工具介绍

 |

陈霖 陈思宜 王子杰 赵玉峰

**目 录**

[1. Git 1](#_Toc5019)

[1.1 下载Github 1](#_Toc4826)

[1.2 命令行方式使用GitHub 1](#_Toc28892)

[1.3 使用图形界面GitHub Desktop 6](#_Toc24648)

[2. SVN 9](#_Toc24478)

[2.1 SVN简介 9](#_Toc31483)

[2.2 下载SVN 9](#_Toc5685)

[2.3 VisualSVN Server配置与使用 9](#_Toc4163)

[2.4 SVN的使用 13](#_Toc17366)

[2.5 SVN冲突 14](#_Toc29981)

[3. Maven 17](#_Toc10677)

[3.1下载maven 17](#_Toc27984)

[3.2 安装Maven 17](#_Toc3654)

[3.3 安装eclipse maven插件 17](#_Toc30785)

[3.4 新建maven项目 18](#_Toc25384)

[3.5 导入maven项目 19](#_Toc14745)

[3.6 配置pom.xml 21](#_Toc31686)

[3.7 运行Maven 22](#_Toc1012)

[4. Jenkins 24](#_Toc25015)

[4.1 Jenkins介绍 24](#_Toc32308)

[4.2 安装与全局配置 24](#_Toc32097)

[4.3 建立新的集成任务 27](#_Toc24612)

[4.4 执行与调试集成任务 28](#_Toc15465)

[4.5 修改、删除集成任务 29](#_Toc27033)

[5. Junit 30](#_Toc27899)

[6. Checkstyle 30](#_Toc3890)

[6.1安装eclispe插件 30](#_Toc31722)

[6.2使用Checkstyle 31](#_Toc29274)

[6.2使用Maven时Checkstyle检查 32](#_Toc30006)

# Git

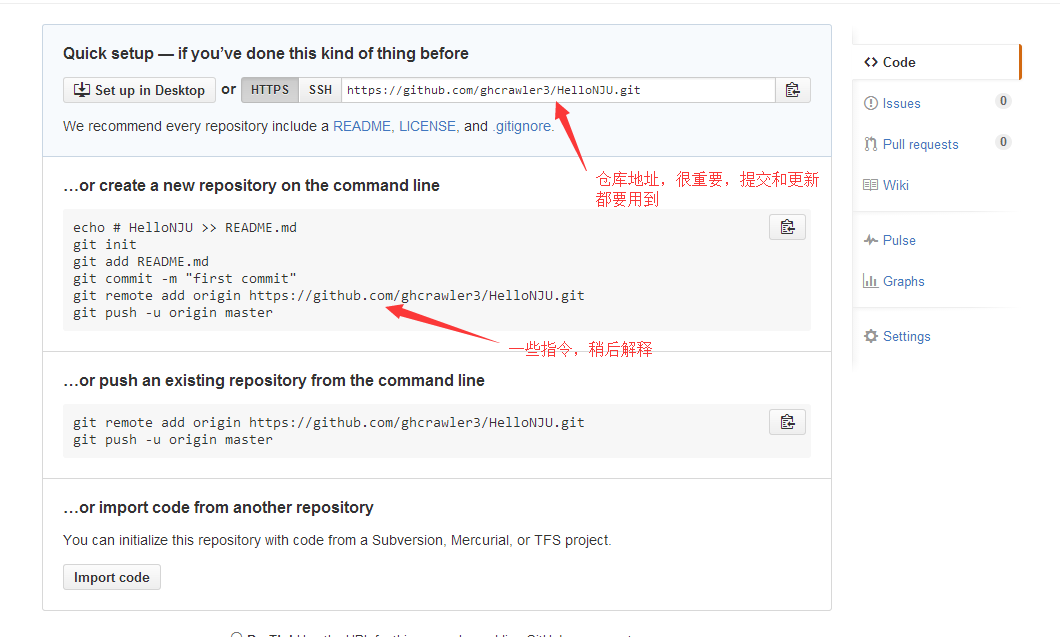
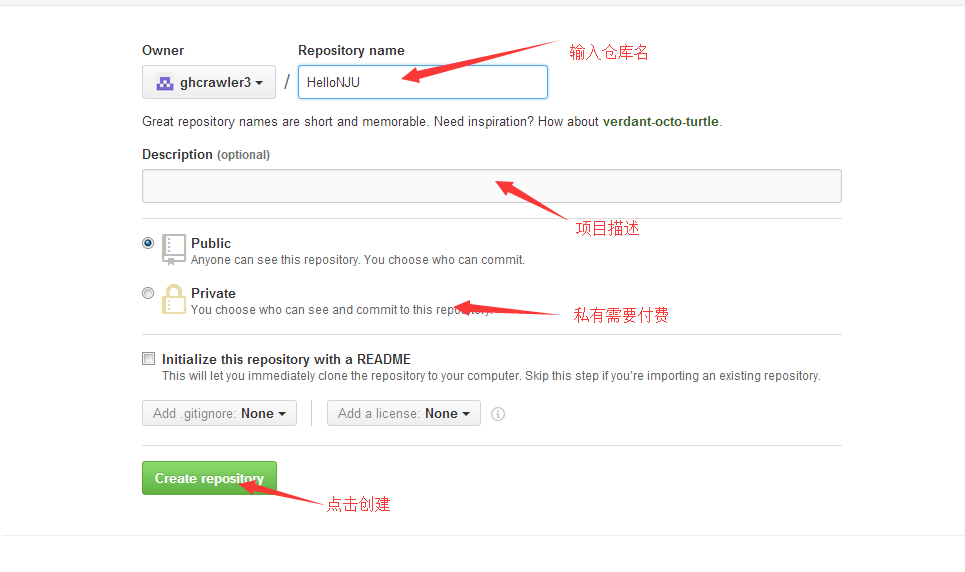
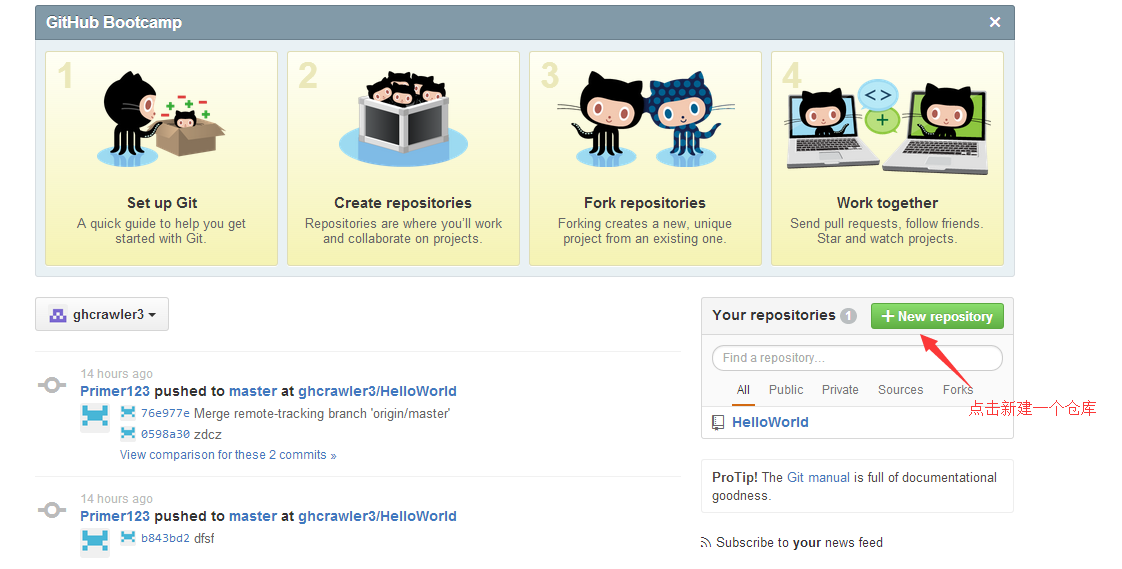
## 下载Github

