Django安装配置

1. Git版本控制系统

下载地址：<http://git-scm.com/>

1. 查看是否安装了pip：cmd中执行：**pip3**



1. 安装有python2和python3

在安装Python3（>=3.3）时，Python的安装包在系统中安装了一个启动器py.exe，默认放置在文件夹C:\Windows\下面。这个启动器允许我们指定使用Python2还是Python3来运行代码（当然前提是你已经成功安装了Python2和Python3）。

py.exe这个启动器允许你在代码中加入说明，表明这个文件应该是由python2解释运行，还是由python3解释运行。说明的方法是在代码文件的最开始加入一行

#! python2 或者 #! python3

分别表示该代码文件使用Python2或者Python3解释运行。这样，运行的时候你的命令就可以简化为py hello.py

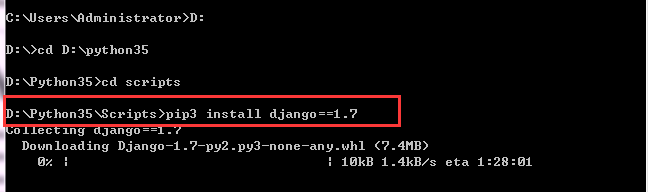
1. 安装有python2和python3，要明确pip版本

Py -2 –m pip install xxx

Py -3 –m pip install xxx

1. python3 安装Djando、selenium

进入D：\python35\scripts目录下，输入**pip3 install django==1.8**即可下载安装



Selenium：输入 **pip3 install –upgrade selenium** 即可下载安装



## Python3 与 Python2 的区别：

1. 去掉print语句，加入print()函数实现相同的功能

去掉exec语句，改为exec()函数

具体有哪些区别，详细见：

<http://www.cnblogs.com/codingmylife/archive/2010/06/06/1752807.html>

1. selenium用firefox浏览器来测试时(firefox47以上版本)，需安装第三方driver，即geckodriver，并将firefox目录添加到Path变量中

geckodriver下载url：<https://github.com/mozilla/geckodriver/releases>

否则，用Firefox()作为webdriver时，会报错：

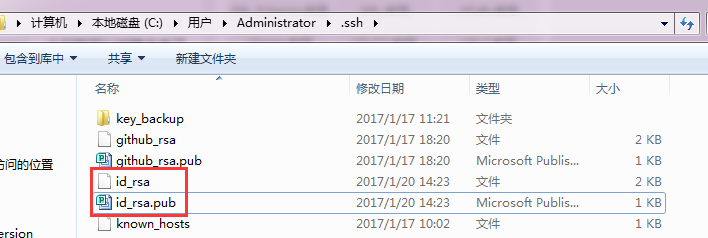
selenium.common.exceptions.WebDriverException: Message: 'geckodriver' executable needs to be in PATH.

## Github 配置：

1. 在本地创建ssh key，

进入本地github目录，执行命令：ssh-keygen –t rsa –C “**441650355@qq.com**” (“在github上注册的邮箱”)

生成key后，一般在C:\Users\Administrator目录有个“.ssh”文件夹，里面放有生成的key



1. 回到github上，进入Account Settings，左边选择SSH Keys，Add SSH Key，将在本地生成的key粘贴进去。

设置后，在git bash执行命令：ssh –T [git@github.com](mailto:git@github.com) ，会提示你已成功认证。

1. 在提交本地仓库到github上之前，需设置username和email，因为github每次commit都会记录这些信息

执行命令：

git config –global user.name “github注册时填写的username”

git config –global user.email “github注册时填写的email”

1. 进入要上传的仓库，右键git bash，添加远程地址：

git remote add origin [git@github.com:yourName/yourRepo.git](mailto:git@github.com:yourName/yourRepo.git)

