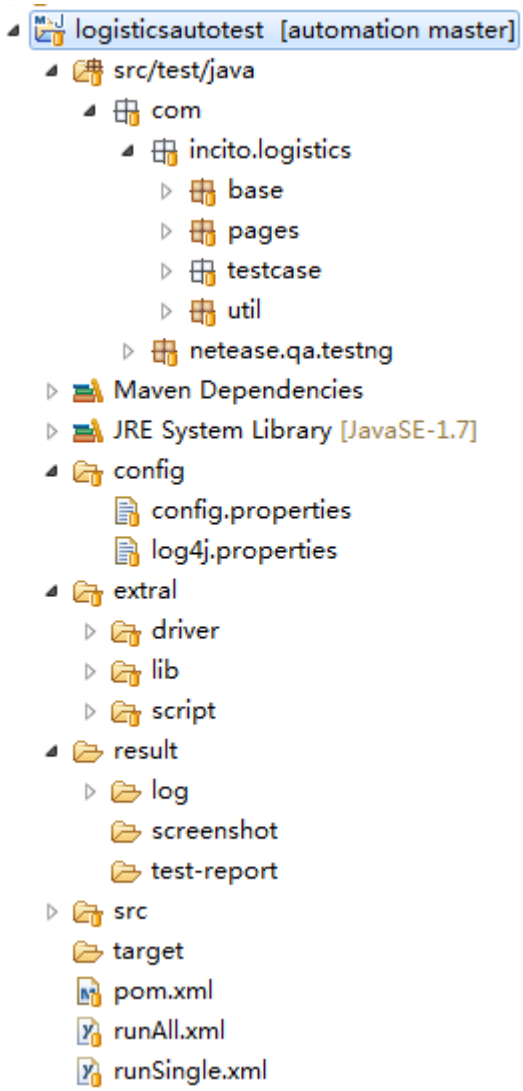
web 自动化测试框架介绍与使用

一、框架介绍篇

web 自动化测试框架是基于 selenium + maven + testng + github +Jenkins 搭建的，其中 selenium 用来编写测试用例，maven 用于项目的构建，testng 用于执行测试，相当于执行者，github 用于代码的托管，jenkins 用于持续集成。整个框架可实现用例的自动执行测试、失败的用例自动重试、失败的用例自动截图，加之整个框架集成了 jenkins,可使所有功能模块的测试用例定时并持续的集成下去。最后的测试报告可以通过 jenkins 邮件服务系统发送给相关人员。 在此着重介绍下 selenium，selenium 是一个用于 Web 应用程序测试的工具。Selenium 测试直接运行在浏览器中，就像真正的用户在操作一样。支持的浏览器包括 IE(7、8、9、10、11)、Mozilla Firefox、Google Chrome 等。由于是 JAVA 语言编写，所以具有跨平台性，支持 windows、Linux 和 MAC OS 等。Selenium 是 ThoughtWorks 专门为 Web 应用程序编写的一个验收测试工具。
注：本框架中用到的工具都是开源的，不涉及到收费工具，故不需要担心版权问题。

二、目录解析篇



项目整体是一个 maven 项目，因为 maven 可以自动帮助我们构建项目、自动下载我们所需要的依赖（jar 包），只需要在 pom.xml 文件下配置好相关数据即可。

- src/test/java：maven 项目中存放测试代码的目录
  - com/incite/logistics：物流项目目录
    - base：里面有个 BaseParpare.java 用于初始化浏览器和结束浏览器的操作
    - pages：存放每个页面上的元素的 java 类（每个页面就是一个 java 类，里面存放着声明此页面上要用到的所有元素）
    - testcase：存放测试用例的目录，此目录下按照功能模块再细分目录（模块），每个子目录都是各个模块的测试用例
    - util：存放逻辑操作的代码目录，此目录下目前有 2 个 java 类：SelectExplorer.java 和 SeleniumUtil.java
      - SelectExplorer.java：顾名思义，这是一个浏览器选择的逻辑操作类
      - SeleniumUtil.java：包装了 selenium 所有常用的方法包括自定义的方法
  - com/netease/qa/testing：这个是网易的开发的一个基于 testng 的插件，名字叫做：arrow，此插件可以实现的功能有：用例失败自动重试、失败用例会自动截图，生

成的测试报表比 testng 自带的更直观，自动去重结果（比如第一次用例跑失败了，但是经过第二次重试，用例成功执行，arrow 就会把第一轮失败的记录移除，保留第二次正确的测试记录）

- config：存放 log4j 的配置文件（log4j.properties）和 arrow 插件的配置文件（config.properties）,关于 log4j 的配置网上比比皆是，可以参考这里，这里介绍下 arrow 的配置：

  config.properties：testng 插件 arrow 的配置文件,内容如下：

  retrycount=n //定义重跑次数，就是用例跑失败之后，再跑 n（n 是正整数）遍

  sourcecodedir=src/test/java/com/incito/logistics/testcase/ //指定测试用例的目录

  sourcecodeencoding=UTF-8 //指定源码的字符编码

- extral：存档驱动，类库和第三方脚本的目录

  driver：存放不同平台下的不同位数不同浏览器的的 driver

  lib：里面存放的备用 jar 包，比如有的 jar 包在 maven 中央仓库找不到（自己编写的），就只能存档，便于以后直接导入使用

  script：存放第三方的脚本辅助测试

- result：存放测试结果的目录

  log：存放 log4j 的输出日志，只有运行了测试用例才会生成

  screenshot：存放失败的用例的网页截图

  test-report：测试报告生成目录，以 HTML 的方式显示，如图所示，附上一份完整的测试报表：

  测试报告.zip

Test	Methods Passed	Scenarios Passed	# skipped	# failed	Total Time	Included Groups	Excluded Groups
登录	6	6	0	2	57.6 seconds		

Class	Method	Authors	# of Scenarios	Running Counts
登录 — failed				
com.incito.logistics.testcase.login.LoginPage_8_EXIT_Test	exitLoginTest	xy-incito	1	1
com.incito.logistics.testcase.login.LoginPage_2_Success_Test	loginSuccessTest	xy-incito	1	1
登录 — passed				
com.incito.logistics.testcase.login.LoginPage_3_Fail_All_Empty_Test	loginFailTest_All_Empty	xy-incito	1	1
com.incito.logistics.testcase.login.LoginPage_5_Fail_Password_Test	loginFailTest_Password	xy-incito	1	1
com.incito.logistics.testcase.login.LoginPage_4_Fail_Username_Test	loginFailTest_Username	xy-incito	1	1
com.incito.logistics.testcase.login.LoginPage_7_Fail_Incorrect_Password_Test	loginFailTest_incorrectPassword	xy-incito	1	1
com.incito.logistics.testcase.login.LoginPage_6_Fail_Incorrect_Username_Test	loginFailTest_incorrectUsername	xy-incito	1	1
com.incito.logistics.testcase.login.LoginPage_1_UI_Test	uiTest	xy-incito	1	1

