# 自动化测试框架的安装及使用

## Python + Nosetests + Webdriver + Eclipse

## 目录


自动化测试框架的安装及使用 1

Python + Nosetests + Webdriver + Eclipse 1

1 Python + Nosetests 安装及配置 2

1.1 Python 安装，以 Windows 为例 2

1.2 Python 配置 2

1.3 Python 插件安装 2

1.3.1 setuptools 安装 2

1.3.2 pip 安装（可选安装，如需在线安装 selenium，则需要安装） 3

1.3.3 Nosetests 安装 3

1.3.4 nose-testconfig 安装 3

1.3.5 Unittest 安装 3

1.3.6 requests 安装 4

1.3.7 nose-selenium-0.07 安装 4

1.3.8 其他插件安装 4

2 Webdriver 安装 5

2.1 Selenium Webdriver 安装 5

2.2 Chrome Webdriver 安装 5

2.3 IE Webdriver 安装 5

3 Eclipse 安装 5

4 框架使用说明 8

4.1 代码目录结构 8

4.2 使用说明 9

# 1 Python + Nosetests 安装及配置

下面所有安装均以离线安装为例（考虑公司网络问题），以下所有安装文件均可从“/hjqa/06 技术分享/03 Selenium/automation”中获取

## 1.1 Python 安装，以 Windows 为例

获取 python-2.7.6.msi 或者 python-2.7.5.msi （ /hjqa/06 技术分享 /03 Selenium/SeleniumInstall/Python_pack）双击 msi 文件安装即可

## 1.2 Python 配置

右单击我的电脑（或计算机）->属性->高级->环境变量，在系统变量中找到 PATH 变量并编辑，添加 Python 安装路径到 PATH 变量（如：C:\Python27；）
打开 cmd 窗口，运行 python，显示如下图：


## 1.3 Python 插件安装

### 1.3.1 setuptools 安装

目的：为安装其他插件做准备
获取 setuptools-3.3.zip， 拷贝 setuptools zip 包到 C 盘根目录解压，然后打开 cmd 窗口，切换路径至 setuptools-3.3，运行 python setup.py install， 如下图所示

```
Installed c:\python27\lib\site-packages\setuptools-3.3-py2.7.egg
Processing dependencies for setuptools==3.3
Finished processing dependencies for setuptools==3.3
```

安装完成后，可以在 C:\Python27\Lib\site-packages 下面找到 setuptools 文件夹及相

关目录， 和 C:\Python27 下面多了一个 Scripts 目录，并能在 C:\Python27\Scripts 下面看到 ez_install.exe

## 1.3.2 pip 安装（可选安装，如需在线安装 selenium，则需要安装）

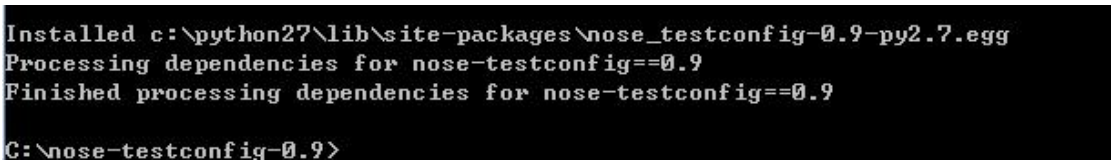
目的：可以使用该命令在线安装插件
获取 pip-1.0.2.zip，拷贝 pip-1.0.2 zip 包到 C 盘根目录解压，然后打开 cmd 窗口，切换路径至 pip-1.0.2，运行 python setup.py install
安装完成后，可以在 C:\Python27\Scripts 下面找到 pip.exe，和能在 C :\Python27\Lib\site-packages 下面找到 pip-1.0.2-py2.7.egg 文件夹
安装成功后，可以删除解压后的目录

## 1.3.3 Nosetests 安装

目的：该插件属于我们的框架，为运行用例做准备
获取 nosetests 文件，解压文件，将目录拷贝到 C:\ 下面，打开 cmd 窗口，切换目录到 C:\nose-1.3.1，运行 python setup.py install
安装完成后，可以在 C:\Python27\Scripts 下面看到 nosetests.exe 及相关文件
为了确保 nosetests 对其他目录也有效，参看 1.2 添加 C:\Python27\Scripts;到环境变量 PATH 里面。
打开 cmd 窗口，输入 nosetests，回车，没有错误显示表明安装成功