## Github提交：

1. 将本地目录提交到github上：先进入该目录，执行以下命令：

1）将本地目录变成Git能管理的仓库： **git init**

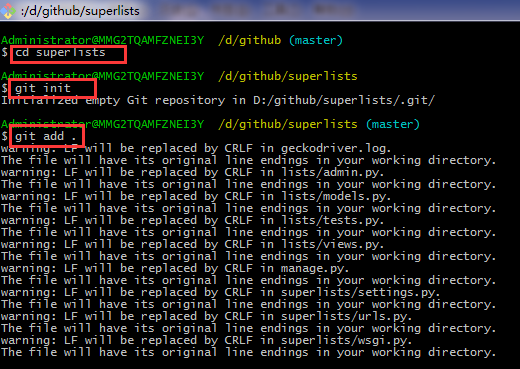
2）执行指令添加文件： **git add .**

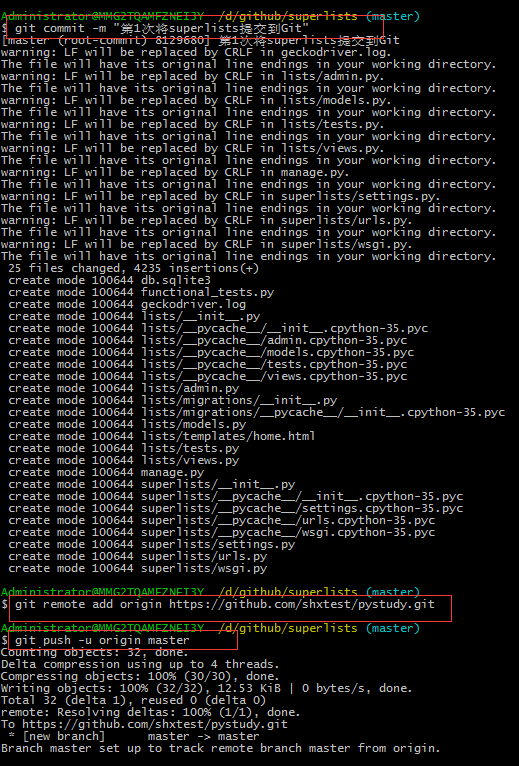
3）提交文件： **git commit –m “提交文件”**

4）关联github仓库： **git remote add origin <github的仓库路径>**

5）推送文件： **git push –u origin master[第一次推送]，后续推送最新修改，使用命令 git push origin master**

**以上步骤，首页提交时需执行git init，后续代码或文件有变动，只需执行2至5步。**





**提交时提示冲突，解决方法**：[用此方法解决前，要先把修改的文件备份下]

git fetch origin

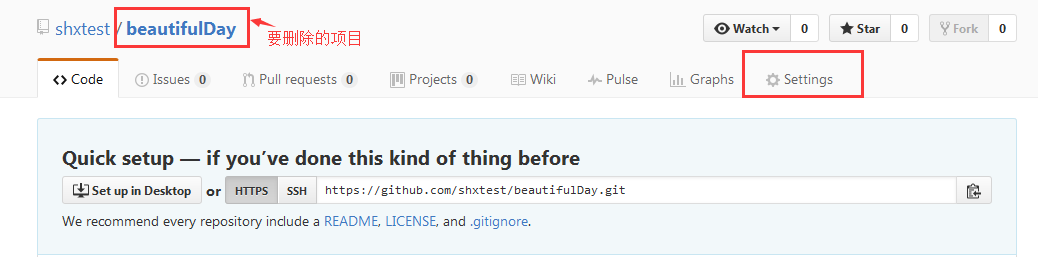
git reset –hard origin/master 或者回退到指定版本：git reset –hard 3628164[commit\_id]

git pull

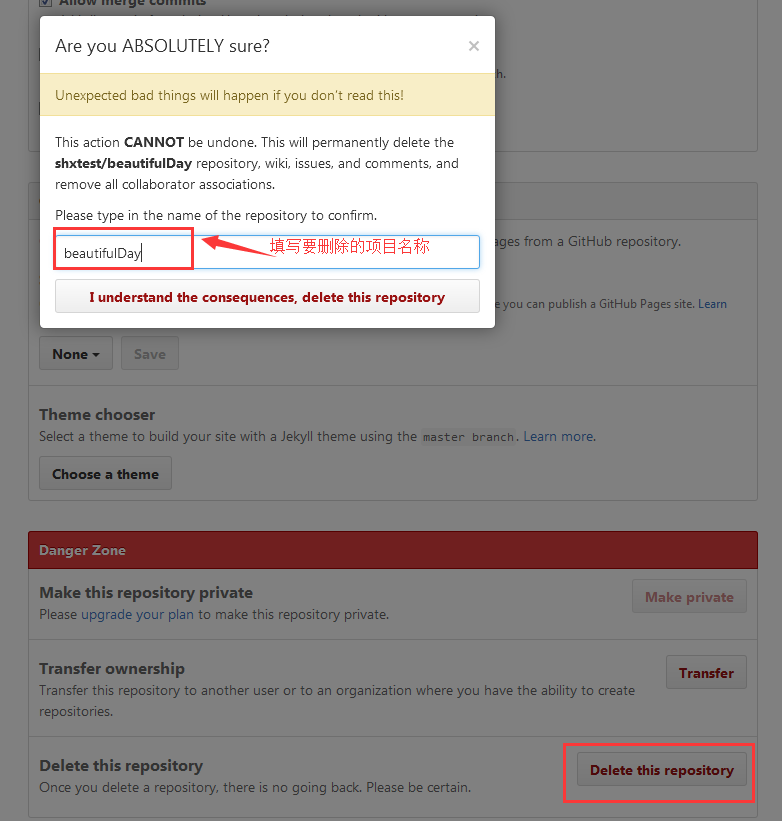
再重新提交

1. 删除github上的项目Repositories

第一步：进入项目Repositories，找到右边Settings项



第步：点击Settings，拖到最下边找到“Delete this repository”，填写要删除的项目名称即可删除该项目



1. **创建分支，执行以下命令：**

git checkout –b dev #创建dev分支

Switched to a new branch ‘dev’ #指向dev分支

git branch #查看当前分支

Git checkout master #切换回master分支

Switched to branch ‘master’