git下载地址：<http://www.git-scm.com/download/>

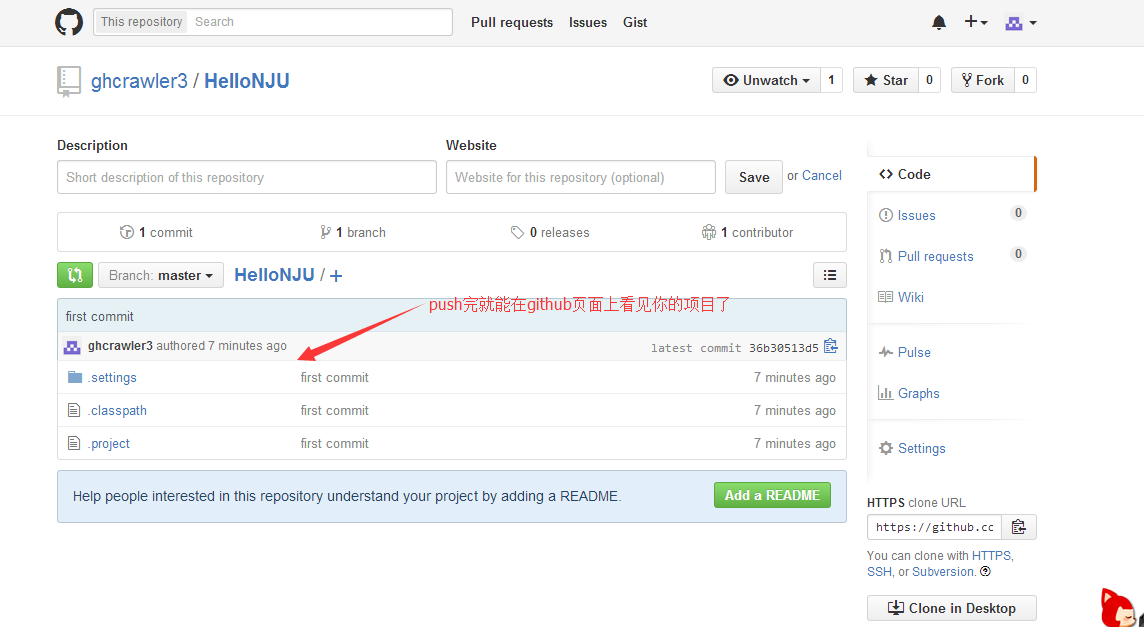
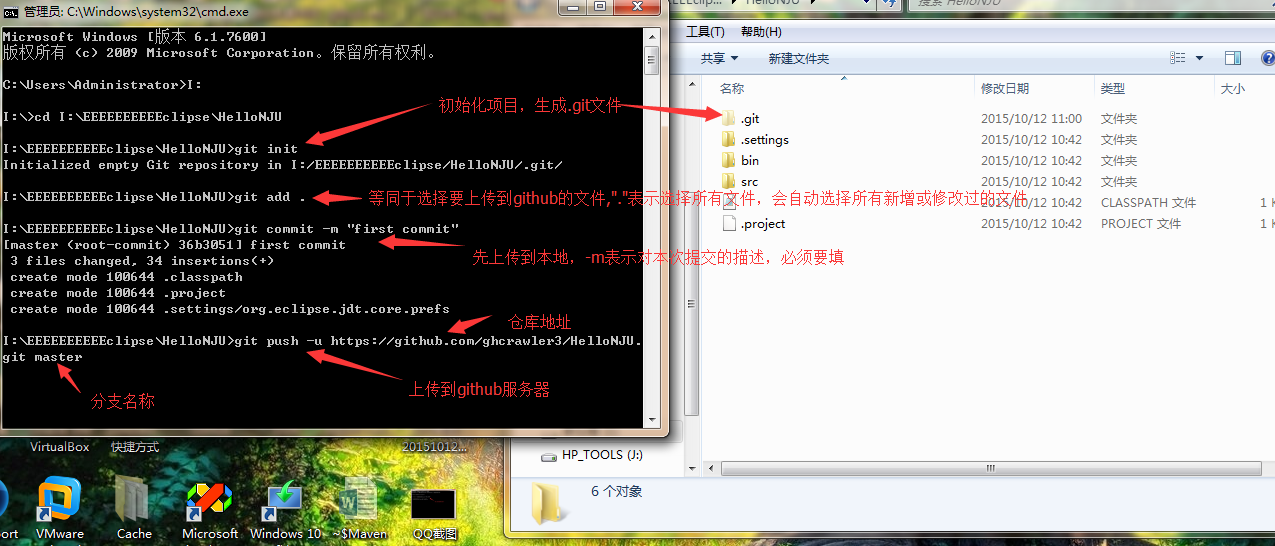
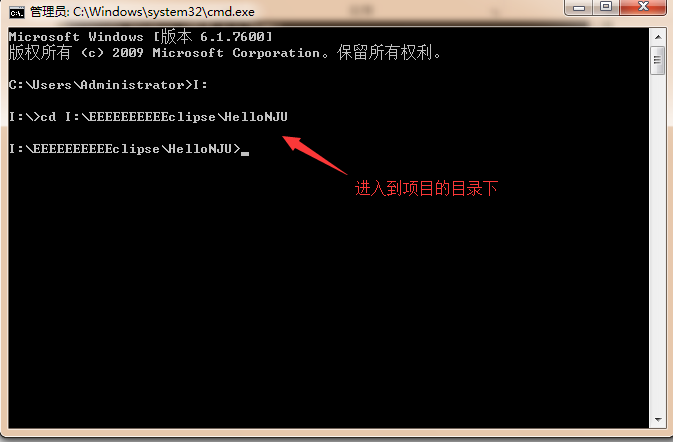
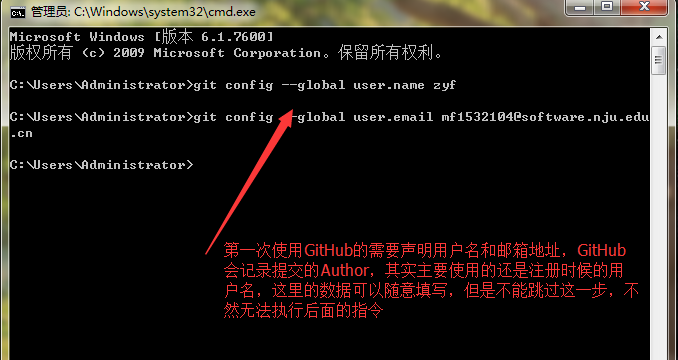
GitHub Desktop下载地址：<http://desktop.github.com/>（可视化工具）

## 1.2 命令行方式使用GitHub

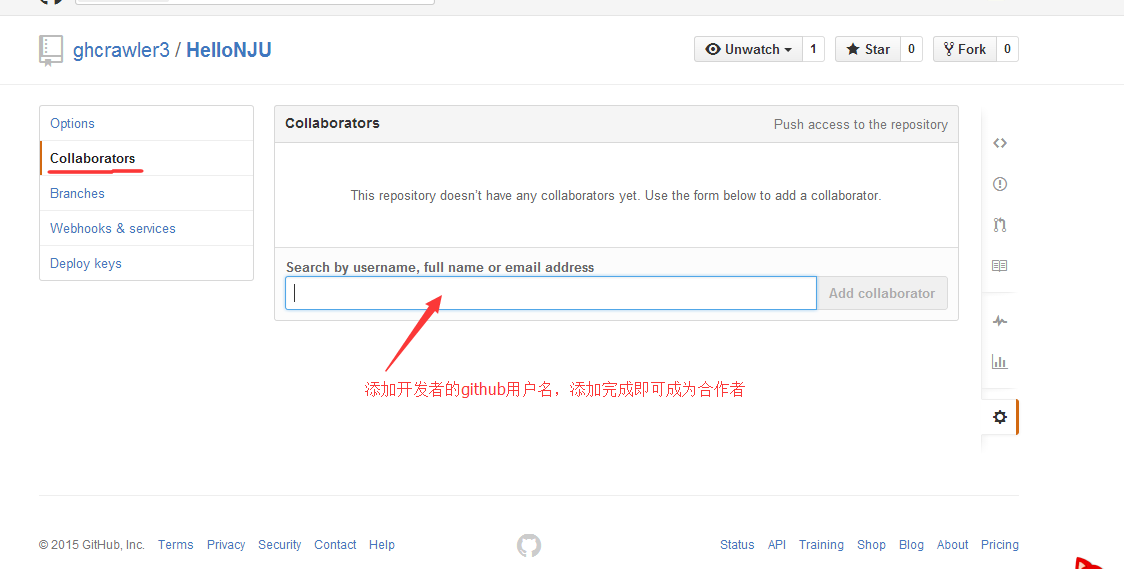
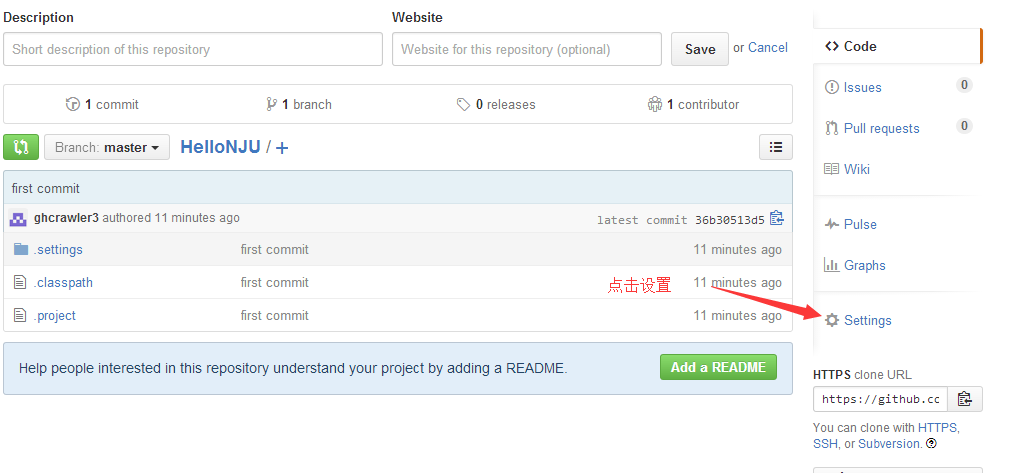
首先，进入<https://github.com/>，在GitHub官网创建一个GitHub账号。登录之后，如下图：



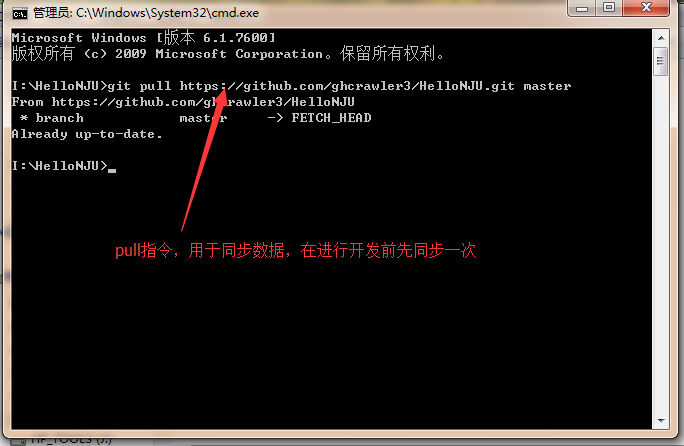
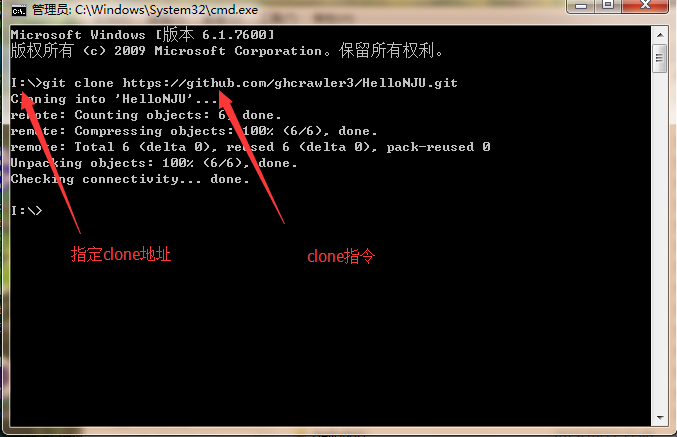
完成仓库的创建之后就要将本地的项目放入仓库之中，如图：



然后添加合作者，有多少个开发人员就添加多少个合作者，如果不添加合作者则每次都要pull request，经过项目所有者同意之后才能并入分支中，在这里不加多描述，只需添加合作者就没有这些麻烦，如图：



