Contents

[Django项目创建流程-基于Linux环境 1](#_Toc24375713)

[1. 创建项目 1](#_Toc24375714)

[2. 创建应用 1](#_Toc24375715)

[3. 建立模板文件夹 1](#_Toc24375716)

[4. 配置模板文件 1](#_Toc24375717)

[5. 添加自己的应用服务 1](#_Toc24375718)

[6. Auth用户表拓展 2](#_Toc24375719)

[6.1. 扩展User表 2](#_Toc24375720)

[6.2. auth\_user表常用操作 2](#_Toc24375721)

[6.3. Auth组件常用功能 3](#_Toc24375722)

[7. 静态文件和媒体文件配置 3](#_Toc24375723)

[8. 项目目录下配置数据库 4](#_Toc24375724)

[9. 项目设置数据库 4](#_Toc24375725)

[10. 创建数据库所需要使用的表格 5](#_Toc24375726)

[11. 注册表格 5](#_Toc24375727)

[12. 数据库迁移 6](#_Toc24375728)

[12.1. 数据库准备迁移 python manage.py makemigrations 6](#_Toc24375729)

[12.2. 将准备迁移的数据同步到数据库: python manage.py migrate 6](#_Toc24375730)

[13. 非Django创建文件加载Django环境 6](#_Toc24375731)

[14. 数据库操作 6](#_Toc24375732)

[15.1. 表数据增,删,改 6](#_Toc24375733)

[15.2. 单表操作函数 7](#_Toc24375734)

[15.3. 模糊查询 7](#_Toc24375735)

[15.4. F查询 8](#_Toc24375736)

[15.5. Q查询 8](#_Toc24375737)

[15. 关于csrf\_token 9](#_Toc24375738)

[15.1. form表单中设置 9](#_Toc24375739)

[15.2. 视图函数views.py操作 9](#_Toc24375740)

[15.3. 网页中的内容展示 11](#_Toc24375741)

[16. 启动服务 12](#_Toc24375742)

[17. 关于应用的编程注意事项 12](#_Toc24375743)

[17.1. 应用app中的views写处理业务的逻辑 12](#_Toc24375744)

[17.2. urls中写路由urls.py 13](#_Toc24375745)

[17.3. 注意import进来所有的views函数 13](#_Toc24375746)

[17.4. setting文件中的debug 14](#_Toc24375747)

[17.5. cbv模式 14](#_Toc24375748)

[17.6. local变量的使用 14](#_Toc24375749)

[17.7. html中变量的引用 14](#_Toc24375750)

[18. 等上线后需要用wsgi.py来启动项目,就是项目里面的wsgi.py文件 14](#_Toc24375751)

# Django项目创建流程-基于Linux环境

## 创建项目

# 先前往目标路径, /usr/local/fea\_database

# 创建项目: *django-admin startproject proj\_name*

*django-admin startproject fea\_library*

## 创建应用

# 进入项目根目录

cd fea\*

# 创建应用:

*python3 manage.py startapp app\_name*

# 去项目的settings文件添加 应用名 到INSTALLED\_APPS

*‘应用名称.apps.AppConfig’,*

'app.apps.AppConfig',

*python3 manage.py startapp main*

## 建立模板文件夹

所有的html文件都放在这个文件夹下:

*mkdir templates*

## 配置模板文件

'DIRS': [os.path.join(BASE\_DIR, 'templates')],

## 添加自己的应用服务

INSTALLED\_APPS = [  
 'django.contrib.admin',  
 'django.contrib.auth',  
 'django.contrib.contenttypes',  
 'django.contrib.sessions',  
 'django.contrib.messages',  
 'django.contrib.staticfiles',  
 #增加自定义应用注册  
 'main',  
]

## Auth用户表拓展

根据需求在django基础上创建自己的用户表格,这个用户表是存储在settings.py中的database中的.



'''  
what：Django为开发者提供了一套可以完成用户注册、登录、登录校验、登录信息保存、密码的密文存储等一系列功能的模块，该模板默认关联着auth\_user表操作用户信息，django\_session表操作session信息。  
why：方便快捷的帮开发者完成登录相关的认证交互功能  
'''

### 扩展User表

需要在models.py文件中新建用户类,继承AbstractUser类.