- pom.xml：maven 的配置文件，项目核心配置，用于构建项目、自动下载项目依赖以及后续的和 testng、jenkins 配合持续集成等
- runAll.xml：这是 testng 的配置文件，用于存放部分测试数据以及测试的平台，浏览器的配置、加入第三方插件监听（arrow 插件）、设置用例执行策略（多线程还是单线程，顺序执行还是无序执行以及是否依赖执行等）以及设置要执行的用例。之所以改成 runAll 顾名思义就是说执行整个 web 项目的所有模块的用例的测试

- runSingle.xml：和 runAll.xml 一样，不多解释，不一样的地方就是用它来做单个用例的调试改错，只涉及到单个的类（用例），所以调试改错专用。比如我用 runAll.xml 跑完了所有的用例，但是发现个别用例失败，此时先去分析 log，如果是代码问题就去调试代码，调试完毕之后就去用 runSingle.xml 去跑一边此用例。

三、如何使用篇

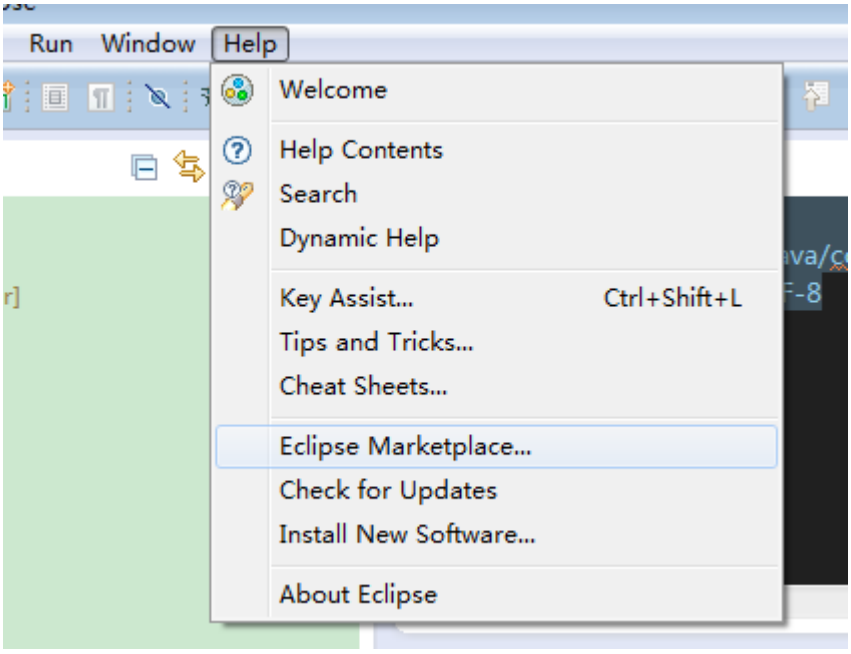
注：下面所讲到的是基于 windows 平台

3.1、安装配置 JDK

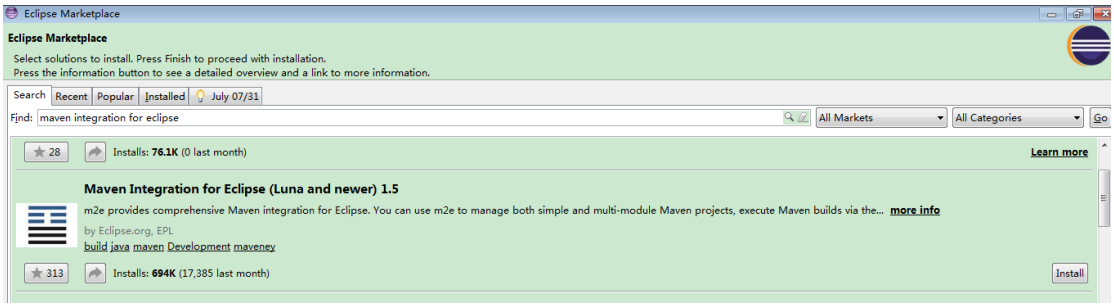
1. 下载安装 JDK 7(32bit)，点我下载
2. 配置 jdk 环境变量，点我查看如何配置

3.2、安装配置 Maven

1. 下载 maven,点我下载
2. 配置 maven 环境变量，点我查看如何配置
3. 下载安装 Eclipse（32bit）,点我下载
4. 解压刚才下载好的 Eclipse 到任意目录，双击 Eclipse.exe 打开
5. 安装必须的插件 maven integration for eclipse
6. 打开 Eclipse 菜单栏->Help->Eclipse Marketplace... 如图：



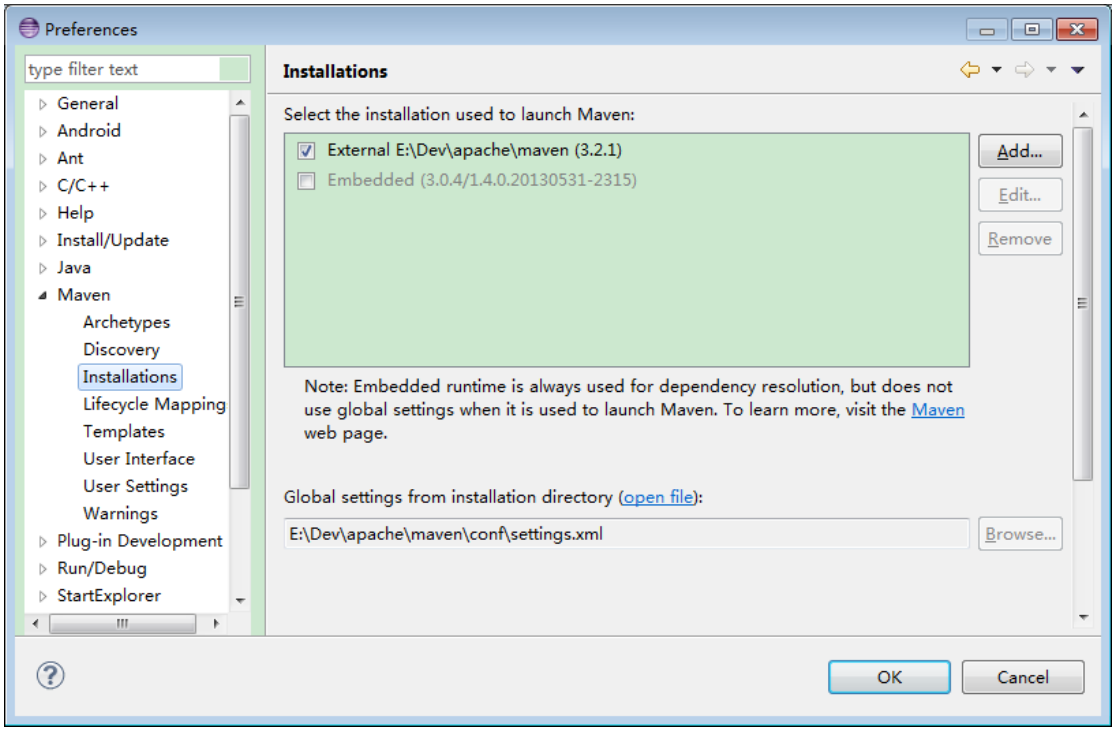
7. 搜索” maven integration for eclipse”安装，如图，点击 install 安装会重启 Eclipse