## 1.3.4 nose-testconfig 安装

目的：该插件用于框架里面的 config 文件导入，比如 from testconfig import config
获取 nose-testconfig-0.9.tar.gz 文件，解压文件，将目录拷贝到 C:\ 下面，打开 cmd 窗口，切换目录到 C:\ nose-testconfig-0.9，运行 python setup.py install 安装成功，如下图所示

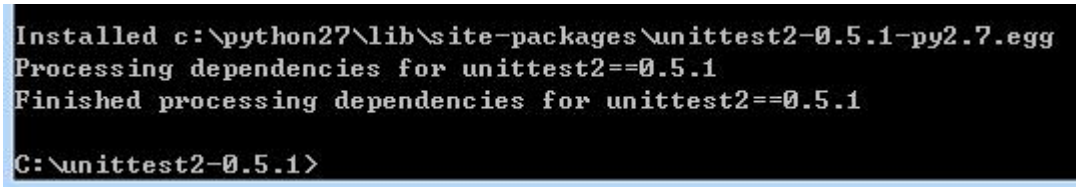



## 1.3.5 Unittest 安装

目的：该插件其实 Python 有自带，这里是安装最新版本，为安装 nose-selenium 插

件做准备

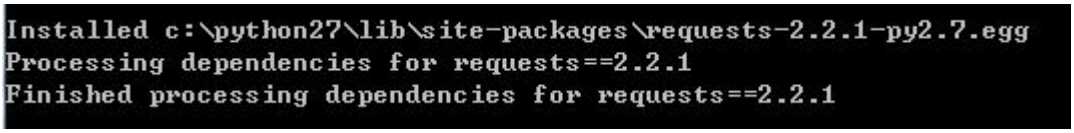
获取 unittest2-0.5.1.zip 文件，解压文件，将目录拷贝到 C:\ 下面，打开 cmd 窗口，切换目录到 C:\ unittest2-0.5.1，运行 python setup.py install 安装成功，如下图所示




## 1.3.6 requests 安装

目的：为安装 nose-selenium 插件做准备

获取 requests-2.2.1.tar.gz 文件，解压文件，将目录拷贝到 C:\ 下面，打开 cmd 窗口，切换目录到 C:\ requests-2.2.1，运行 python setup.py install 安装成功，如下图所示




## 1.3.7 nose-selenium-0.07 安装

目的：该插件用于 nose 框架和 selenium 结合使用

获取 nose-selenium-0.07.tar.gz 文件，解压文件，将目录拷贝到 C:\ 下面，打开 cmd 窗口，切换目录到 C:\ nose-selenium-0.07，运行 python setup.py install 安装

## 1.3.8 其他插件安装

如需安装其他插件，请按上述方法操作

Pydoc 插件：用于连接 SQL Server 数据库 ，获取 pydoc.zip 文件，直接解压后，运行 EXE 即可。

Lxml 插件（直接安装）：用于解析 XML 文件，获取 lxml-3.3.5.win32-py2.7.zip 文件，直接解压后，运行 EXE 即可。

Openpyxl 插件安装，获取 openpyxl-1.8.6.tar.gz 文件，解压文件，将目录拷贝到 C:\ 下面，打开 cmd 窗口，切换目录到 C:\ openpyxl-1.8.6，运行 python setup.py install 安装即可。

# 2 Webdriver 安装

## 2.1 Selenium Webdriver 安装

目的：WebDriver 用于启动浏览器，模拟用户在浏览器上做操作
获取 selenium-2.40.0.zip 文件，解压到 C:\，打开 cmd 窗口，切换目录到 C:\ selenium-2.40.0，运行 python setup.py install，安装成功后，输入 python 命令换行，进入 python 编辑模式，输入 from selenium import webdriver，不报错，表明已经安装成功

