Releases**, 然后点击 **OK** 就可以解决项目出错的问题了。原因可能是远程仓库更新或增加了对应的 jar 包, 所以要通过这种方式来强制更新项目。



项目恢复正常:



3.3 运行实例 CloudSimExample1.java



3.3.1 运行结果控制台显示

```
Starting CloudSimExample1...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: Sending cloudlet 0 to VM #0
400.1: Broker: Cloudlet 0 received
400.1: Broker: All Cloudlets executed. Finishing...
400.1: Broker: Destroying VM #0
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

===== OUTPUT =====
Cloudlet ID   STATUS   Data center ID   VM ID   Time   Start Time   Finish Time
    0         SUCCESS       2         0    400       0.1       400.1
CloudSimExample1 finished!
```

3.3.2 对源码和运行过程的学习和分析

通过查阅资料以及阅读源代码,我了解到 `CloudSimExample1.java` 是模拟一个云仿真环境,即创建一个一台主机(一个云用户),一个任务的数据中心。主要的运行流程如下:

首先,在类 `CloudSimExample1` 中定义声明了两个队列:云任务队列 `cloudletList` 和虚拟机队列 `vmList`,然后整个程序由几个方法组成:

(1) `main()`方法:主线程方法

运行过程:1. 打印云仿真开始提示信息;

2. 定义云用户数量为1;

3. 调用日历实例:

4. 设置故障错误标志位 `trace_flag`。初始值为 `false`;

5. 通过2、3、4初始化 `CloudSim` 库;

6. 调用 `createDatacenter()`方法创建名为“Datacenter_0”的数据中心;

7. 调用 `createBorker()`方法创建 `DatacenterBroker` 类对象

broker，即代理；

8. 创建并实例化虚拟机队列对象 **vmList**；
9. 声明和赋值虚拟机的各个属性参数，包括虚拟机 **Id**，**MIPS**(运算能力)、镜像大小、虚拟机内存大小、带宽、**CPU** 数量以及虚拟机名称；
10. 根据 9 所定义的属性参数和 **broker** 的 **Id**，创建并实例化一台虚拟机 **Vm** 对象；
11. 将 **Vm** 对象加入到虚拟机队列中；
12. 将虚拟机队列提交给代理 **broker**
13. 创建一个云任务 **Cloudlet** 对象，过程和创建一台虚拟机对象一样，先创建和实例化云任务队列，再声明和赋值所有云任务所需的参数（任务 **Id**，长度，文件大小，输出大小以及使用模式），然后根据这些参数来创建和实例化云任务对象，最后加入到云任务队列中，并将队列提交给代理；
14. 调用 **CloudSim.startSimulation()** 启动仿真；
15. 调用 **CloudSim.stopSimulation()** 停止仿真；
16. 调用 **broker.getCloudletReceivedList()** 来获取类型为 **List<Cloudlet>** 的结果对象 **newList**，并调用 **printCloudletList(newList)** 打印云任务运行结果，以及调用 **datacenter0.printDebts()** 用户使用数据中心的情况；
17. 打印云仿真结束提示信息；

(2) **createDatacenter()** 方法：创建数据中心

- 运行过程：
1. 创建 **Host** 类型队列对象 **hostList**，用于存储 **host** 机器
 2. 创建 **Pe** 类型队列对象 **peList**，由于存储多个内核或 **CPU**
 3. 定义 **int mips = 1000**；用于表示运算速度
 4. 根据 **PEid**(这里为 0)和 **mips** 创建 **Pe** 对象，并加入到 **peList** 队列中；
 5. 定义 **Host** 的各项参数（**hostId**、内存、存储容量、带宽），并以此创建和实例化 **Host** 对象，加入到 **hostList** 中。
 6. 创建数据中心特征 **DatacenterCharacteristics** 对象来存

储数据中心的参数特征：结构、操作系统、机器列表、分配策略（时间/空间共享）、时间区域、花费（以 G 和 Pe 为单位，Pe 指的是 CPU 单元）、内存花费、存储操作花费、带宽花费；

7. 根据以上所有结果创建数据中心 **Datacenter** 对象；

8. 返回对象；

(3) **createBorker()**方法：创建代理部分

运行过程：1. 将创建的方法封装为单例模式，保证全局只有唯一一个代理对象

2. 返回该对象

(4) **printCloudletList(List list)**方法：循环遍历云任务结果列表，并按照一定格式打印输出。

3.3.3 运行结果分析

```
Cloudlet ID : 0
STATUS : SUCCESS
Data center ID : 2
VM ID : 0
Time : 400
Start Time : 0.1
Finish Time : 400.1
```

即：云任务 0 运行成功，开始时间为 0.1，结束时间为 400.1。总共花费了 400 时间。