git branch –d dev #删除dev分支

Deleted branch dev

git branch #查看当前分支

**总结：**

**查看分支**：git branch

**创建分支**：git branch <name>

**切换分支**：git checkout <name>

**创建+切换分支**：git checkout -b <name>

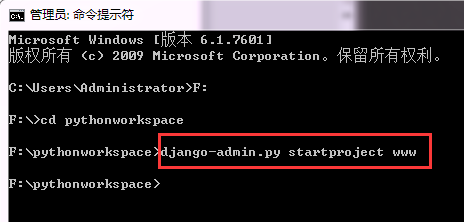
**合并某分支到当前分支**：git merge <name>

**删除分支**：git branch -d <name>

## Django框架：

1. 使用django创建项目，执行命令：

**Django-admin.py startproject www** # www是要创建的项目名称



1. 创建项目后，生成的文件目录如下：

|----www

|----manage.py

|----www

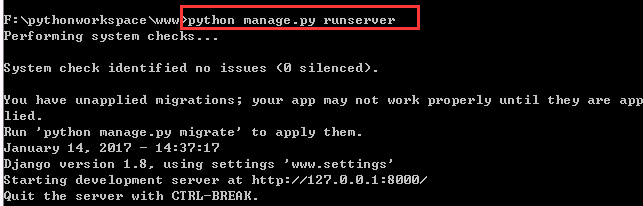
|----\_\_init\_\_.py

|---settings.py

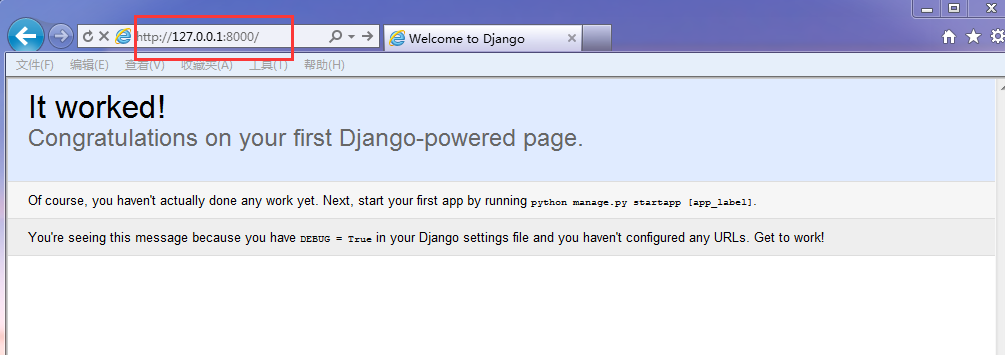
|----urls.py

|----wsgi.py

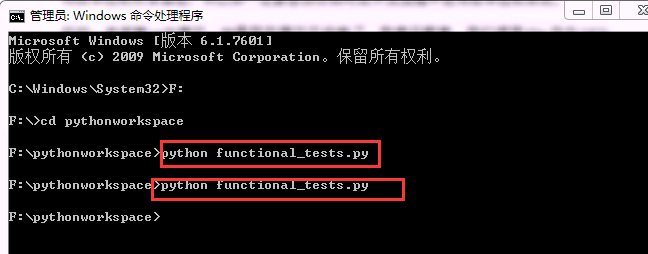
1. 运行开发服务器进行测试，执行命令：**python manage.py runserver**



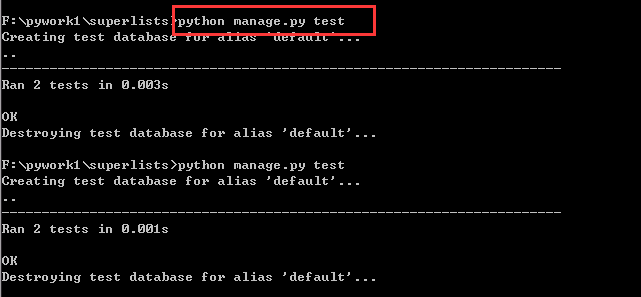
Django服务器运行起来后，可访问<http://127.0.0.1:8000> 打开页面测试是否启动成功



1. 运行功能测试，执行命令：**python test1.py**



1. 创建单元测试，执行命令：**python manage.py startapp lists**
2. 运行单元测试，执行命令：**python manage.py test**



Django 1.8 文档

**URL解析器异常**

url解析器异常，定义在django.core.urlresolvers中

### ViewDoesNotExist

\_\_exception\_\_ ViewDoesNotExist[source]

当所请求的视图不存在时，ViewDoesNotExist异常由django.core.urlresolvers产生

### Django 关于请求和响应