## 2.2 Chrome Webdriver 安装

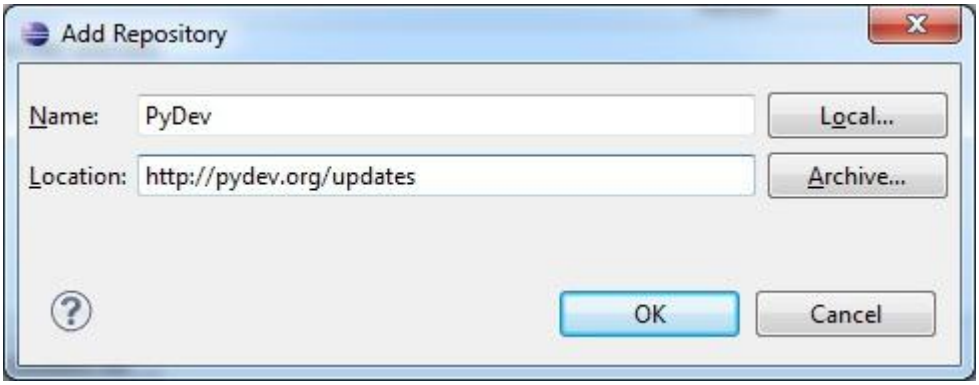
目的：用于 Webdriver 能够启动 Chrome 浏览器
获取 chromedriver.zip 文件，解压文件，将 chromedriver.exe 拷贝到 C:\Python27 目录下，并添加 C:\Users\Administrator\AppData\Local\Google\Chrome\Application\（chrome 安装路径，这里是 win7 下的安装路径）到环境变量 path

## 2.3 IE Webdriver 安装

目的：用于 Webdriver 能够启动 IE 浏览器
获取 IEDriverServer_x64_2.33.0.zip 文件，解压文件，将 IEDriverServer.exe 拷贝到 C:\Python27 目录下

# 3 Eclipse 安装

**前置条件：系统已经安装 JDK**
获取 eclipse.zip 文件，解压文件，比如：D:\eclipse，直接打开 eclipse.exe 即可运行
如果你的 eclipse 里面没有安装 pyDev，请按以下步骤安装：
1） 在 Help 菜单中，选择 Install New Software…
2） 选择 Add 按钮，Name：PyDev（这个随便起），Location：http://pydev.org/updates（PyDev 的更新地址），点击 OK




3） 选择 PyDev 下的 PyDev for Eclipse，别的都不要选，否则依赖检查那关过不去

4） 不要勾选“Contact all update sites during install to find required software”，点击 Next 安装即可