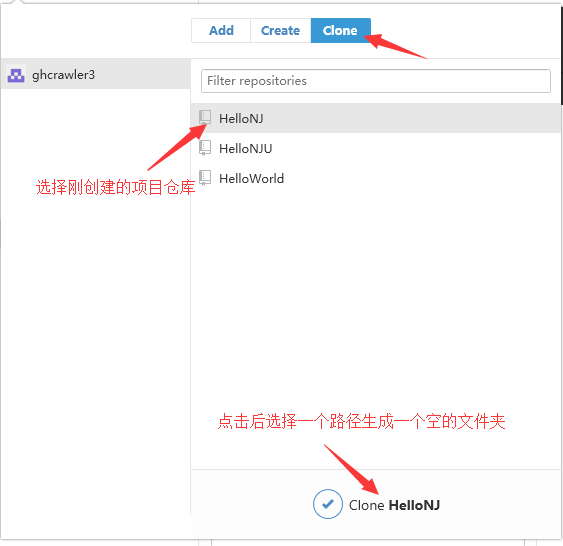
合作开发者先clone项目所有者的项目，再进行开发更新提交，如图：



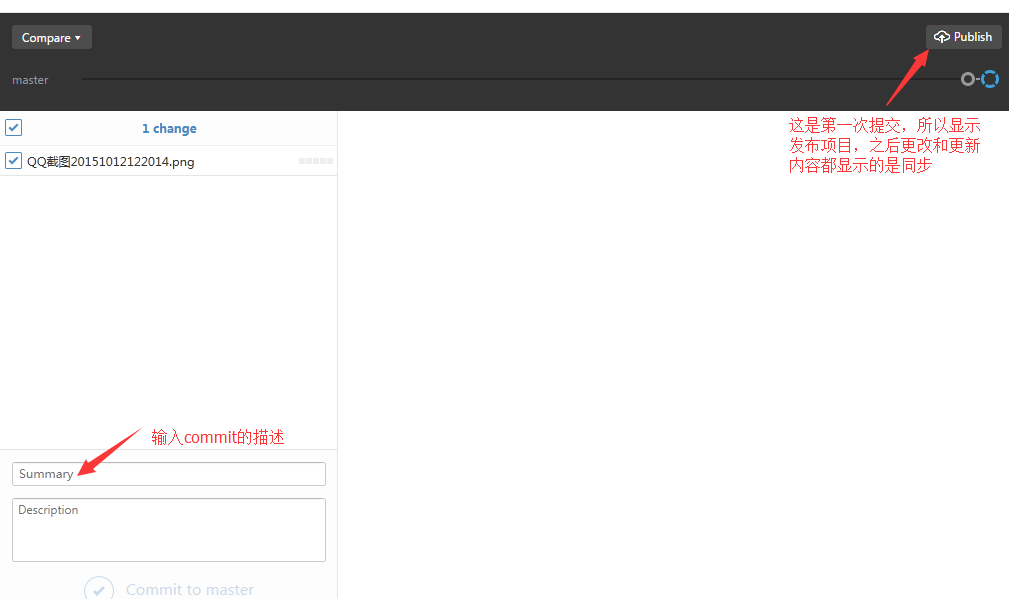
提交指令之前已经加以描述过，这里不再描述。

## 1.3 使用图形界面GitHub Desktop

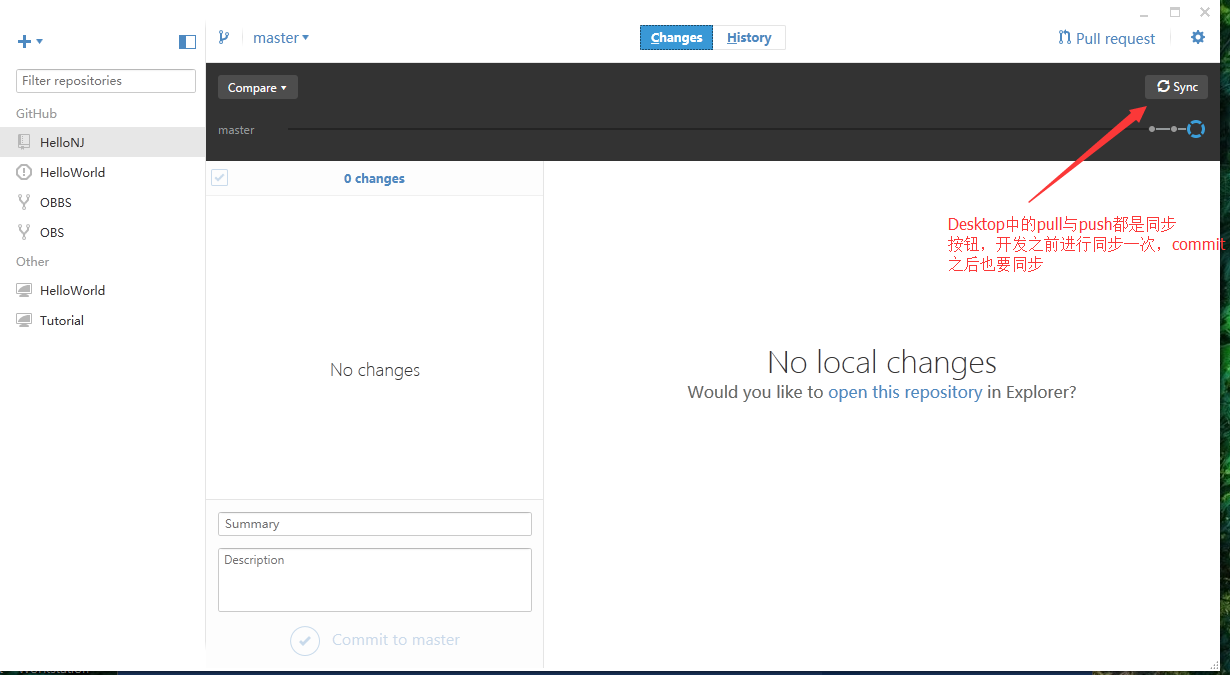
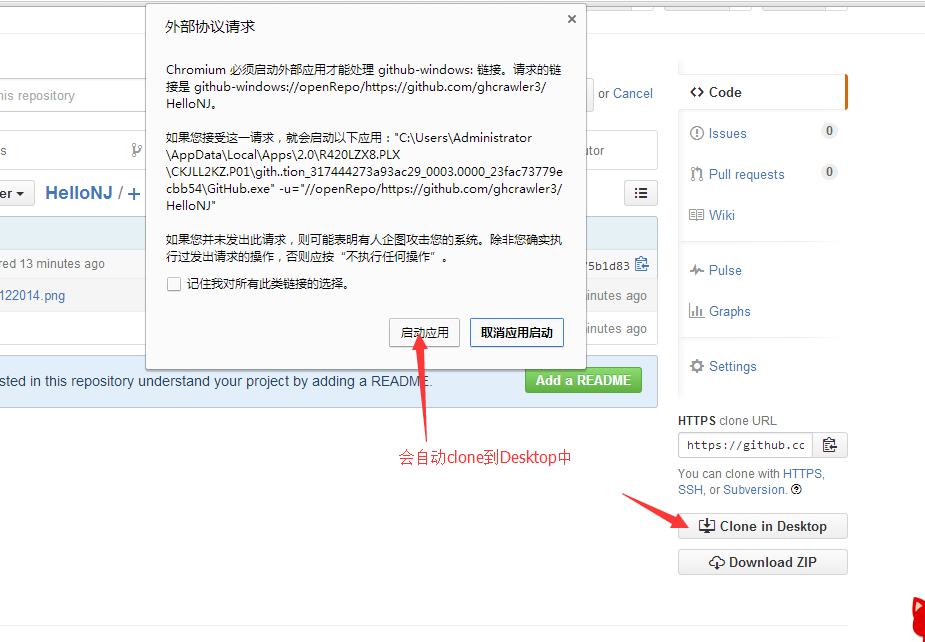
GitHub Desktop集成了命令行操作，用按钮等一系列可视化组件替代了git指令。使用时首先和之前一样，在github网页上新建一个仓库，然后如图：



然后将你的项目复制进文件夹中，你可以在GitHub Desktop中看见如下所示：



同样的，项目所有者先将其他开发者都设为合作开发者，合作开发者在项目主页上clone项目，如图所示：



# SVN

## 2.1 SVN简介

SVN是Subversion的简称，是一个开放源代码的版本控制系统。在团队开发过程中，确保由不同人所编辑的同一文件都得到更新，并且记录每次改动的内容，方便维护与回滚。SVN的使用需要一个固定的服务器，可以选择使用网上付费的服务器，也可以使用自己的电脑来搭建一个SVN服务器。网上付费服务器优点在于它有一个固定的IP访问地址并且随时可以访问，但是购买一台服务器的花费较大。自己搭建服务器开销较小，但是使用较为繁琐。作为服务器的主机的IP地址可能发生变化，访问地址也会随之改变，并且为了让其他开发人员随时访问，主机服务器也必须一直保持开启状态。

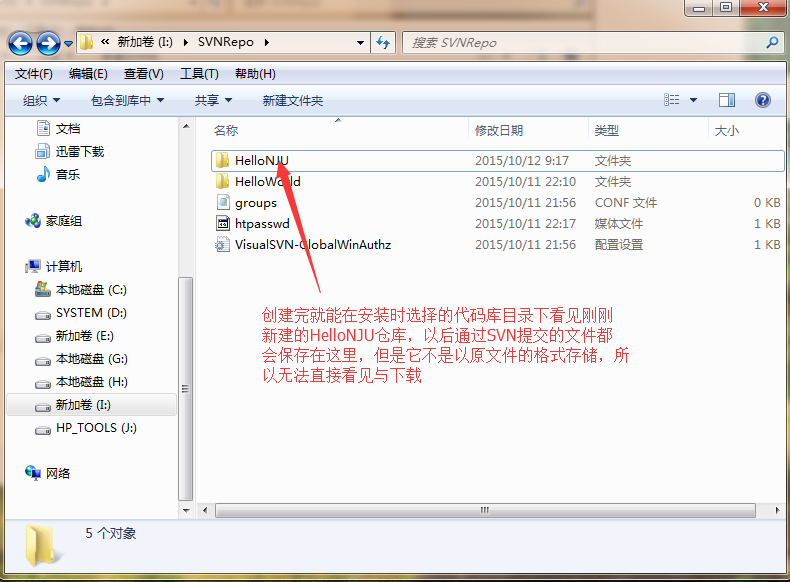
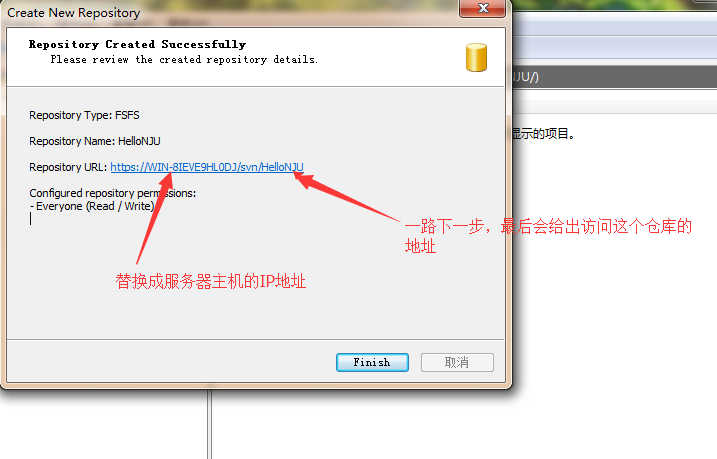
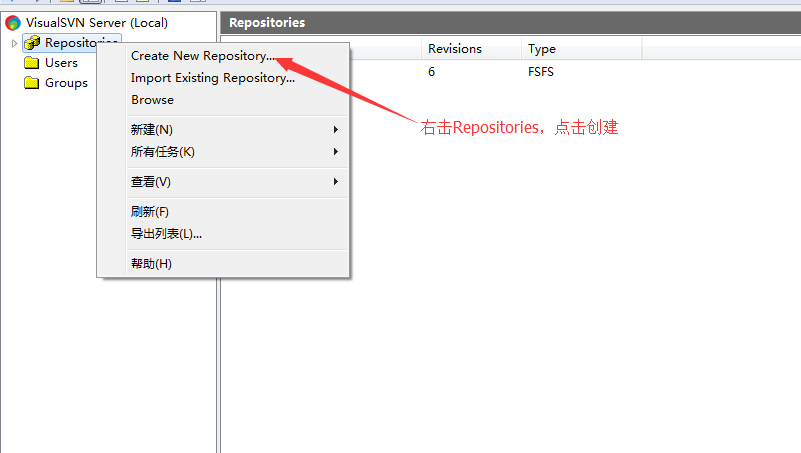
## 2.2 下载SVN