8. 重启之后在 Eclipse 菜单栏->Window->Preferences 点击打开，找到 maven 项，这个地地方需要配置 2 个地方（如图）：

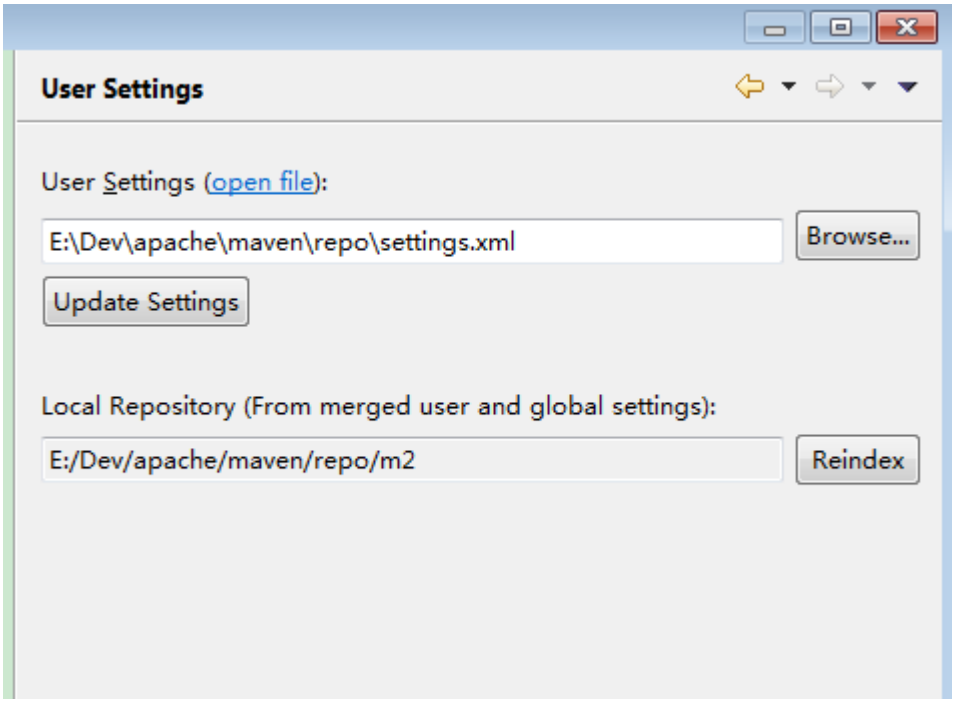
一、 Installations, 点击 Add，指向第三部中下载的 maven 的目录，比如我的是 e:\Dev\appache\maven(就是 maven home 路径)，指定成功以后 下图中的 Global Settings from installation diectory 会自动定位到 maven 所在路径下的 conf/settings.xml 文件（settings.xml 是 maven 的设置配置文件，它可以指定本地仓库的存放路径、可以指定远程中央仓库的地址），在这里提供下我自己的 settings.xml 文件：

里面已经配置了中央仓库为中国的服务器（开源中国的 maven 仓库）



二、User Settings： 用户设置（User Settings）这里会再次指定一个 settings.xml 这里是相当于局部变量，对当前用户适用，刚才前面的 settings.xml 是全局变量 针对所有用户。由于前面我提供的 settings.xml 指定了本地仓库的路径为：e:\Dev\apache\maven\repo\m2 这个是 maven 的目录下的，如果 repo\m2 目录不存在，请自建，然后把 settings.xml 文件放入 repo 目录 作为 user settings 的配置文件。如果想更改本地仓库目录，请修改刚才提供的 settings.xml 文件中的：

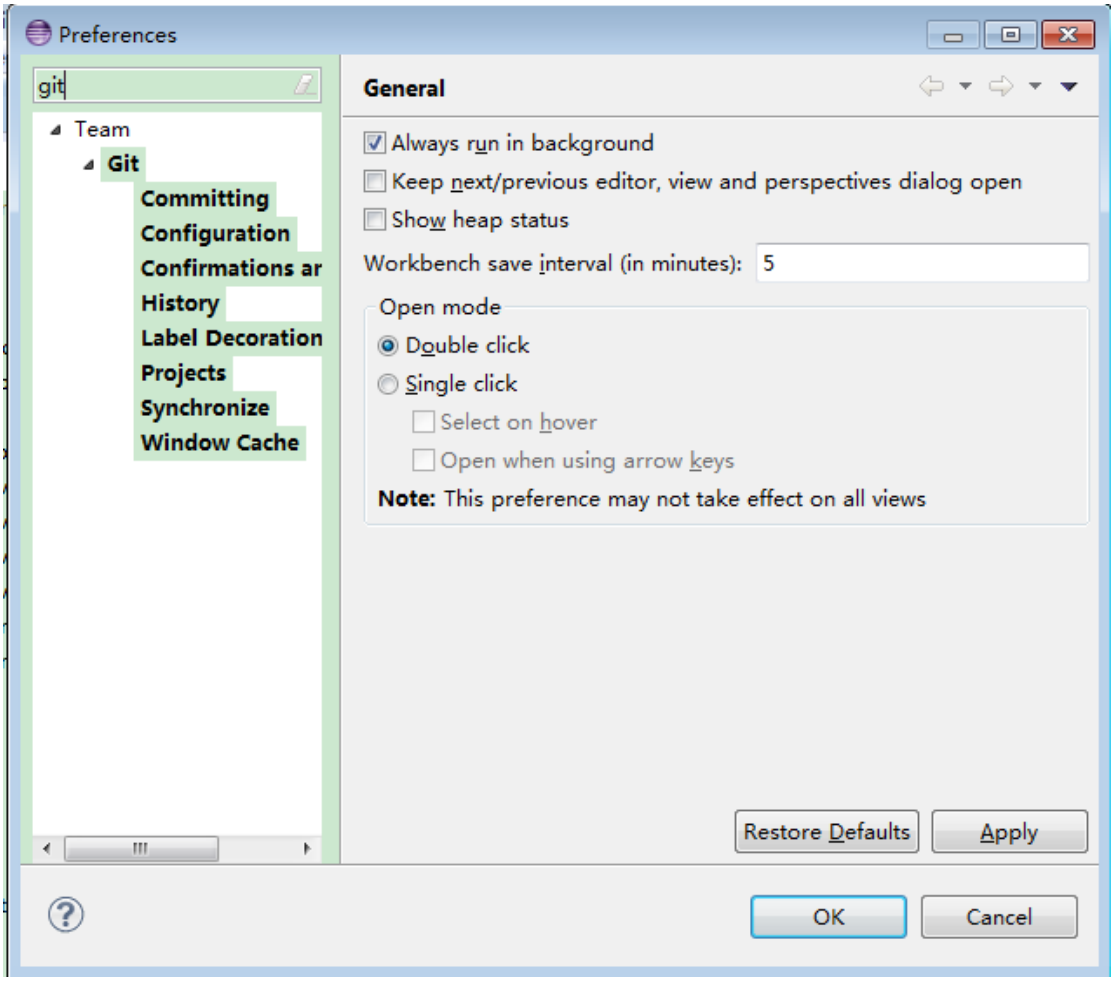
<localRepository>E:/Dev/apache/maven/repo/m2</localRepository> 即可