5） 重启 Eclipse，会看到 pyDev 插件，系统会自动为其配置，可以创建 PyDev project 表示已经安装成功

Eclipse - SVN 插件安装

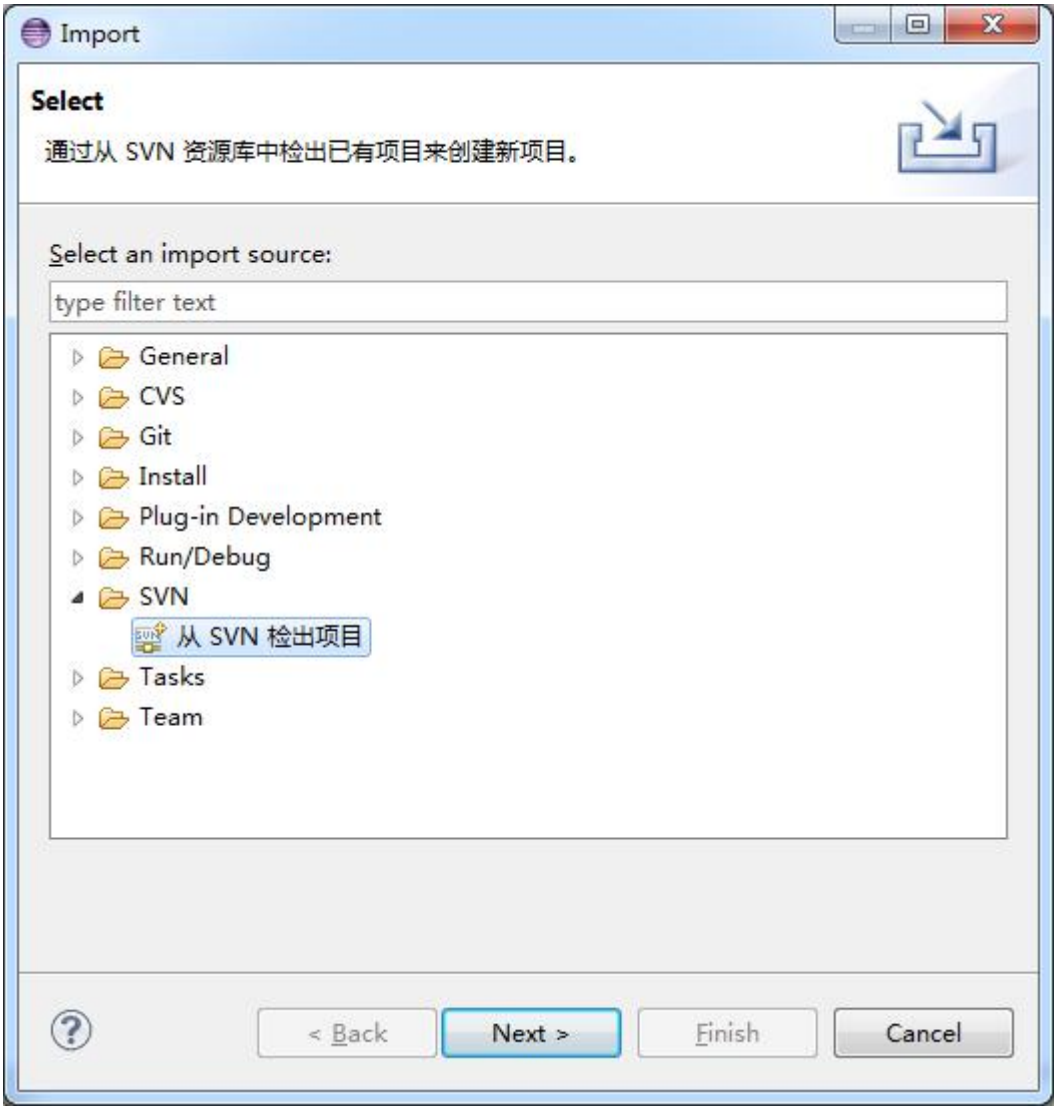
1） 在 Help 菜单中，选择 Install New Software・・・

2） 选择 Add 按钮，Name：SVN，Location：http://subclipse.tigris.org/update_1.8.x， 点击 OK

3） 勾选 Subclipse 和 SVNKit

4) **不要勾选**“Contact all update sites during install to find required software”，点击 Next 安装即可，重启 Eclipse 即可