VisualSVN Server下载地址：<https://www.visualsvn.com/server/download/>（需要搭建SVN服务器的下载，其他人无需下载）

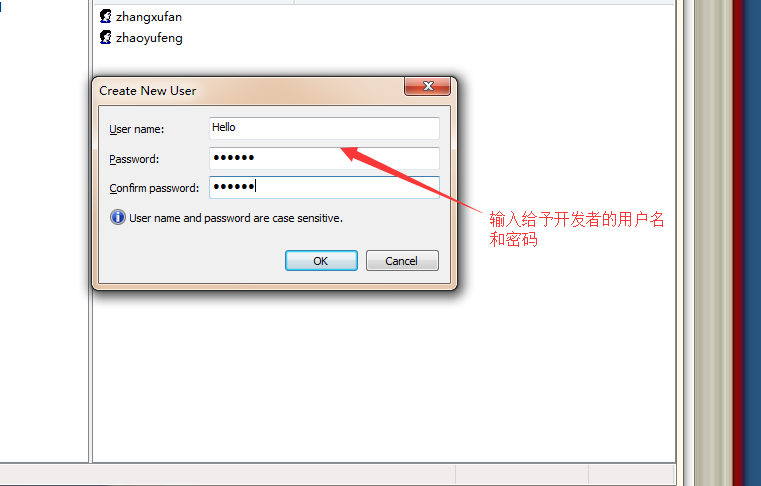
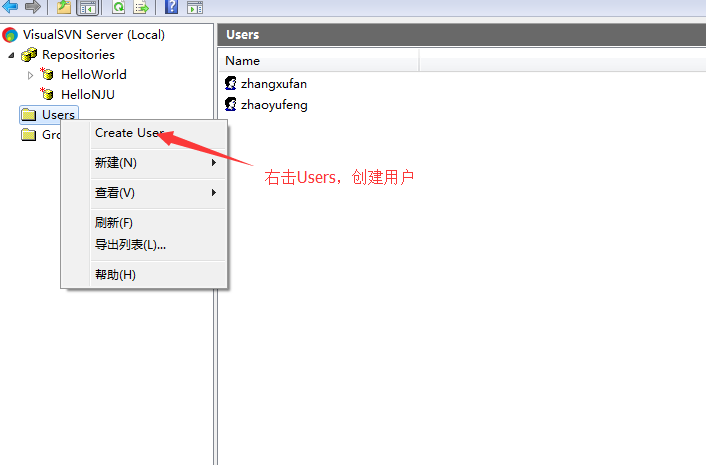
TorToiseSVN下载地址：<http://tortoisesvn.net/downloads.html>

## 2.3 VisualSVN Server配置与使用

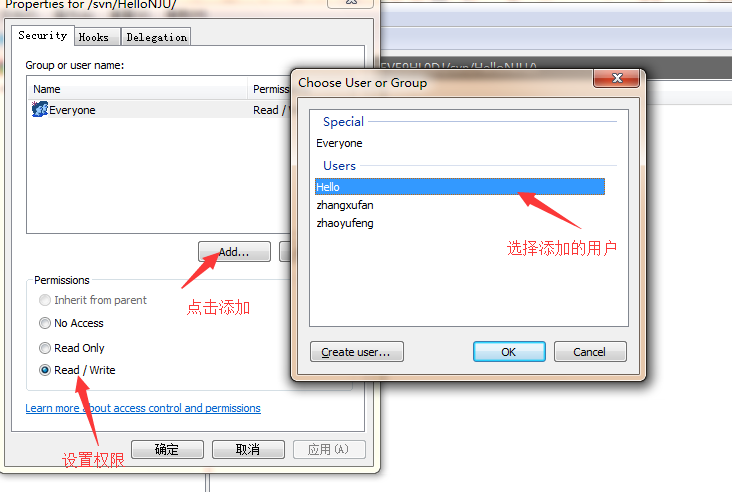
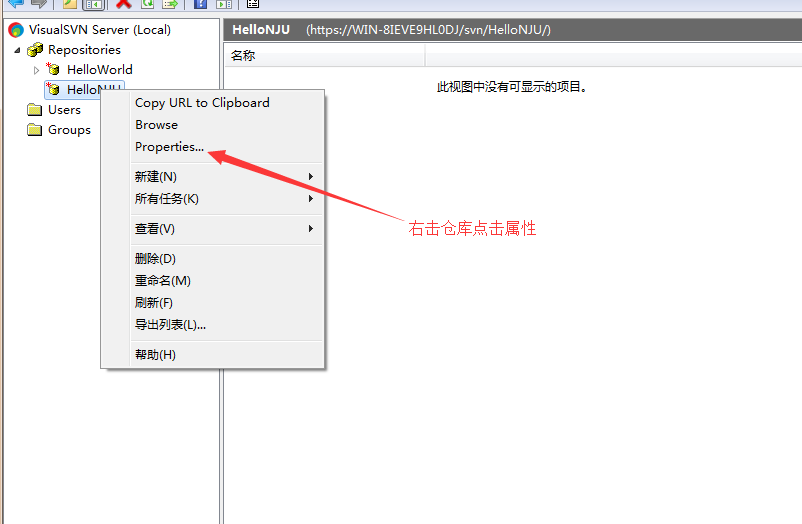
步骤一：新建一个repository，如图：



步骤二：创建用户，有多少个开发者就创建多少个用户，如图：

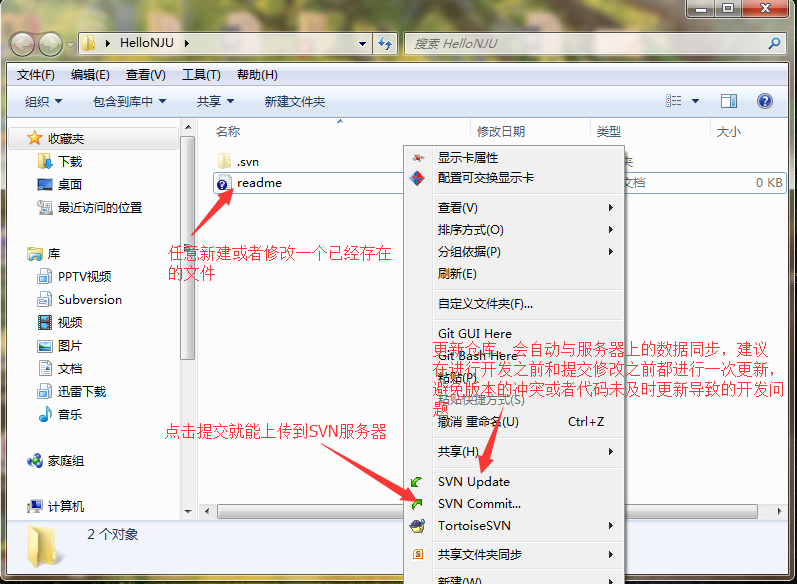
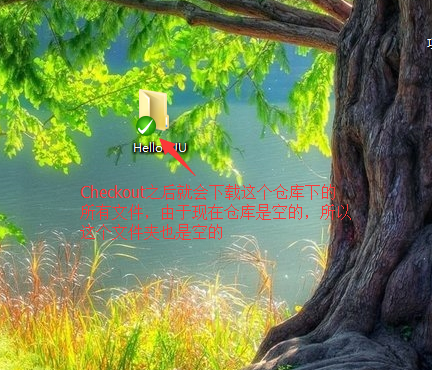
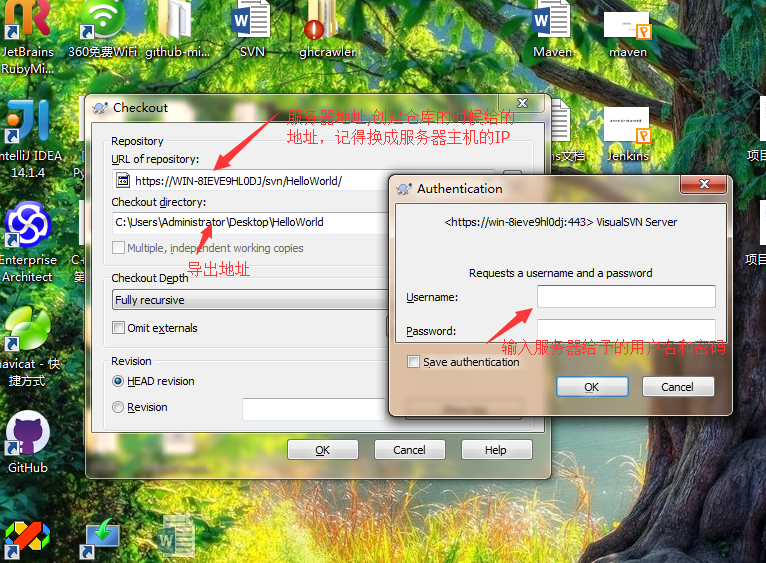
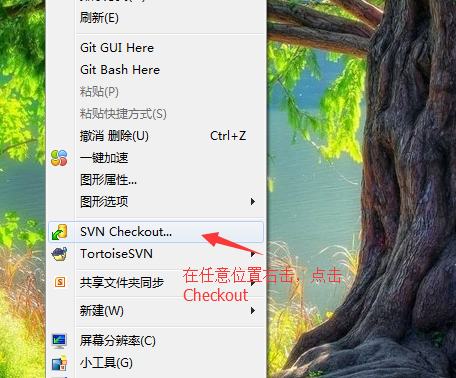