# app/models.py  
# Create your models here.  
from django.contrib.auth.models import AbstractUser  
from django.db import models  
  
  
class UserModel(AbstractUser):  
 # Add my own defined items  
 # ['FEA Leader', 'FEA analyst', 'PE Leader', 'PE Member']  
 position=models.CharField(max\_length=20)  
 # ['CH', 'EU', 'NA']  
 location=models.CharField(max\_length=20)  
  
 class Meta:  
 db\_table='user\_table'  
  
 def \_\_str\_\_(self):  
 return self.first\_name+'.'+self.last\_name  
  
# 这句话注释掉了,因为只能创建一次超级用户,在第一次创建好以后就要注释掉了,这个是用来创建Django超级用户的.

UserModel.objects.create\_superuser(username='liw00073',password='cadfea73',email='wei.li@tenneco.com')

# settings.py配置  
# 配置AUTH表,可以放在任何一个地方  
AUTH\_USER\_MODEL = 'user.UserModel'

这个配置必须在数据库迁移前进行,也就是第一次进行操作.因为如果已经迁移过,那么就会有系统自带的auth\_user,而这个不是我们期待的表,我们希望建立具有auth\_user以及自定义字段的表.

自定义Django的显示字段,在user/admin.py中

from django.contrib import admin  
from .models import UserModel  
  
  
class UserModelAdmin(admin.ModelAdmin):  
 empty\_value\_display = '-empty-'  
 list\_display = ['username', 'first\_name', 'last\_name', 'email', 'is\_superuser', 'position', 'location']  
 list\_display\_links = ['username', ]  
  
  
# Register your models here.  
admin.site.register(UserModel, UserModelAdmin)

### auth\_user表常用操作

from app.models import MyUser  
# 1、创建超级用户,可以登录admin后台,管理整个项目  
MyUser.objects.create\_superuser(username='root', password='root', email='root@root.com')

# 2、创建普通用户  
MyUser.objects.create\_user(username='Owen', password='123')  
​  
# 3、获取第一个用户  
user = MyUser.objects.first()  
​  
# 4、修改密码  
user.set\_password('000')  
user.save()  
​  
# 5、校验密码  
res = user.check\_password('000')

### Auth组件常用功能

# 1、校验用户账号及密码，校验成功返回user对象  
from django.contrib.auth import authenticate  
user = authenticate(username=usr, password=pwd)  
​  
# 2、注册用户到request对象中，注册成功可以request.user访问当前登录用户（会形成session记录）  
from django.contrib.auth import login  
login(request, user)  # 注册authenticate成功(当前登录)的用户  
​  
# 3、注销当前注册的user (用户注销)  
from django.contrib.auth import logout  
logout(request)  
​  
# 4、校验用户登录状态  
# 视图函数中使用  
if request.user.is\_authenticated(): pass  
# 模板语言中使用  
{% if request.user.is\_authenticated %}  
{% else %}  
{% endif %}  
​  
# 5、校验登录状态的装饰器  
from django.contrib.auth.decorators import login\_required  
@login\_required(login\_url='/user\_login/')  
def user\_home(request):  
   return render(request, 'user.html', locals())

## 静态文件和媒体文件配置

STATIC\_URL = '/static/'  
STATICFILES\_DIRS = [  
 os.path.join(BASE\_DIR, 'static'),  
]  
  
MEDIA\_URL = '/media/'  
MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')

# 还需在urls.py中做路由配置

#urls.py

# serve就是可以渲染媒体文件的视图函数

from django.views.static import serve

#导入的是django框架的settings,

from django.conf import settings

#这个路由放在最后,因为他是可以匹配多个结果,为了防止覆盖其他的路由,所以..采用正则的有名分组

url(r'^media/(?P<path>.\*)', serve, {'document\_root': settings.MEDIA\_ROOT}),

因为CSS文件不需要进入数据库,而media中间的一些文件是需要放在数据库中的,即使是路径,图片,音频之类的文件需要解码编码,所以需要导入serve函数

## 项目目录下配置数据库

# 项目目录下**\_\_init\_\_.py**

# 该文件是启动项目最新执行的文件，在该文件中将Django原操作数据库的mysqldb模块切换成pymysql

*import pymysql*

*pymysql.install\_as\_MySQLdb()*

## 项目设置数据库

**# settings.py**

*DATABASES = {*

*'default': {*

*'ENGINE': 'django.db.backends.mysql',*

*'NAME': '数据库名',*

*'HOST': '127.0.0.1',*

*'PORT': 3306,*

*'USER': 'root',*

*'PASSWORD': 'root'*

*}*

*}*

## 创建数据库所需要使用的表格

下面的操作是在数据库中新建一个表

与数据库的表所对应的类称之为model类,这个类的定义放在models.py中.