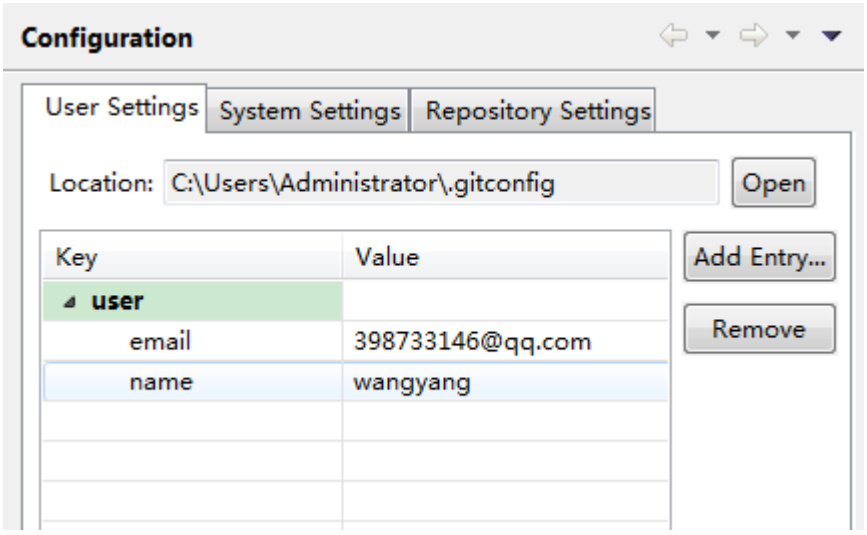
3.3、配置 git

1. 从 Eclipse 中搜索安装 egit 插件
2. 配置 egit，步骤：Eclipse 菜单栏->Window->Preferences 点击打开，搜索”git”(Team->Git->Congiguation->Add Entry..),然后分别建立一个 key 为 user.email 和 user.name 的变量， value 分别填上你自己申请的 github 的 email，name 可以随便填写（区分身份）

图一：



图二：

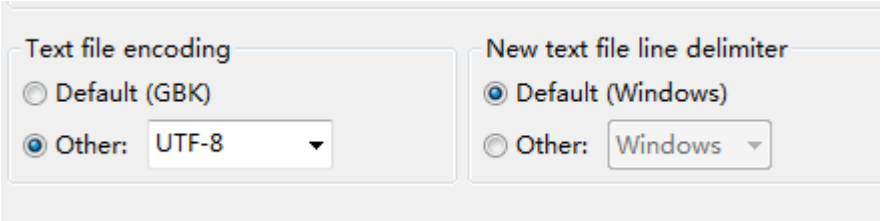


3.4、安装 testng

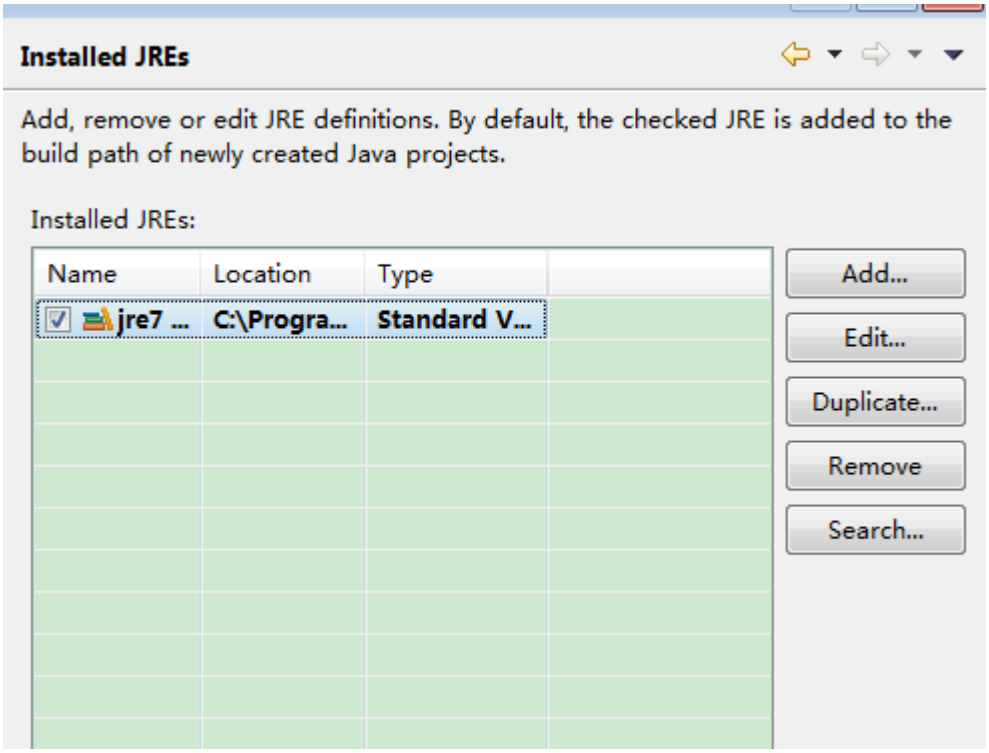
直接从 Eclipse 中搜索安装 testng 即可，不需要额外配置

3.5、eclispe 中的一些设置

1. 更改 Eclipse 编码为 UTF-8，操作如下：Eclipse 菜单栏->Window->Preferences 点击打开->General->Workspace->Text file encoding –>other:UTF-8,如图：