步骤三：将开发者与项目仓库绑定，如图：



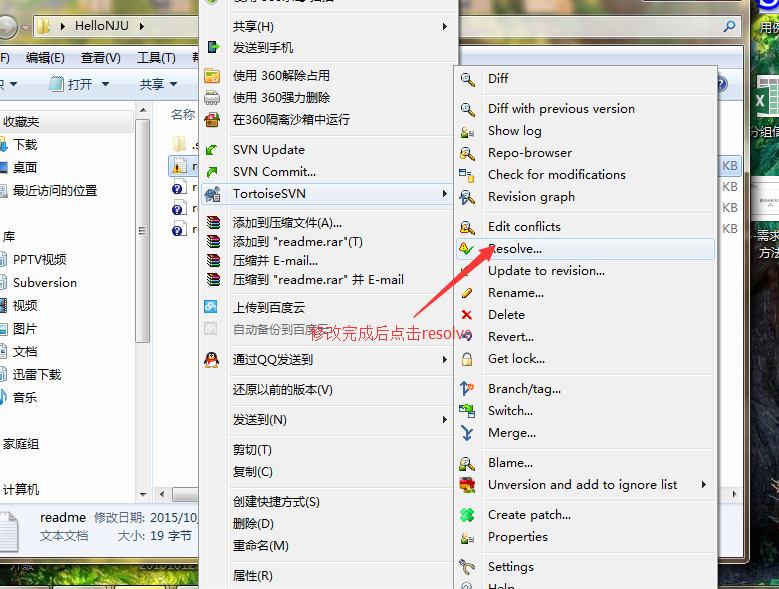
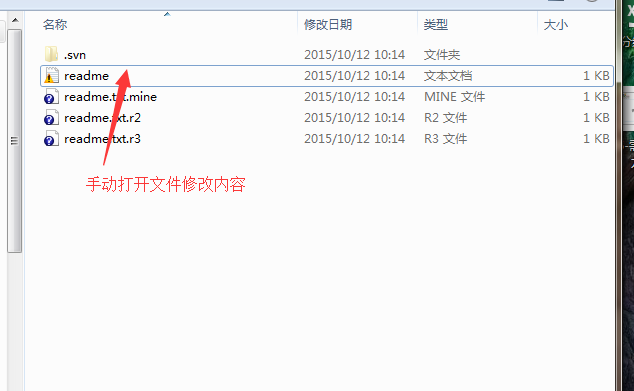
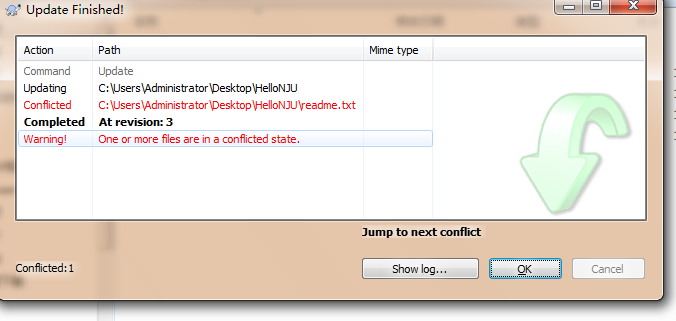
到这里在SVN服务器上的配置就完成了。

## 2.4 SVN的使用



## 2.5 SVN冲突

当两个开发者同时修改同一个文件并提交时就会有冲突产生。冲突只能通过手动来解决。如图：



最后再commit一下就解决了。

# Maven

## 3.1下载maven

* 地址：<http://maven.apache.org/download.cgi>

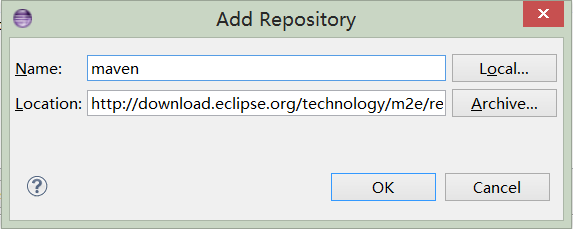
## 3.2 安装Maven

* 解压安装文件到指定目录中，例如“D:\bin\apache-maven-3.2.3”
* 配置环境变量，在系统变量中新建一个变量，变量名为“MAVEN\_HOME”，变量值为刚刚解压的目录，如“D:\bin\apache-maven-3.2.3”，然后在用户变量中的“PATH”变量中，在变量值的末尾加上%MAVEN\_HOME%\bin;，注意多个值之间需要有分号隔开。
* 检查安装，打开cmd，运行命令：mvn –v，运行结果如图：

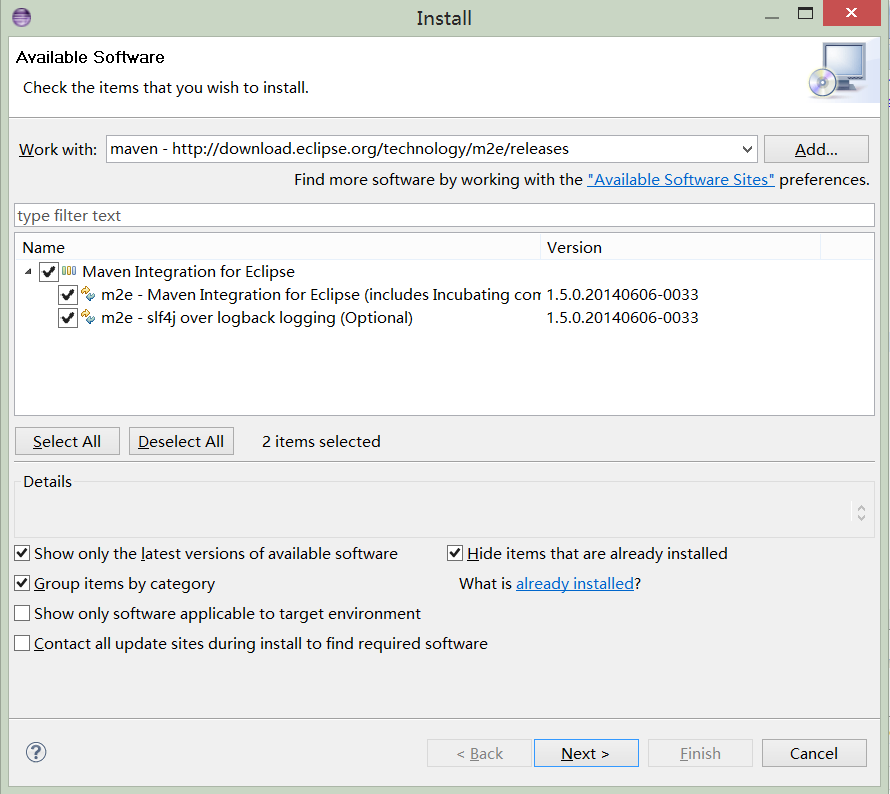


## 3.3 安装eclipse maven插件

* 打开eclipse，点击菜单 Help -> Install new software 进行插件安装。点击Add，输入插件地址，插件地址为：<http://download.eclipse.org/technology/m2e/releases>

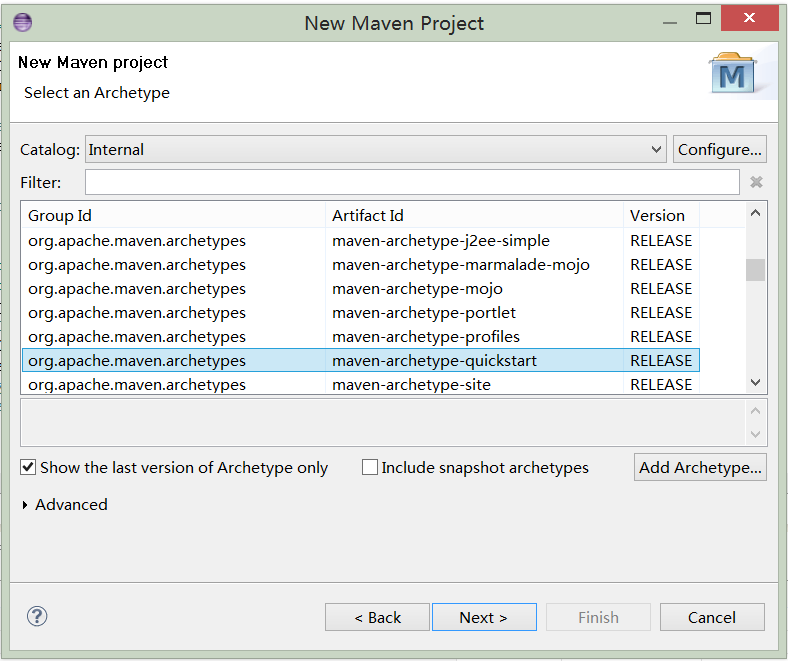


* 然后选中Maven插件开始安装（不要勾选contact all update sites选项，否则可能会需要安装很久）

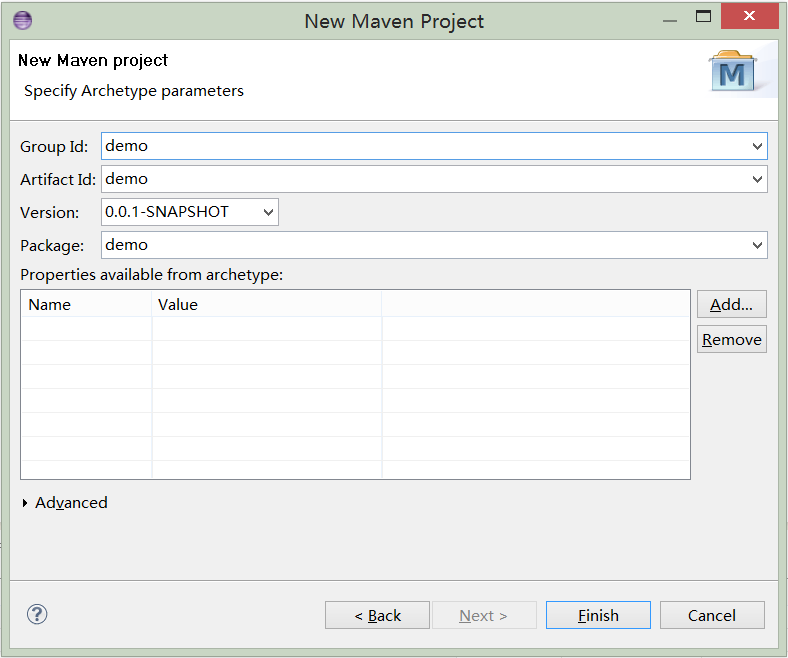


## 3.4 新建maven项目

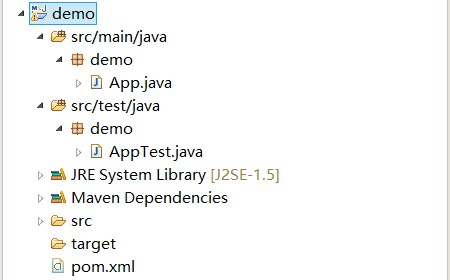
* 新建一个项目，在其他项目中选择Maven -> Maven Project
* 在Catalog选择Internal，Filter选择maven-archetype-quickstart，点击next