class User(models.Model):  
 id = models.AutoField(primary\_key=True)  
 login=models.CharField(max\_length=16, default='')  
 password = models.CharField(max\_length=225, null=False)  
 name = models.CharField(max\_length=225, null=False)  
 active = models.IntegerField()  
 position=models.CharField(max\_length=225, null=False)  
 location=models.CharField(max\_length=2, null=False)  
 email=models.EmailField(max\_length=225)  
 date\_format = models.DateField()  
 #设置User的表名  
 class Meta:  
 db\_table='user'

verbose\_name='书'#如果变成中文界面,会显示为中文

verbose\_name\_plural= verbose\_name #意思是如果是复数,也用和单数一样的名字

def \_\_str\_\_(self):

return self.name#这句话相当于在表中可以显示出名字

## 注册表格

为了能够在Django的admin界面中操作所有的表格,必须要注册.

在应用app下的admin.py中进行注册.

# /app/admin.py  
​  
from Django.contrib import admin

from app import models

admin.site.register(models.User)

admin.site.register(models.Book)

…

## 数据库迁移

### 数据库准备迁移 python manage.py makemigrations

### 将准备迁移的数据同步到数据库: python manage.py migrate

## 非Django创建文件加载Django环境

对于不是通过Django创建的文件,如果需要使用数据库,使用ORM关系映射,那么必须配置Django环境, 所以需要加载django配置

# 在项目根目录下新建一个测试脚本文件，eg：t\_dg.py  
​  
# 导入环境  
import os, django  
# 加载django配置  
os.environ.setdefault("DJANGO\_SETTINGS\_MODULE", "dg\_proj.settings")  
# 启动django  
django.setup()

## 数据库操作

### 表数据增,删,改

# 现在需要导入模型类

from app.models import User

#增

user=User(name=’zero’, age=8, birthday=2010.1.11’)

#需要保存到硬盘

user.save()

#改

user.age=10

user.save()

#删除

user.delete()

### 单表操作函数

'''  
1. all()：查询所有结果list，支持正向索引取值[i]，支持索引切片[m:n],类型为queryset,其实就是一个列表,但是索引不要有负值,也就是user\_query=User.objects.all()[1],1可以,-1不可以.  
2. filter(\*\*kwargs)：查询满足条件的所有结果list  用的最多,filter(name=’aa’)  
3. get(\*\*kwargs)：查询满足条件的唯一对象obj，查询结果不为obj则抛出异常    
4. exclude(\*\*kwargs)：查询满足条件的对立面的所有结果list  
5. order\_by(\*field)：查询按照指定字段进行排序后的所有结果list，'tag\_name'代表正序，'-tag\_name'代表降序  
6. reverse()：反转排序查询的所有结果list  
7. count()：统计返回查询结果list的长度  
8. first()：返回查询结果中的第一个对象obj  
9. last()：返回查询结果中的最后一个对象obj  
10. exists()：判断查询结果是否存在，值为布尔类型  
11. values(\*field)：按照指定字段(们)进行查询，返回存放包含字段(们)的字典的列表list  
12. values\_list(\*field)：同values类似，返回存放数据的元组的列表list  
13. distinct()：从查询结果中剔除重复字段(一般和values结合使用)  
'''

### 模糊查询

'''  
# 整型相关  
age\_\_exact=8 # 确切匹配8  
age\_\_in=[8, 10] # 8或10  
age\_\_gt=8 # 大于8  
age\_\_gte=8 # 大于等于8  
age\_\_lt=8 # 小于8  
age\_\_lte=8 # 小于等于8  
age\_\_range=[8, 10] # 8到10之间  
age\_\_isnull=0|1 # 0：is not null | 1：is null  
​  
# 字符串相关  
name\_\_startswith # 后方模糊匹配  
name\_\_endswith # 前方模糊匹配  
name\_\_contains # 前后方均模糊匹配  
name\_\_regex # 正则匹配  
name\_\_istartswith # 不区分大小写后方模糊匹配(i开头就是不区分大小写)  
​  
# 时间相关  
birthday\_\_year=2008 # 时间年份模糊匹配  
'''

### F查询

'''  
# 通过字段名获取可以直接做运算的查询结果  
from django.db.models import F  
F('tag\_name')  
​  
# 案例一：将id为1的结果年龄增加1  
User.objects.filter(id=1).update(age=F('age')+1)  
# 案例二：查询id是年龄1/4的结果  
user = User.objects.filter(id=F('age')/4)  
'''