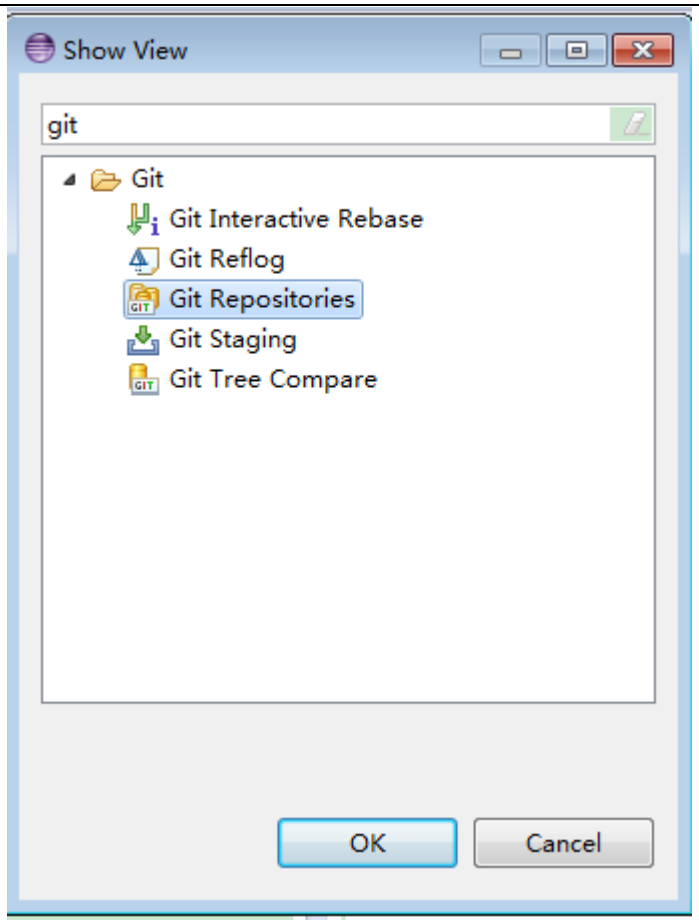


2. 更改 Installed JREs 的路径，操作如下：Eclipse 菜单栏->Window->Preferences 点击打开->Java->Installed JREs->Edit->然后选择前面步骤安装好的 java 的 jdk 目录，比如我的：C:\Program Files (x86)\Java\jdk1.7.0_65

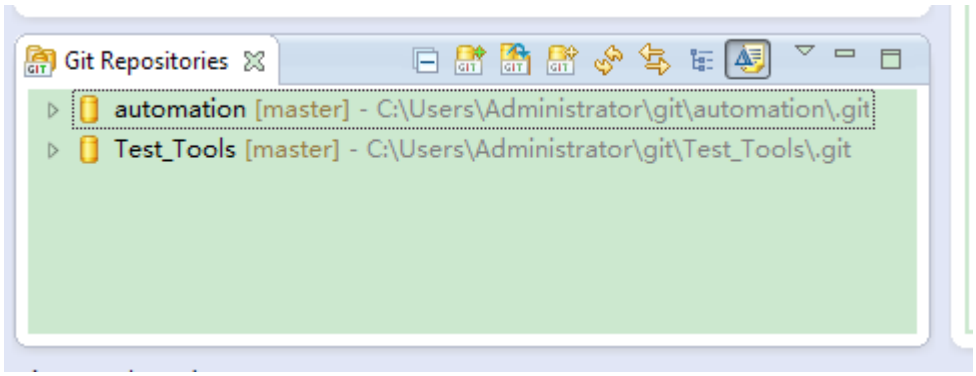


3.6、从 github 克隆项目

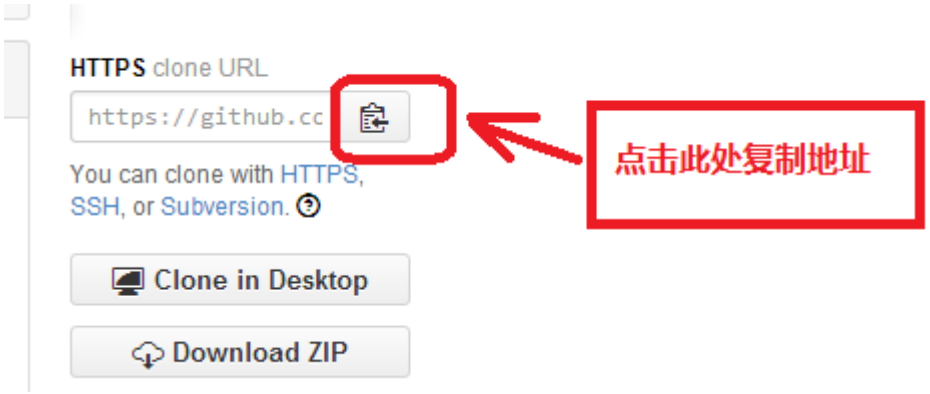
1. 进入 Eclipse 菜单栏->window->Show view->other->搜索 git->选择 Git Repositories，然后点击 OK