* 输入Group Id、Artifact Id、Version、Package名，点击Finish完成项目创建

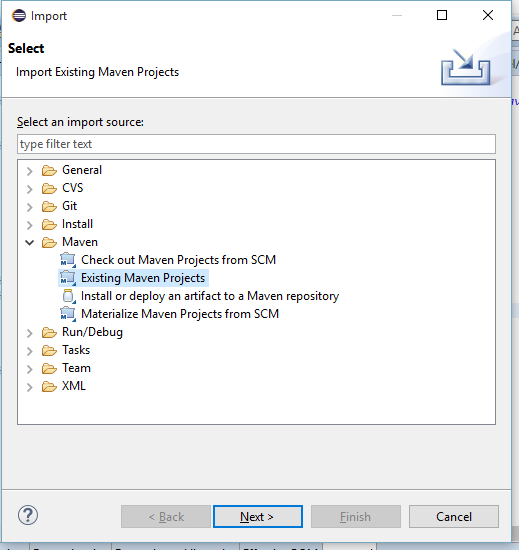


* 项目结构如图所示，可能会遇到Project configuration is not up-to-date with pom.xml. 的Error，通过右键项目->Maven->Update Project，然后选中出错项目确定即可

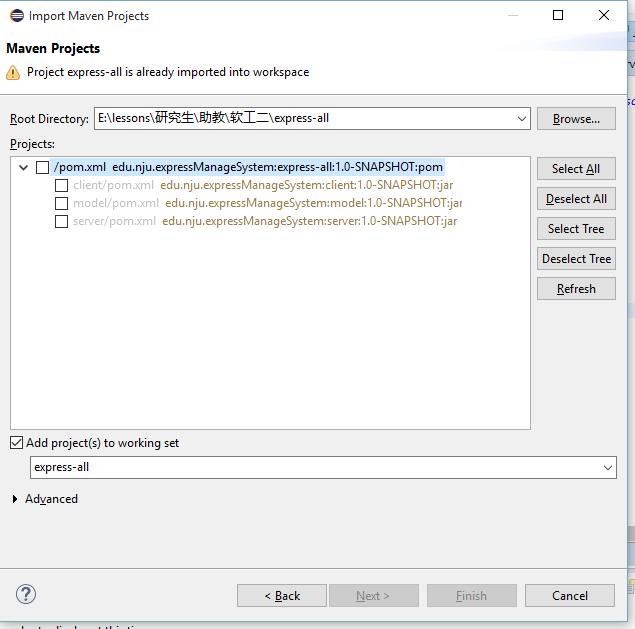


## 3.5 导入maven项目

打开eclipse，file->import->Maven->Existing Maven Projects，如下图所示：



选择maven project，导入全部，如下图所示。



## 3.6 配置pom.xml

在pom.xml配置需要在build时执行的单元测试：

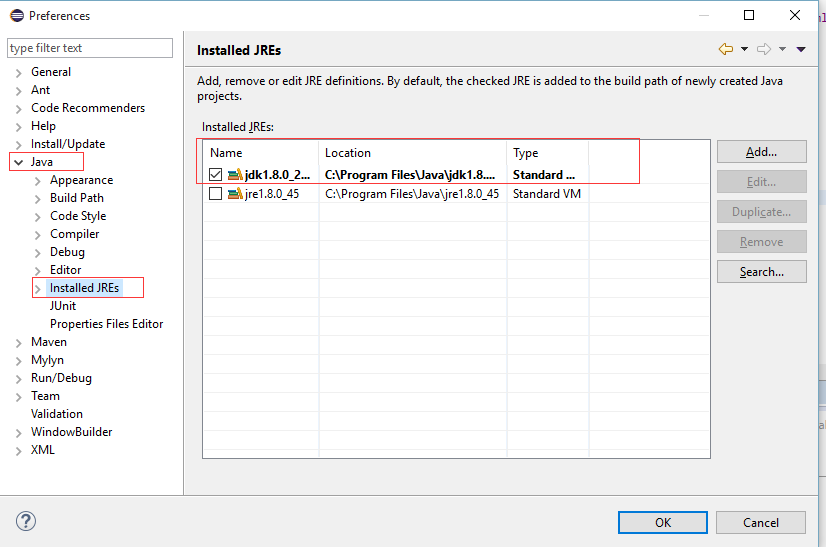
|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>demo</groupId>  <artifactId>demo</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>jar</packaging>  <name>demo</name>  <url>http://maven.apache.org</url>  <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  </properties>  <dependencies>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.8.2</version>  <scope>test</scope>  </dependency>  </dependencies>      </project> |

## 3.7 运行Maven

### 3.7.1 在eclipse中运行

* 右键项目 -> Run as -> Maven Test，可以在本地进行Maven的测试
* 右键项目 -> Run as -> Maven Install，可以在本地使用Maven进行编译
* 若提示应该使用JDK而非JRE，应该在eclipse中做如下配置：

在Window->Preferences->Java->Installed JREs中添加jdk的路径，并选中。



### 3.7.2 在命令行运行

* 打开命令行进入项目目录，执行mvn install，即使用maven进行编译。注意到install的阶段在test后面，因此执行mvn install的过程中也会执行mvn test进行测试。如果仅想编译以及打包而不想进行单元测试，可以运行mvn install -DskipTests
* 配置好Git和Jenkins之后，每次提交新版本都会自动进行一次这样的Maven Install

# Jenkins

## 4.1 Jenkins介绍

**持续集成**

持续集成（Continuous integration）简称CI，是一种软件开发的实践，可以让团队在持续集成的基础上收到反馈并加以改进，不必等到开发的后期才寻找和修复缺陷。当然要明白的是持续集成环境的搭建也不是一劳永逸的，随着软件项目复杂度的增加，持续集成的环境同样要加以维护以确保集成环境的可靠性。

持续集成的重要组成部分：（1）统一的代码版本控制工具（SVN，git），（2）CI工具（Hudson，Jenkins），（3）项目管理和自动化构建工具（Ant，Maven，Gradle）。

持续集成的流程：CI工具控制持续集成的整个过程，询问代码库更新，根据预定义的脚本进行项目的构建。

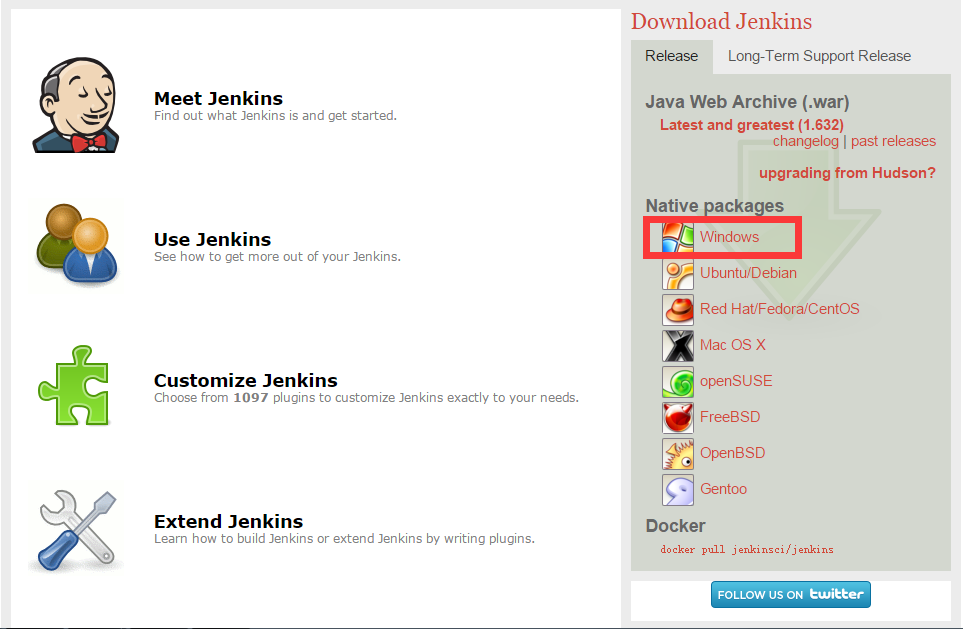
**why Jenkins？**

Jenins是现在非常流行的持续集成CI服务器，这与它的前身Hudson也有着很大的关系，Jenkins易于安装，不需要数据库的支持，直接通过Web界面进行配置，而且集成了RSS/Email的通知机制，支持分布式构建，具有丰富的插件，这些都是Jenkins相比其他持续集成服务器的优势所在。