### Q查询

'''  
# 完成逻辑运算方式的查询  
from django.db.models import Q  
# 与  
User.objects.filter(Q(id=1) & Q(age=10))  
# 或  
User.objects.filter(Q(id=1) | Q(id=2))  
# 非  
User.objects.filter(~Q(id=1))  
'''

## 关于csrf\_token

### form表单中设置

在中间件中不要禁用csrf组件,那么如果在登陆页面中没有拿到csrf\_token,就会报错.所以需要在登陆页面中拿到这个token.即在login.html中,在form表单后加上:

<form action="" method="post">  
  {% csrf\_token %}  
</form>

### 视图函数views.py操作

*from django.contrib.auth import authenticate, login, logout*

*# 登录*

*class LoginView(View):*

*def get(self, request, \*args, \*\*kwargs):*

*return render(request, 'login.html')*

*def post(self, request, \*args, \*\*kwargs):*

*username = request.POST.get('username')*

*password = request.POST.get('password')*

*user = authenticate(username=username, password=password)*

*if user: # 登录校验通过，代表登录成功*

*# 将登录信息告诉前台，并跳转到图书页面*

*login(request, user)*

*return redirect('/books/')*

*else: # 登录失败重新渲染登录页面*

*return render(request, 'login.html')*

*# 注销*

*class LogoutView(View):*

*def get(self, request, \*args, \*\*kwargs):*

*# 注销，会清除前台的sessionid*

*logout(request)*

*return redirect('/login/')*

*# 注册*

*class RegisterView(View):*

*def get(self, request, \*args, \*\*kwargs):*

*return render(request, 'register.html', {'title': '注册'})*

*def post(self, request, \*args, \*\*kwargs):*

*username = request.POST.get('username')*

*password = request.POST.get('password')*

*mobile = request.POST.get('mobile')*

*re\_password = request.POST.get('re\_password')*

*if not username or not password or not mobile or password != re\_password:*

*return render(request, 'register.html', {'title': '信息有误，重新注册'})*

*try:*

*models.MyUser.objects.create\_user(username=username, password=password, mobile=mobile)*

*return redirect('/login/')*

*except:*

*return render(request, 'register.html', {'title': '注册失败，重新注册'})*

*from app import models*

*class BooksView(View):*

*def get(self, request, \*args, \*\*kwargs):*

*# 没有登录，不能访问，重定向到登录页面*

*if not request.user.is\_authenticated():*

*return redirect('/login/')*

*book\_list = models.Book.objects.all()*

*return render(request, 'books.html', {'book\_list': book\_list})*

### 网页中的内容展示

**books.html:**

*<body>*

*<h1>图书们</h1>*

*<a href="/logout/">注销</a>*

*<hr>*

*<table border="1">*

*<tr>*

*<th>书名</th>*

*<th>价格</th>*

*<th>出版社</th>*

*<th>作者们</th>*

*</tr>*

*{% for book in book\_list %}*

*<tr>*

*<td>{{ book.name }}</td>*

*<td>{{ book.price }}</td>*

*<td>{{ book.publish.name }}</td>*

*<td>*

*{% for author in book.authors.all %}*

*{{ author.name }}*

*{% endfor %}*

*</td>*

*</tr>*

*{% endfor %}*

*</table>*

*</body>*

## 启动服务

# 在项目根目录下:

*python3 manage.py runserver 0.0.0.0:8000 &*

## 关于应用的编程注意事项

### 应用app中的views写处理业务的逻辑

from django.shortcuts import render, HttpResponse  
​  
def url\_1(request):  
   return HttpResponse('url\_1 %s ok' % request.method)  
​  
# def url\_2(request, num, string):  
#     return HttpResponse('url\_2 %s ok:（%s）（%s）' % (request.method, num, string))  
​  
def url\_2(request, \*args):  
   return HttpResponse('url\_2 %s ok:（%s）（%s）' % (request.method, args[0], args[1]))  
​  
# def url\_3(request, pk, mark):  
#     return HttpResponse('url\_3 %s ok:（%s）（%s）' % (request.method, pk, mark))  
​  
def url\_3(request, \*\*kwargs):  
   return HttpResponse('url\_3 %s ok:（%s）（%s）' % (request.method, kwargs['pk'], kwargs['mark']))  
​  
​  
def url\_4(request, pk, name='url4'):  
   return HttpResponse('url\_4 %s ok:（%s）（%s）'% (request.method, pk, name))  
​  
# 总结：可以匹配任意 正则格式的url路径 的视图函数  
def url(request, \*args, \*\*kwargs):  
   pass