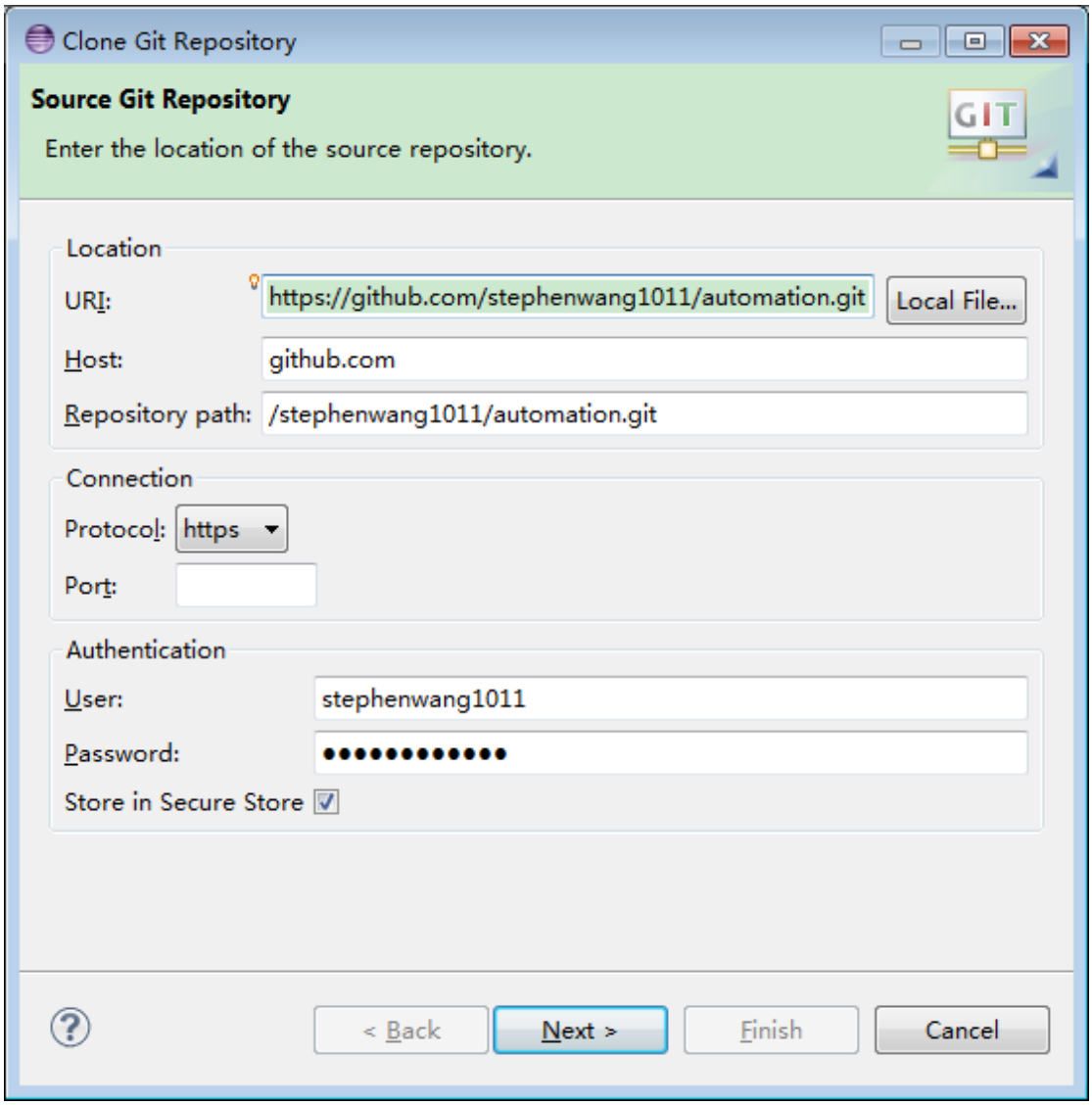
2. 之后 Eclipse 中会出现这样一个视图：



3. 打开此网址：https://github.com/stephenwang1011/automation，然后按照下图操作



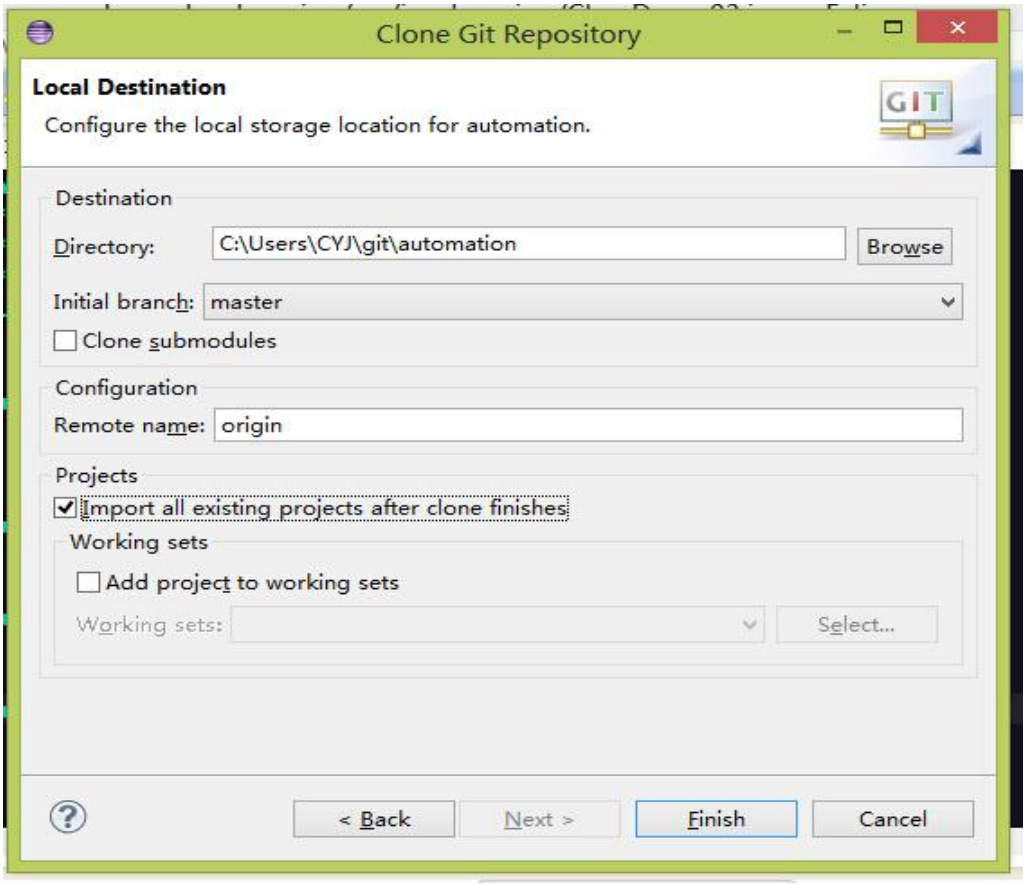
4. 复制完成以后点击第二步中的 clone 图标：  ，你刚才复制的地址会自动补全，如图：



其中的

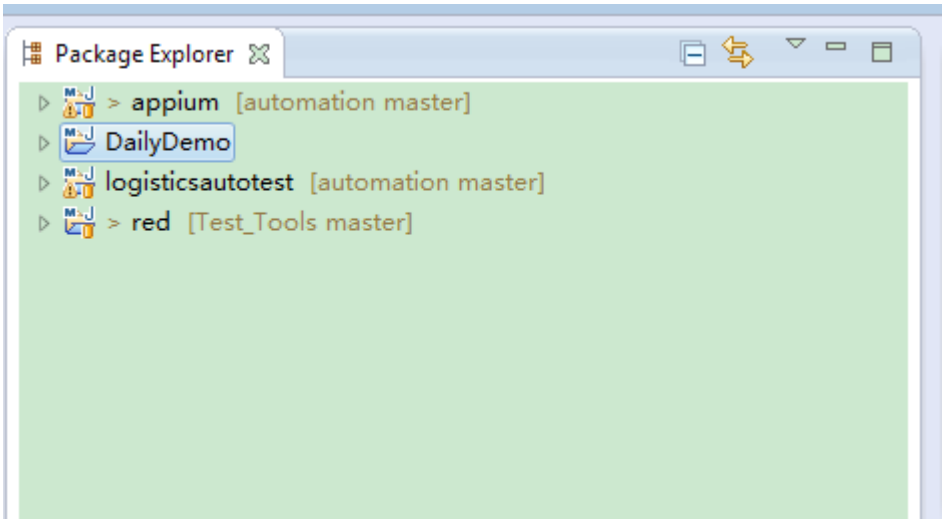
需要填写你的 github 用户名和密码，前提是你需要先注册一个。

5. 点击上图的“Next”直到下个页面，此时勾选下“Import all existing projects after clone finshes”