## 4.2 安装与全局配置

官网：http://jenkins-ci.org/

1从官网下载对应平台的安装包。

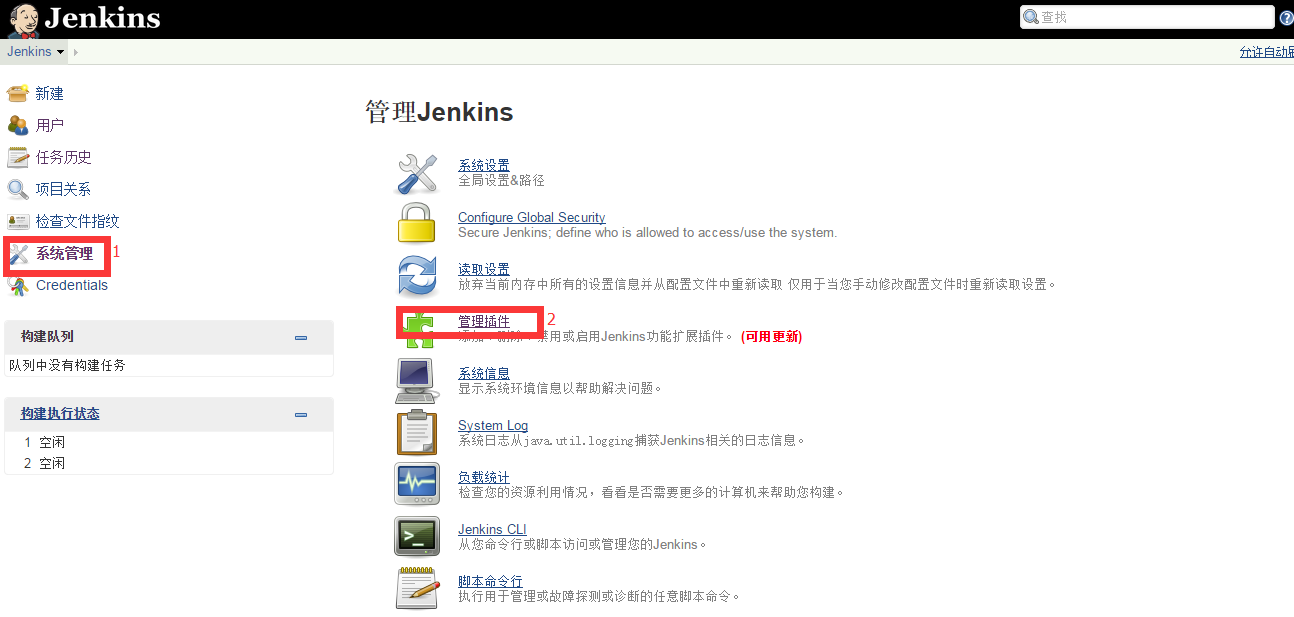


2通过安装包安装。

3在浏览器访问http://127.0.0.1:8080/，等待jenkins的初始化。



4系统管理->管理插件->可选插件，勾选GIT plugin和GitHub plugin并安装。



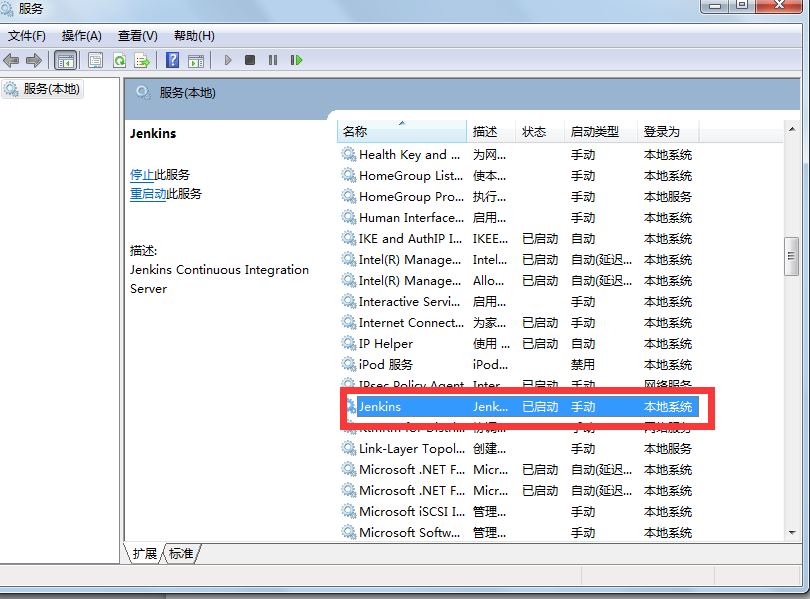
如果无法成功下载，可以直接将插件文件拷贝到**<jenkins根目录>/plugins**目录下。

5 插件安装完毕后需要重启Jenkins服务。

windows下开启任务管理器，选择服务选项卡，开启服务管理。



找到Jenkins服务，右键选择重新启动。



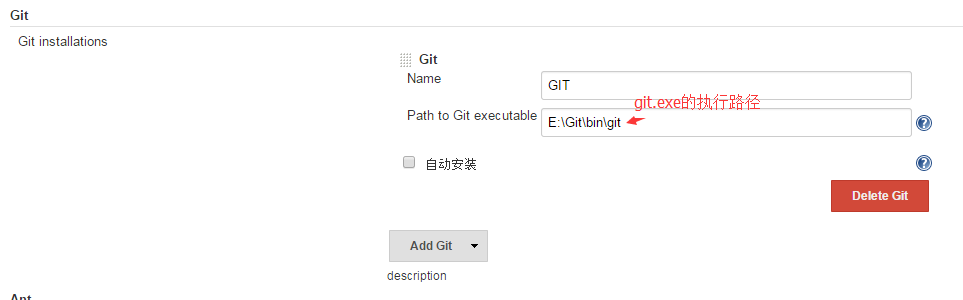
6待Jenkins服务重启后，在浏览器中重新访问http://127.0.0.1:8080/，进入系统管理->系统设置。



7配置JDK，不要勾选自动安装。



8配置Git。



9配置Maven，不要勾选自动安装。



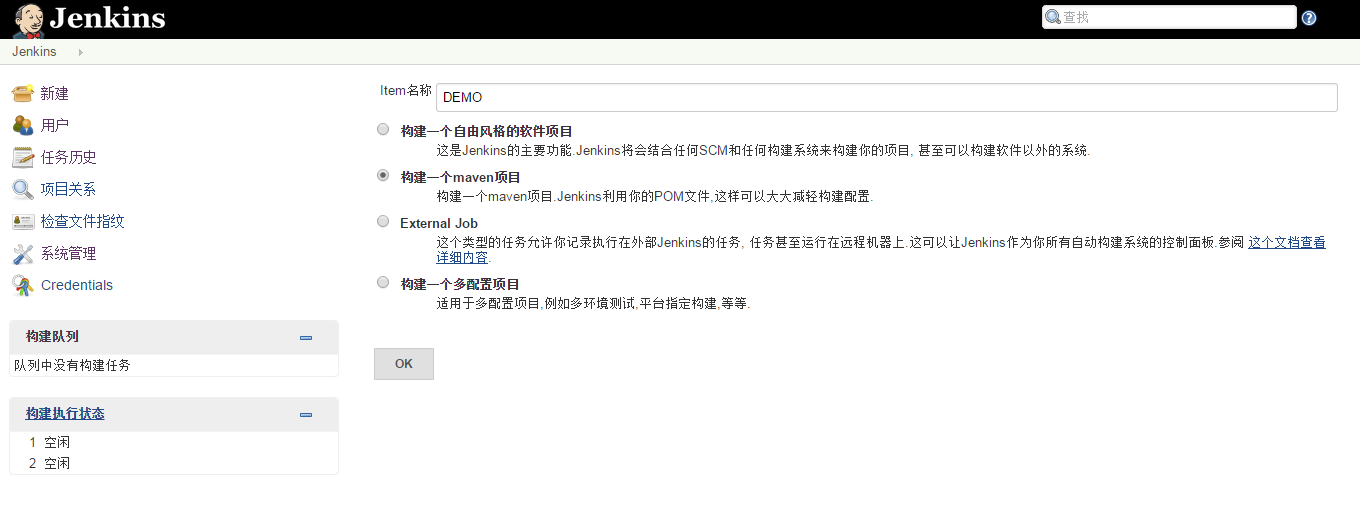
10配置完毕后点击保存按钮。

## 4.3 建立新的集成任务

1. 在浏览器中访问http://127.0.0.1:8080/，选择创建一个新任务。



2输入任务名称，选择构建一个maven项目，点击OK按钮。



3勾选丢弃旧的构建，填写适当的保持构建天数与保持构建最大个数。



4填写GitHub Project的URL。

5源码管理选择Git，填写对应的Repository URL和Git账户信息。

6根据实际情况创建前置步骤和后置步骤。

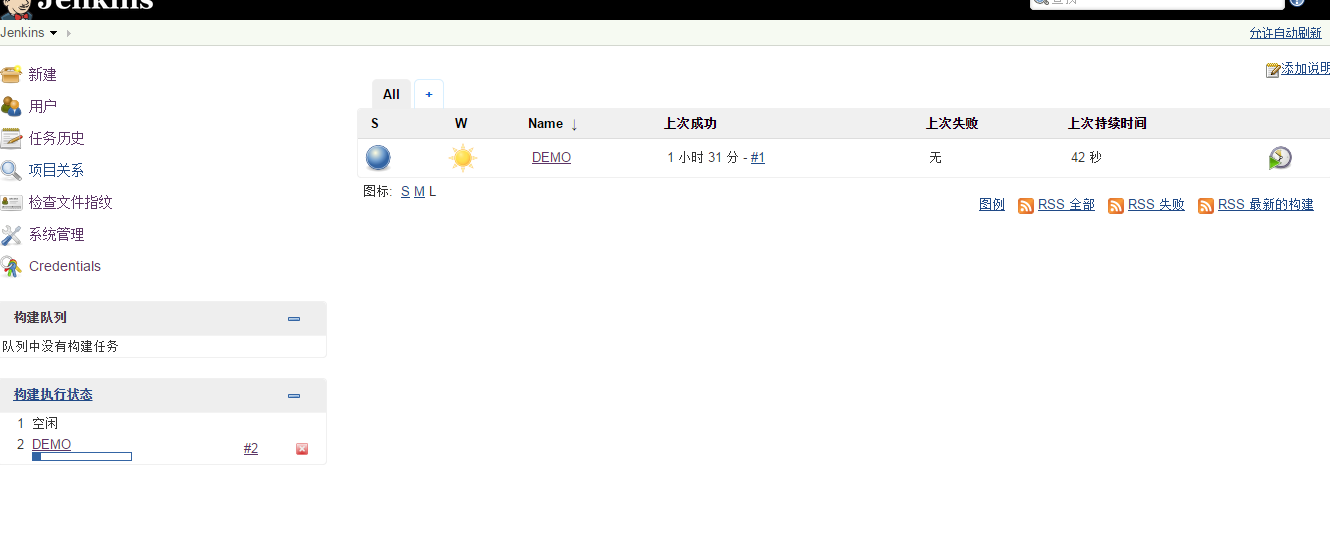
7配置完毕后点击保存按钮。

## 4.4 执行与调试集成任务

在主界面中，点击指定集成任务的执行按钮，就开始一次集成任务。



集成过程中可以在左下角看到任务进程。



如果任务成功，则对应任务的图标显示为蓝色；如果任务执行失败，则对应任务图标显示为红色。

点击任务名称，可以看到构建历史信息。



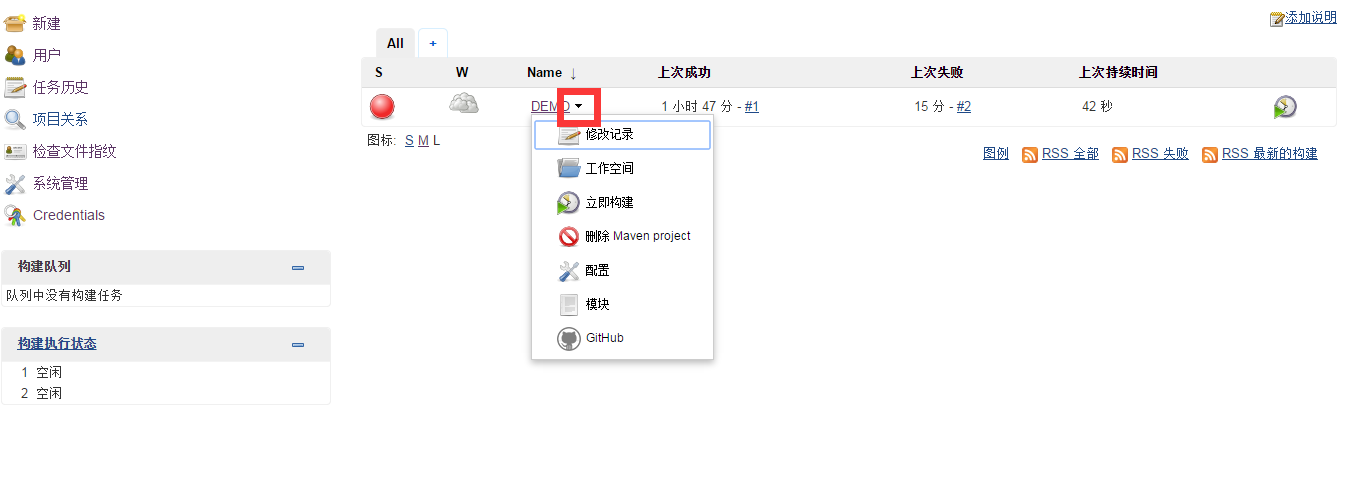
点击某个构建记录，选择Console Output，可以看到执行的相关指令，从而进行调试。