### urls中写路由urls.py

# 1）一个死路径对应一个视图函数，视图函数只有一个参数request  
url(r'^url1/$', url\_views.url\_1),  
​  
# 2）无名分组匹配url路径的数据,应用场景,可以根据用户提供的ID,访问的目标编号来定义网页渲染  
url(r'^url2/(\d+)/(.+)/$', url\_views.url\_2),  
​  
# 3）有名分组匹配url路径的数据  
url(r'^url3/(?P<pk>\d+)/(?P<mark>.+)/$', url\_views.url\_3),  
# 注：有名无名分组不能组合使用  
​  
# 4）路由的默认参数(了解)  
# url\_4第三个参数name有默认值，可以不用传参，采用默认值  
# 如果想传参，用url 默认参数 方式传参  
url(r'^url4/(?P<pk>\d+)/$', url\_views.url\_4, {'name': 'URL4'}),  
​  
# 总结：  
# url(r'url正则', 视图函数, 默认参数),  
# url正则：内部可以包含无名分组，有名分组  
# 视图函数：实现体第一个参数一定是request对象，  
#       \*args来接受所有无名分组参数，  
#       \*\*kwargs用来接收所有有名分组参数 与 默认参数  
# 默认参数：额外给视图函数传递的参数  
#       有名无名分组都是前台url剥离出的参数，而默认参数都是后台自己的数据，一般来源于配置文件  
​

### 注意import进来所有的views函数

* 比如*from home import views as home\_views*
* #注意,^代表以home开头,$代表以其前面的/结束,所以也就是地址必须为home/才可以.这个是满足正则匹配语法的.路由的检测从上到下
* url(r’^home/$’, home\_views.home\_page),
* 通过POST方法,然后用get方法来获取字典里的关键词

name = request.POST.get('username')

* 可以看下request里面的很多参数,比如GET, POST, META的超大字典,客户端的IP端口信息.比如request.META.get(‘QUERY\_STRING’), request.META.get(‘HTTP\_COOKIE)

### setting文件中的debug

debug在开发阶段为true, allowed\_hosts为空,在上线后,debug设置为FALSE, allow\_hosts设置为允许访问服务器的白名单host,不要写端口.

### cbv模式

如果采用cbv模式,需要在view模块中导入

*from django.views import View*

### local变量的使用

*render(request, ‘show.html’,context=locals())*

### html中变量的引用

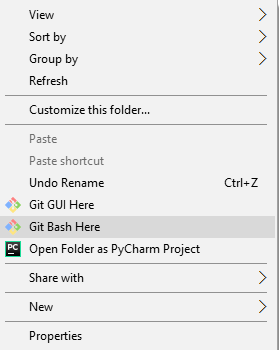
* # 在页面中，{{ key变量 }}访问基础数据，
* # 列表用 {{ key变量.索引 }}
* # 字典用 {{ key变量.key }}
* # 函数用 {{ 函数名 }}

## 等上线后需要用wsgi.py来启动项目,就是项目里面的wsgi.py文件

# GitHub存储文件步骤说明

## 1. 下载与安装

地址: [https://git-scm.com/download/win, 选择64](https://git-scm.com/download/win,%20选择64)位下载,安装的时候,什么都不管,一直NEXT就好.如果安装成功,点击鼠标右键,会出现如图所示:



## 2. 在github官网注册账户.

[我注册的账户关联邮箱为wei\_dora\_hennie@aliyun.com](mailto:我注册的账户关联邮箱为wei_dora_hennie@aliyun.com). 用户名为liwblue1981.