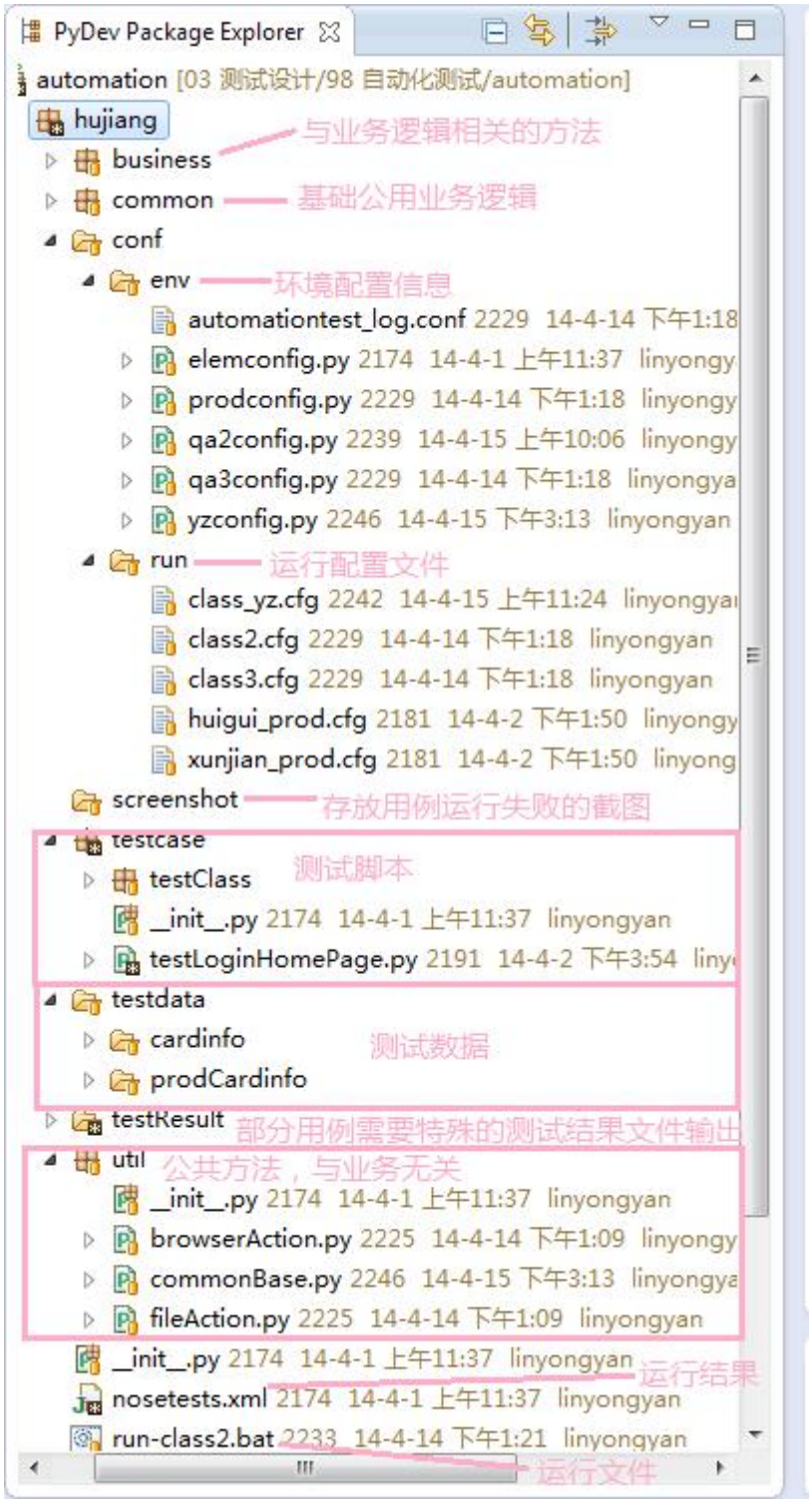
5） 安装完成后即可从 SVN 中直接导入自动化项目工程




6) 点击 Next，创建新的导入文件路径 http://192.168.25.69:8080/svn/hjqa/， 选择 /03 测试设计/98 自动化测试/automation，点击 Next，导入即可。
这样设置后可以方便更新上传自动化测试代码

# 4 框架使用说明

## 4.1 代码目录结构



## 4.2 使用说明

- conf -> env: 存放各个测试环境的信息，比如：
  - prodconfig.py：存放 prod 环境的用户登录信息，主站点以及各产品线站点的 URL，比如：config['class'] = 'http://class.hujiang.com/'
  - yzconfig.py: 存放验证环境的用户登录信息，主站点以及各产品线站点的 URL 比如：config['class'] = 'http://yz.class.hujiang.com/'
  - qa2config.py: 存放 qa2 环境的用户登录信息，主站点以及各产品线站点的 URL 比如：config['class'] = 'http://class2.hujiang.com/'
  - qa3config.py: 存放 qa3 环境的用户登录信息，主站点以及各产品线站点的 URL 比如：config['class'] = 'http://class3.hujiang.com/'

```
# coding = utf-8

global config
config = {}

# The location of automation log configuration file
config['testLogConf'] = 'conf/env/automationtest_log.conf'

# Normal user name and password
config['username'] = 'homeqa2014'
config['password'] = 'pa44wOrd'

#test class use username/pwd
config['classUsername'] = 'qingzi_014'
config['classPwd'] = 'hujiang'

# Main entry
# hujiang home page
config['hjhome'] = 'http://www.hujiang.com/'
# register/login entry
config['regurl'] = 'http://pass.hujiang.com/'
```