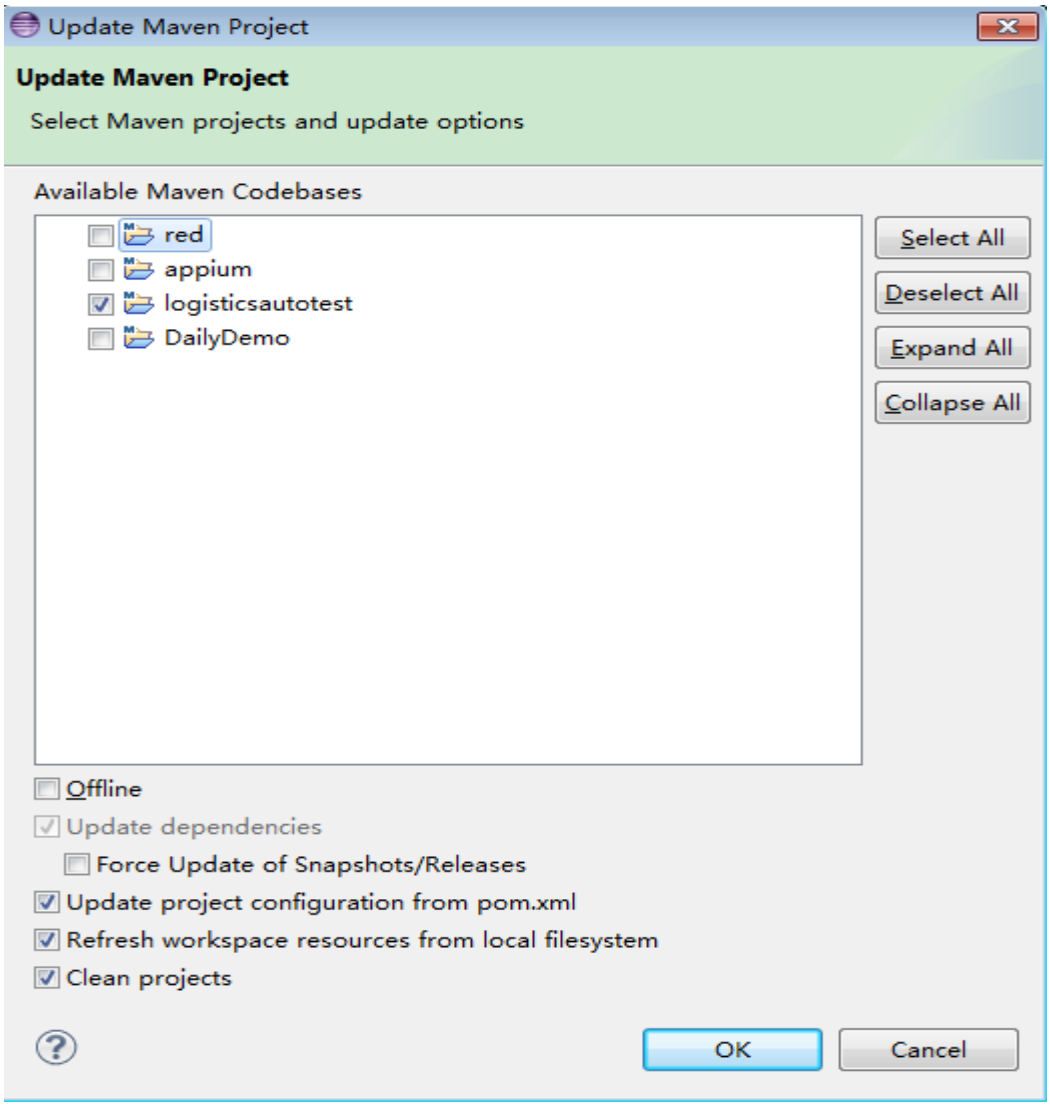


6. 点击“Finish”即可

7. 完成 clone 之后，会在 package explorer 中显示出 github 中的项目，如图

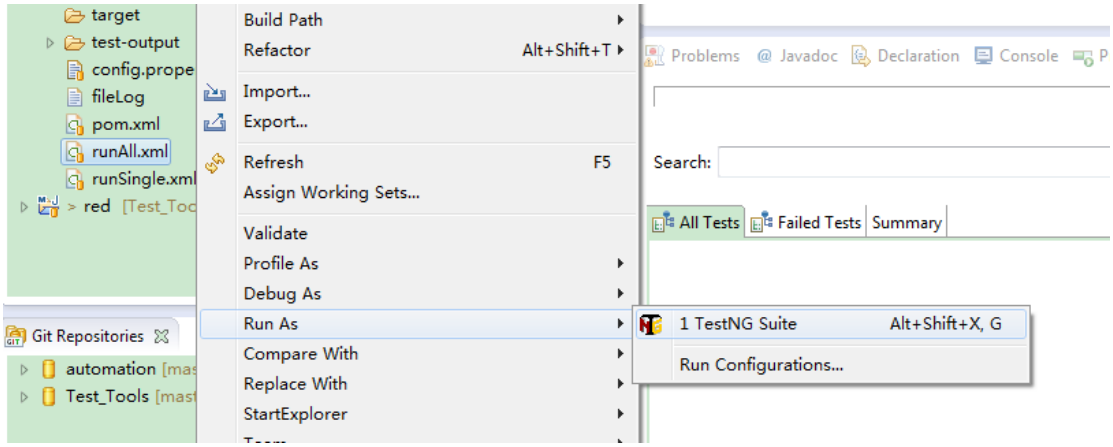


8. 其中的 logisticsautotest 项目就是物流货代 web 端的自动化测试，这个时候还需要借助 maven 构建下。首先选中这个项目，然后右键->maven->Update Project..->然后按照下图操作->点击 OK 即可。首次导入项目 maven 会下载项目依赖以及各种插件，耗时会长点，有可能会由于网络或者其他原因导致失败，多试几次就可以了



3.7、执行测试用例

1. 确保以上步骤全部准备完毕并且无误，鼠标选中 runAll.xml->Run as ->TestNG Suite



2. 此时会运行配置好的所有测试用例

3.8、测试配置更改

1. 如果说你想更改测试的浏览器或者说测试的平台，测试数据等等，可以在 runAll.xml 里面修改，如何修改，可以直接打开此文件，里面我做了详细的注释，哪个地方是做什么的看了就懂
2. 如何修改用例重跑次数？打开 config.properties，修改参数 retrycount=n n 是整数
3. 其他配置待补充…