## 3. 操作本地文件夹

在需要备份的文件夹内点击鼠标右键, Git Bash Here

3.1 git init 用于初始化本地需要备份的文件夹

3.2 git status 用于查看当前管理状态,红色为发生变更的文件,绿色 为放置在缓存中的文件,白色为已经处理备份成功的 文件

3.3 git add . 后面有个.,表示将所有文件存入缓存进行备份

3.4 git commit -m ‘注释信息’ 将所有缓存中的文件进行备份到仓库,注释信息越详 细越好

## 4. 初次提交需要设置邮箱和用户名

根据提示信息设置即可

git config --global user.name 输入用户名 # 设置用户

git config --global user.email 输入邮箱名 # 设置邮箱

git config --global --unset user.name # 删除用户

git config --global --unset user.email # 删除邮箱

## 5. 查看记录

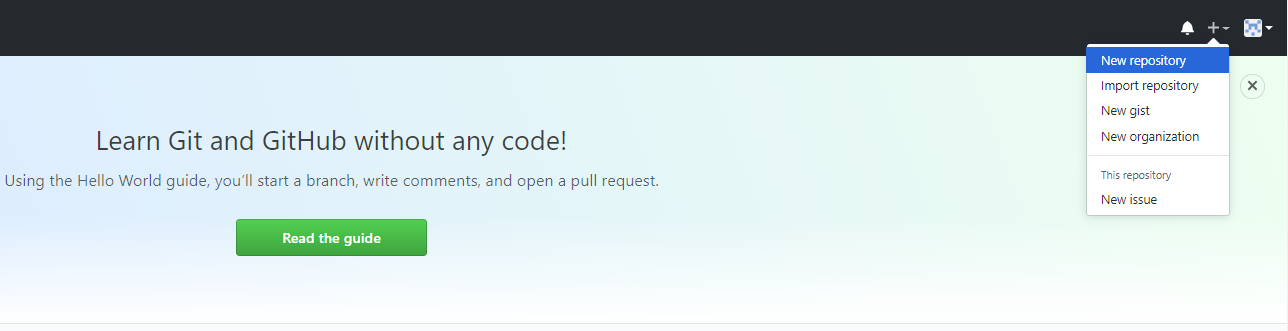
git log # 查看提交记录

git reset --hard '序列号' # 回退到之前版本状态

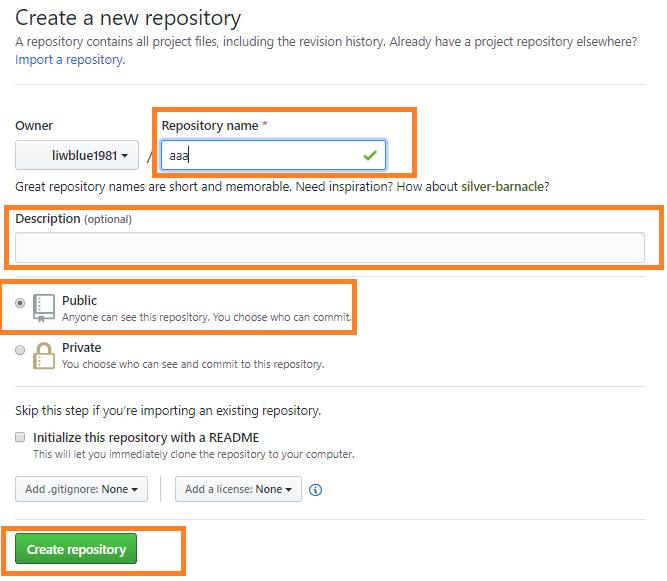
git reflog # 查看所有版本(包括回退之前的新版本)

## 6. 远程仓库,用于远程保存代码

### 6.1 创建新仓库

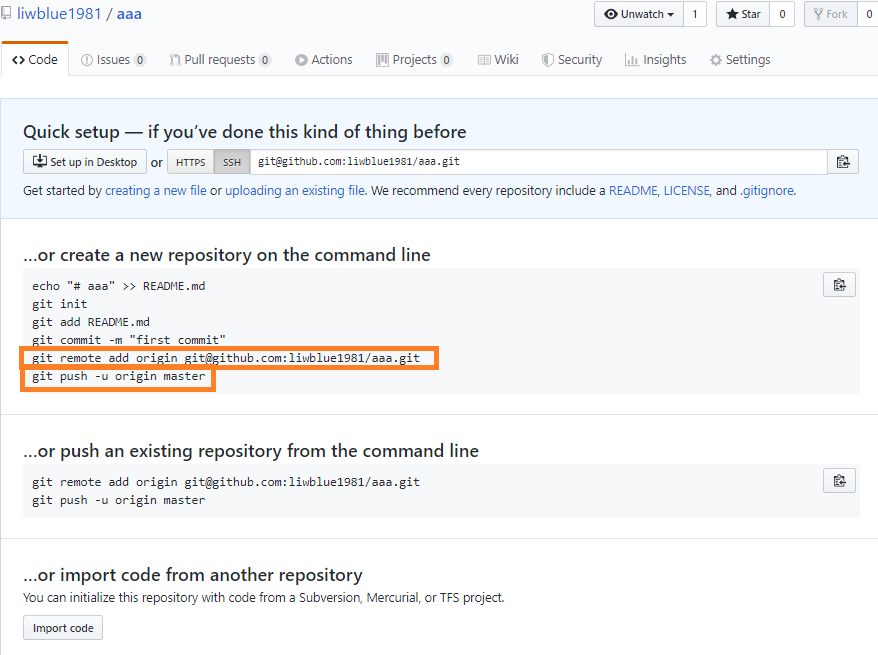


输入仓库名称,填写描述,选择public,然后直接点创建,不需要Initialize this repository with a README



### 6.2 会出现命令代码

如下:

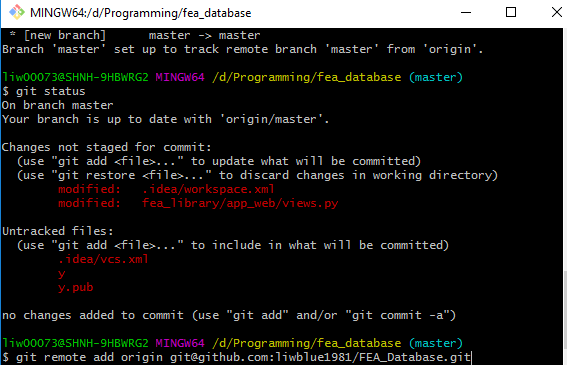


### 6.3 与服务器进行连接

在本地目录下的Git Bash Here窗口中输入

git remote add origin [git@github.com:liwblue1981/FEA\_Database.git](mailto:git@github.com:liwblue1981/FEA_Database.git)

用于建立远程链接



### 6.4 上传文件

git push -u origin master

## 7. 问题解决:

### 7.1 git@github.com: Permission denied (publickey)

出现这个错误的原因是客户端与服务端未生成 ssh key

客户端与服务端的ssh key不匹配.

输入命令:

git config --global –list

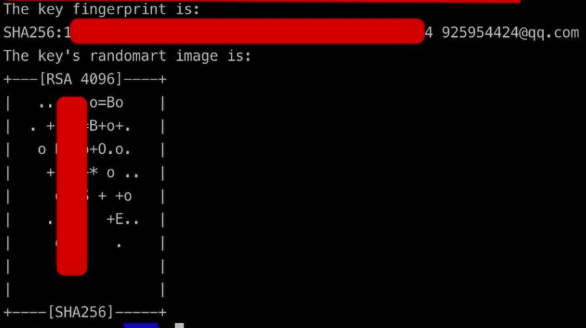
用于确认当前用户的注册邮箱是否与GitHub的邮箱一致.如果不一致,那么在步骤四中设置正确的邮箱和用户.

现在重新生成秘钥

### 7.2 客户端生成SSH KEY

ssh-keygen -t rsa -C [youremail@example.com](mailto:youremail@example.com)

一路回车即可，会生成你的ssh key(会保存在回车时候的路径: Enter file in which to save the key (/c/Users/liw00073/.ssh/id\_rsa))



### 7.3 执行终端命令

ssh -v [git@github.com](mailto:git@github.com)

如果出现:

No more authentication methods to try.

Permission denied (publickey).

那么在终端再次执行

### 7.4 ssh-agent -s

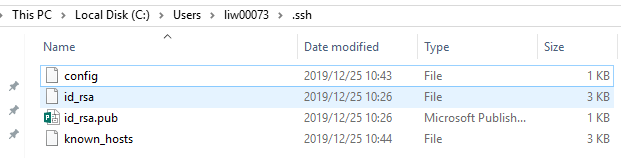
这个命令还有可能报错,错误信息为

Could not open a connection to your authentication agent

### 7.5 ssh-agent bash

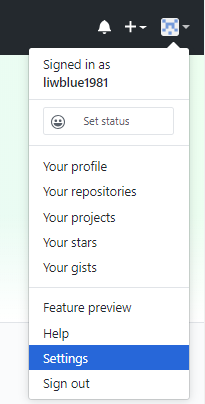
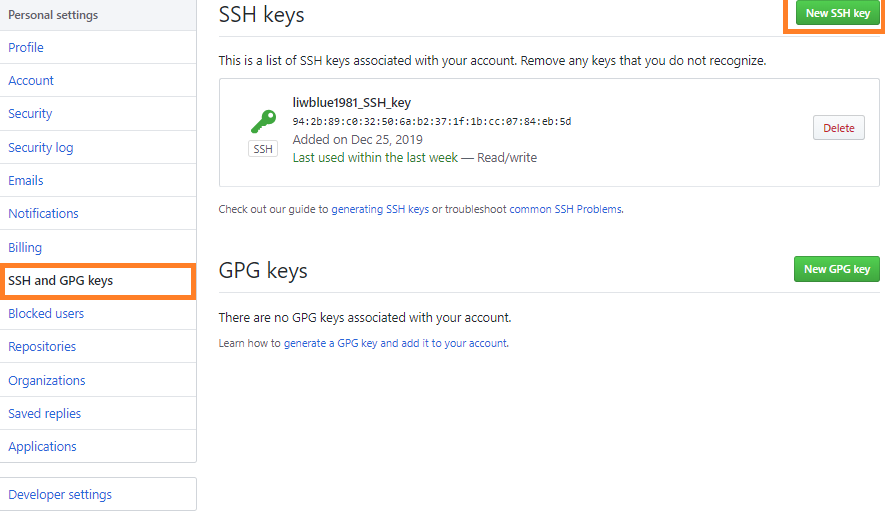
### 7.6 执行ssh-add ~/.ssh/id\_rsa