- conf -> run: 可以在 cfg 文件里面指定要运行的用例和运行环境的配置文件

```
[nosetests]
; Hard-code the nose-testconfig file format
tc-format=python
;with-selenium-driver = true
;webdriver = firefox
;webdriver = chrome
with-xunit=True
; This helps Nose traverse through our directory structure.
;include=Functional|nose
; The list of tests goes here
tests=testcase/testClass   指定运行的用例或者用例目录
;exclude= TC

; Testconfig - override on the nosetests command line for alternate environments
tc-file=conf/env/prodconfig.py   指定配置文件
```

如果需要指定运行某一个特定的用例，只需更新 tests=[用例相对路径]，比如：tests=testcase/testClass/testClassCommonStudyCard.py

如果需要在线上真实环境运行，设置 tc-file=conf/env/prodconfig.py，如果要在验证环境上运

行，设置为 tc-file=conf/env/yzconfig.py 即可

- util：存放与业务逻辑无关的公共方法，比如涉及文件读写操作，浏览器启动，判断一个元素、文件、目录是否存在

- common：存放与基础业务逻辑相关的方法或类，即涉及各个产品之间的业务逻辑公共方法，比如：登录信息录入，注册信息录入，提交订单，取消订单等

- business: 存放与各产品业务相关的逻辑方法，比如：网校课程购买，团购商品购买，听写酷听写等

- testcase：存放测试用例，文件命名，类名和方法名以 test 开头或结尾，比如：testClassCommonStudyCard.py，类名 testClassCommonStudyCard

```
import os
from testconfig import config
from hujiang.util import browserAction, fileAction, commonBase
from hujiang.common import login
from hujiang.business import classBuyFlow

class testClassCommonStudyCard:


    def setUp(self):
        '''
        Constructor
        '''
        self.classUrl = config['class']
        self.username = config['classUsername']
        self.passwd = config['classPwd']

        self.driver = browserAction.startBrowser("FF")
        browserAction.cleanCookie(self.driver)
        self.driver.get("http://www.hujiang.com")

    def tearDown(self):
        browserAction.cleanCookie(self.driver)
        browserAction.closeBrowser(self.driver)

    def testCommonStudyCardAddAndUse(self):
        print "start test common study card add and use..."
        resultList = []
        dataList = [('group ID','class ID','card ID','status','comment')]
        driver = self.driver

        driver.find_element_by_link_text(u"登录").click()
        login.loginPage(driver, self.username, self.passwd)

```

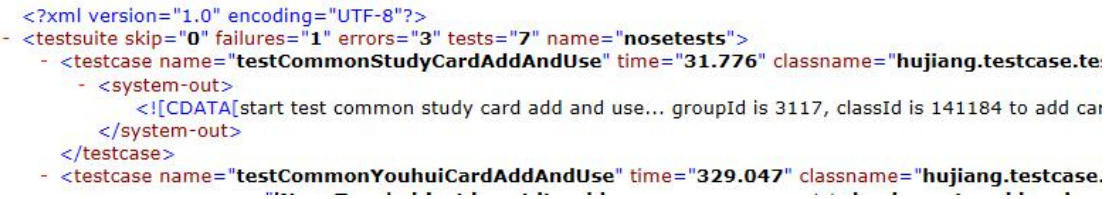
每个用例都至少包含一个 setUp() 和 teardown()。setUp()为准备测试数据，启动测试环境，如启动浏览器，确保运行环境干净等操作。teardown()清理现场，当测试运行结束，清除历史，恢复运行前的现场。

- testdata: 存放测试准备数据

- testresult：当特殊测试用例需要输出特别的测试报告， 保存记录

- bat 批处理文件：执行用例文件，该文件封装了用例执行命令，用户只需双击该文件即可运行相应的测试用例。如果用户不使用 bat 批处理文件，也可以直接打开 CMD 窗口，运行命令：nosetests -s -v -c [指定配置文件]， 如：nosetests -s -v -c conf\run\class2.cfg

- nosetest.xml：保存测试运行结果




Testsuite 结点属性解释：

- tests：运行的用例总量
- failures：运行失败的用例数量
- errors：运行用例出现 error，表明脚本需要更新，有 error 没有被抓住

testcase 结点中的 time：单个用例的运行时间