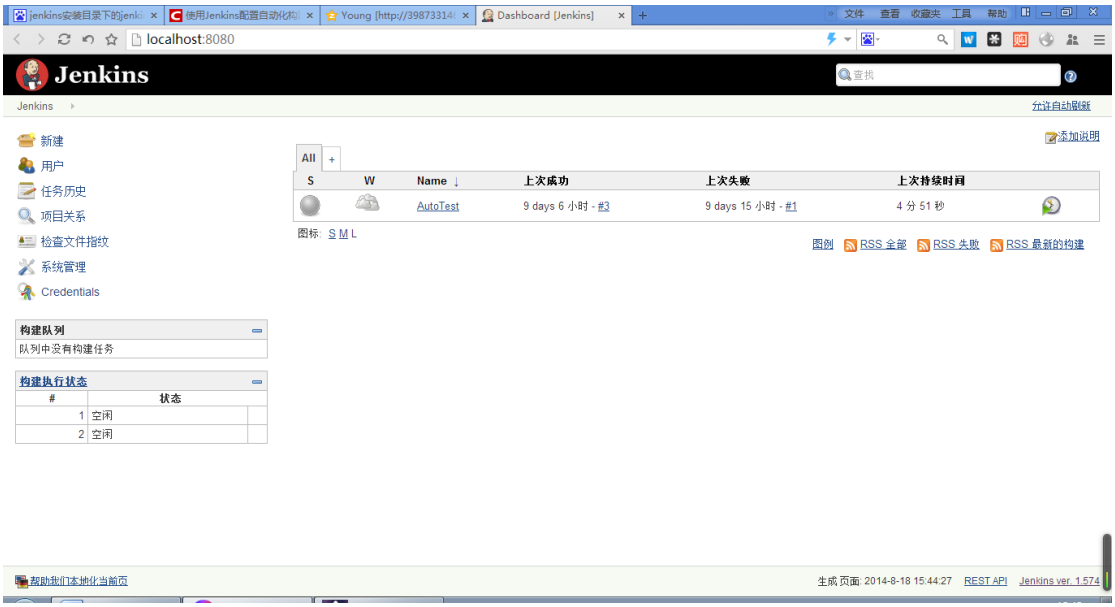
四、jenkins 持续集成测试篇

4.1、安装 jenkins

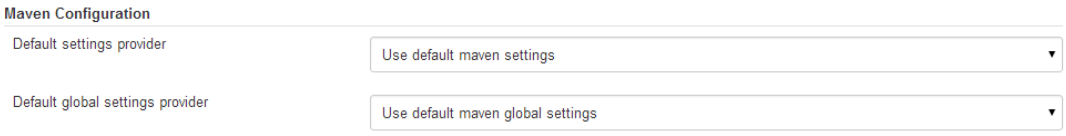
1. 官网下载 jenkins windows 安装版：点击下载
2. 点击 exe 文件执行安装

4.2、配置 jenkins（全局）

1. 在 jenkins 的安装修改 jenkins.xml 中的
   <arguments>-Xrs -Xmx256m -Dhudson.lifecycle=hudson.lifecycle.WindowsServiceLifecycle -jar "%BASE%\jenkins.war" --httpPort=8080</arguments>
   为
   <arguments>-Xrs -Xmx256m –Dfile=utf-8 -Dhudson.lifecycle=hudson.lifecycle.WindowsServiceLifecycle -jar "%BASE%\jenkins.war" --httpPort=8080</arguments>
   这一步主要是为了解决 jenkins 构建时中文出现乱码的情况
2. 浏览器中输入 http://localhost:8080 此时会打开 jenkins 主页，如图



3. 在主页点击“系统管理”->插件管理，在可选插件中搜索 git plugins 和 github plugins 两个插件进行安装，安装后重启 jenkins 生效
4. 再进入“系统管理”->“系统设置”进行更详细的配置
5. Maven 配置，如图



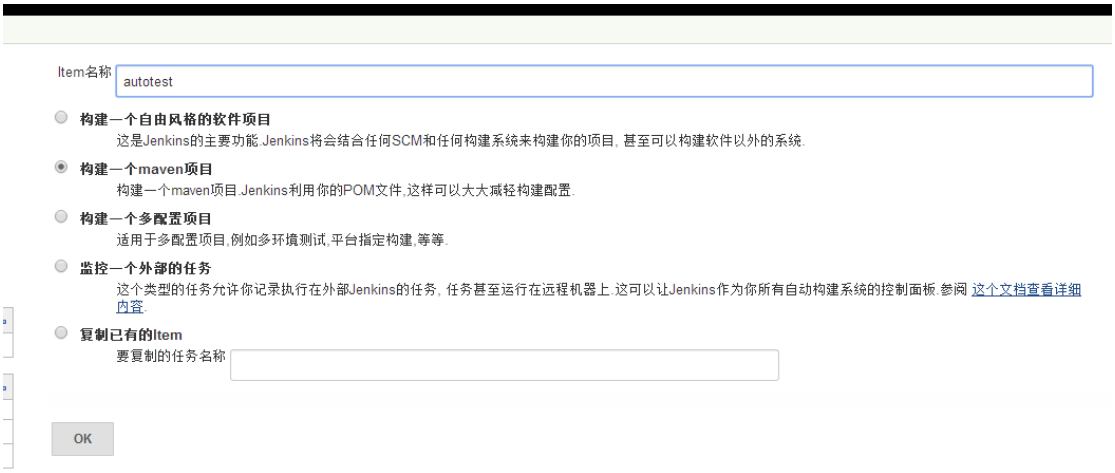
6. Git 配置：可以手动安装下载 git 之后制定 git.exe 路径，也可以直接在配置处勾选自动安装。比如我的是手动安装的 git,如图：

7. 配置 github 的用户名和邮箱

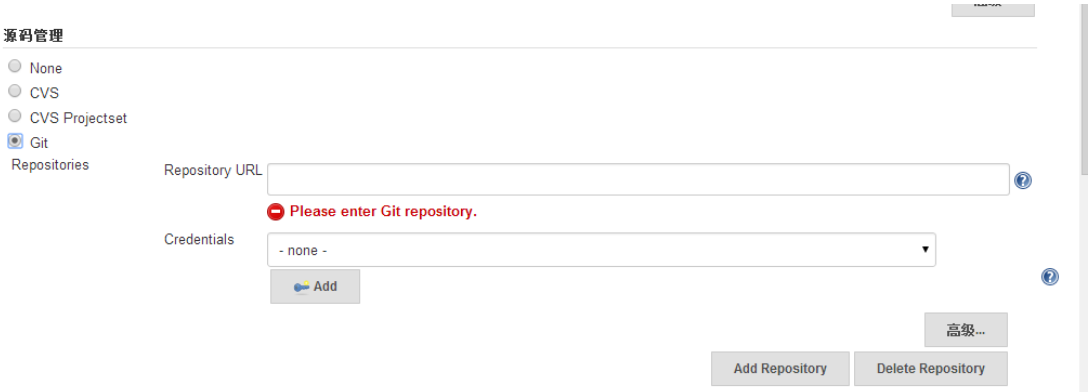
8. 点击应用按钮即可保存此页的设置

4.3、新建构建项目

1. 点击首页新建按钮
2. 构建一个 maven 项目，如图



3. 源码管理：选择 git，URL 中输入项目库地址



4. 构建定时触发器：勾选 Poll SCM，日程表中输入：H 8 * * *意思是：每天上午八点自动运行项目

5. 构建：指定好项目 pom.xml。本项目路径：logistics/pom.xml。Goals and options 输入 test 即可

6. 点击应用按钮本页设置保存生效

4.4、手动执行构建

1. 在 jenkins 首页会看到自己刚才新建的 maven 项目:test
2. 如图所示：进行手动构建

五、其他说明

5.1、页面定位工具

一般使用 firefox 中插件 firebug 和 firepath 可以精准定位页面元素位置，其他浏览器可以通过右键->”审查元素”查看页面元素

5.2、提供 selenium 入门文档

1. Selenium Testing Tools Cookbook 中文 （提取码：2073）
2. selenium webdriver 学习（提取码：df50）

2014 年 8 月