在~/.ssh下可以看到出现了id\_rsa文件



### 7.7 配置服务端

打开id\_rsa.pub,复制里面的内容,进入github账户,在setting下,来进行设置

### 7.8 验证

ssh -T [git@github.com](mailto:git@github.com)

得到结果为:

Hi liwblue1981! You've successfully authenticated, but GitHub does not provide shell access.

代表验证通过,可以开始准备上传文件到服务器

### 7.9 上传文件

可能出现如下错误

ssh: connect to host ssh.github.com port 22: Connection timed out

原因在于端口可能已经被占用.

### 7.10 更改配置文件

那么此时需要在/c/Users/liw00073/.ssh目录下新建一个配置文件

在config文件中复制下面代码：

*Host github.com*

*User wei\_dora\_hennie@aliyun.com*

*Hostname ssh.github.com*

*PreferredAuthentications publickey*

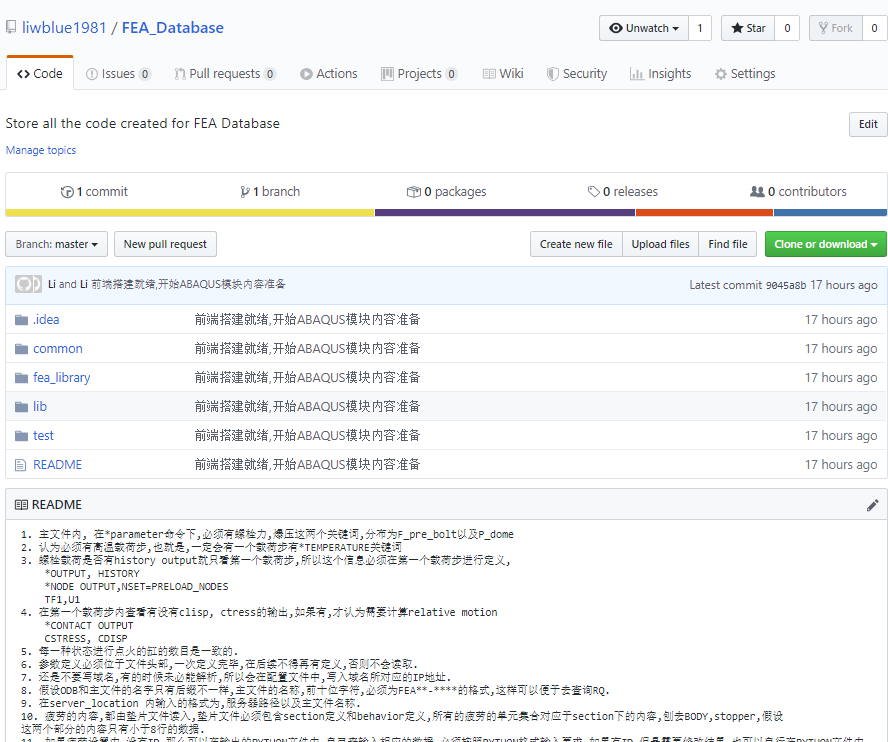
*IdentityFile ~/.ssh/id\_rsa*

*Port 443*

然后在执行下面命令查看是否可以链接

ssh -T git@github.com

再次执行6.5就可以上传文件了,最终结果为:



ABAQUS的PYTHON文件处理

关于导包,因为ABAQUS是PYTHON2.7的版本,所以目录框架下有多个文件需要导入时,需要注意:

1. 必须是包,也就是必须有\_\_init\_\_.py文件.

2. 不支持相对导入,只能采用绝对导入,

如果你的模块的名称是\_\_main\_\_，那么它就不被认为是在一个包当中，因为它的名称当中不含有“点”，所以你不能在它的里面使用from .. import。如果你使用了这个语句，那么程序就会报“relative-import in non-package"错误。

3. 文档注释不能有汉字,只能采用英语

4. 对于Python2,在定义基类的时候,必须继承object.