所有构建后的产物都存在于<jenkins根目录>/jobs/<任务名>/workspace下。

## 4.5 修改、删除集成任务

点击相应任务名称后的倒三角按钮，选择指定的操作。



# Junit

在express-demo-master的pom.xml中配置

|  |
| --- |
| <dependencyManagement>  <dependencies>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.12</version>  <scope>test</scope>  </dependency>  </dependencies>  </dependencyManagement> |

在每个module的pom.xml中配置

|  |
| --- |
| <dependencies>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  </dependency> |

运行mvn test命令即可。

# Checkstyle

CheckStyle项目提供了一个帮助JAVA开发人员遵守某些编码规范的工具。它能够自动化代码规范检查过程，从而使得开发人员从这项重要但是枯燥的任务中解脱出来。CheckStyle检验的主要内容: Javadoc注释、命名约定、标题、Import语句、体积大小、空白、修饰符、块、代码问题、类设计、混合检查等，主要是用来提高代码可读性，提高项目的质量，易于维护。

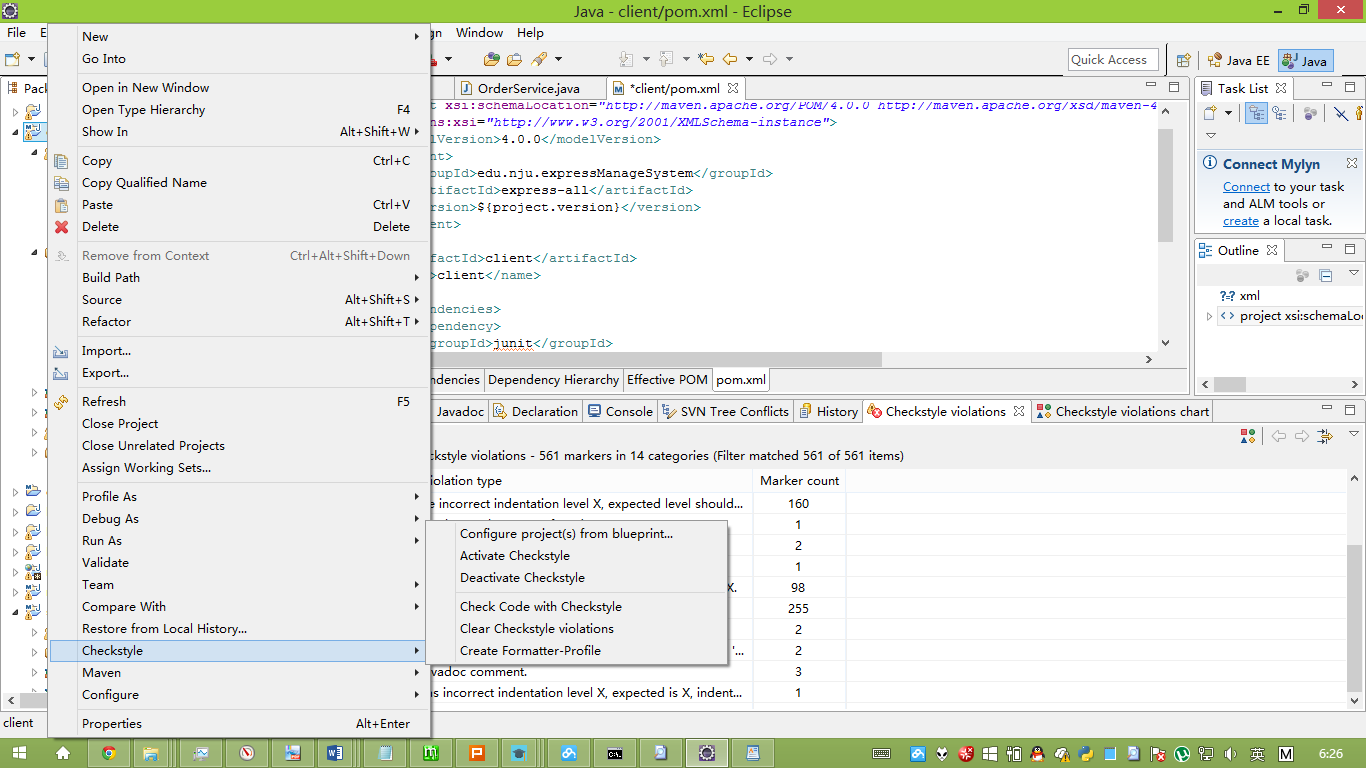
## 6.1安装eclispe插件

eclipse菜单 Help → Install new software

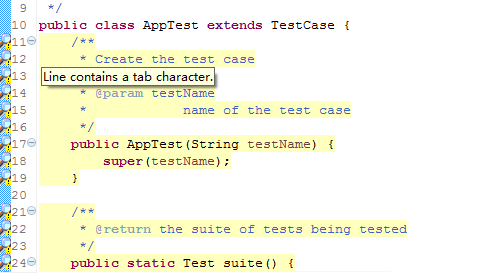
插件地址 <http://eclipse-cs.sourceforge.net/update/> ，安装结束后重启eclipse。

## 6.2使用Checkstyle

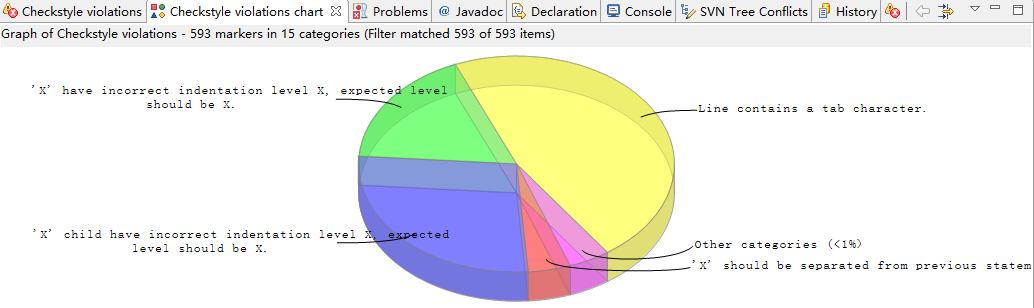
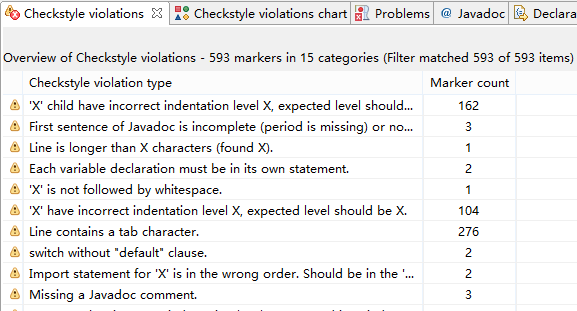
安装完成后，选中类文件、包或工程通过右键菜单Checkstyle→check code with checkstyle检查错误。



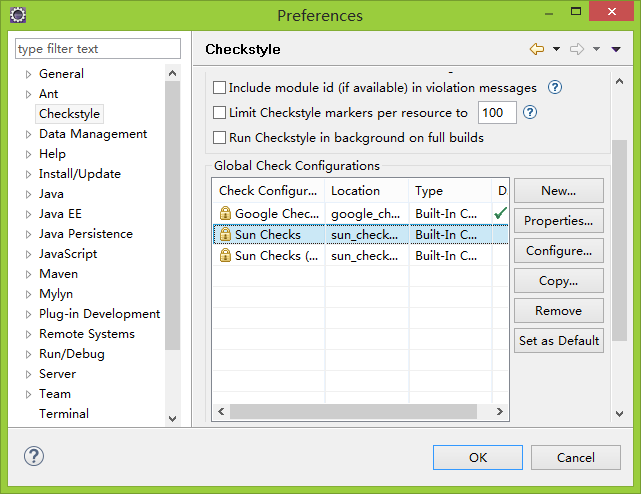
检查完成后，通过左侧放大镜图标查看信息。



统计信息。



自定义配置文件



了解了Checkstyle提供的检查项以后，我们可以依据其提供的sun\_checks.xml作为模板来定制复合项目需要的检查项。在Window → preference的Checkstyle中配置检查项或导入自定义配置文件。

## 6.2使用Maven时Checkstyle检查

项目地址 <http://maven.apache.org/plugins/maven-checkstyle-plugin/>

1. 在项目pom.xml中填入maven的plugin配置

|  |
| --- |
| <reporting>  <plugins>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-checkstyle-plugin</artifactId>  <version>2.16</version>  <reportSets>  <reportSet>  <reports>  <report>checkstyle</report>  </reports>  </reportSet>  </reportSets>  </plugin>  </plugins>  </reporting> |

1. 执行maven checkstyle:check进行代码风格检查

checkstyle插件的可执行任务如下：

mvn checkstyle:help

查看checkstyle-plugin的帮助

mvn checkstyle:check

检查工程是否满足checkstyle的检查，如果没有满足，检查会失败，可以通过target/site/checkstyle.html查看。

mvn checkstyle:checkstyle

检查工程是否满足checkstyle的检查，如果没有满足，检查不会失败，可以通过target/site/checkstyle.html查看。

mvn checkstyle:checkstyle-aggregate

检查工程是否满足checkstyle的检查，如果没有满足，检查不会失败，可以通过target/site/checkstyle.html查看。