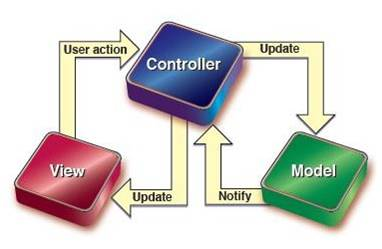
Django - MTV

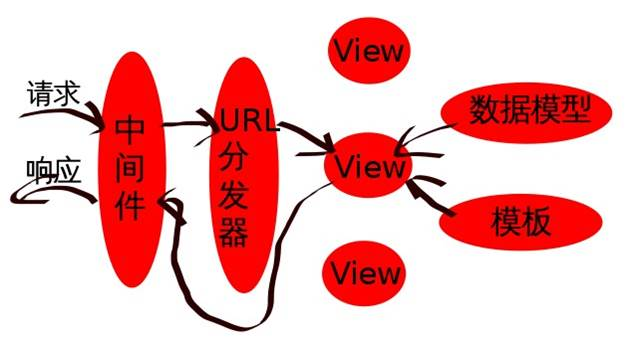
Web服务器开发领域里著名的MVC模式，所谓MVC就是把Web应用分为模型(M)，控制器(C)和视图(V)三层，他们之间以一种插件式的、松耦合的 方式连接在一起，模型负责业务对象与数据库的映射(ORM)，视图负责与用户的交互(页面)，控制器接受用户的输入调用模型和视图完成用户的请求，把数据存取逻辑、业务逻辑和表现逻辑组合在一起.在这个模式中， Model 代表数据存取层，View 代表的是系统中选择显示什么和怎么显示的部分，Controller 指的是系统中根据用户输入并视需要访问模型，以决定使用哪个视图的那部分。其示意 图如下所示：



Django的MTV模式本质上和MVC是一样的，也是为了各组件间保持松耦合关系，只是定义上有些许不同，Django的MTV分别是指：

M 代表模型（Model）：负责业务对象和数据库的关系映射(ORM)。即数据存取层。 该层处理与数据相关的所有事务： 如何存取、如何验证有效性、包含哪些行为以及数据之间的关系等。  
T 代表模板 (Template)：负责如何把页面展示给用户(html)。即表现层。 该层处理与表现相关的决定： 如何在页面或其他类型文档中进行显示。  
V 代表视图（View）：负责业务逻辑，并在适当时候调用Model和Template。即业务逻辑层。 该层包含存取模型及调取恰当模板的相关逻辑。 你可以把它看作模型与模板之间的桥梁。

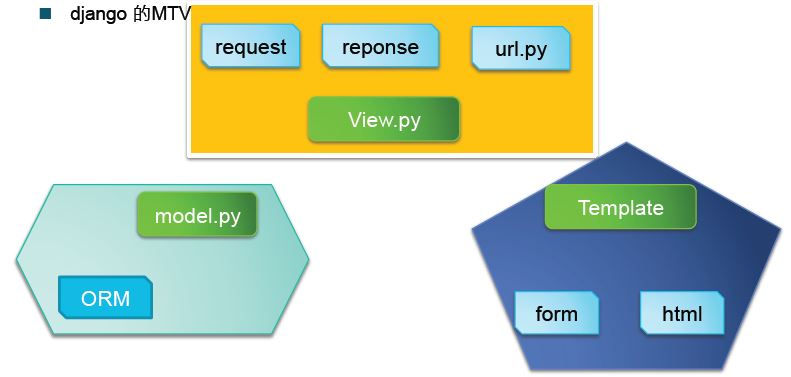
除了以上三层之外，还需要一个URL分发器，它的作用是将一个个URL的页面请求分发给不同的View处理，View再调用相应的Model和Template，MTV的响应模式如下所示：



1.Web服务器（中间件）收到一个http请求  
2.Django在URLconf里查找对应的视图(View)函数来处理http请求  
3.视图函数调用相应的数据模型来存取数据、调用相应的模板向用户展示页面  
4.视图函数处理结束后返回一个http的响应给Web服务器  
5.Web服务器将响应发送给客户端

这种设计模式关键的优势在于各种组件都是松耦合的。这样，每个由 Django驱动的Web应用都有着明确的目的，并且可独立更改而不影响到其它的部分。 比如，开发者更改一个应用程序中的 URL 而不用影响到这个程序底层的实现。设计师可以改变 HTML页面的样式而不用接触Python代码。数据库管理员可以重新命名数据表并且只需更改模型，无需从一大堆文件中进行查找和替换。

落到实处，Django的MTV模式相对应的python文件如下：



project app

project 和 app 之间到底有什么不同呢？它们的区别就是一个是配置另一个是 代码：一个project包含很多个Django app以及对它们的配置。

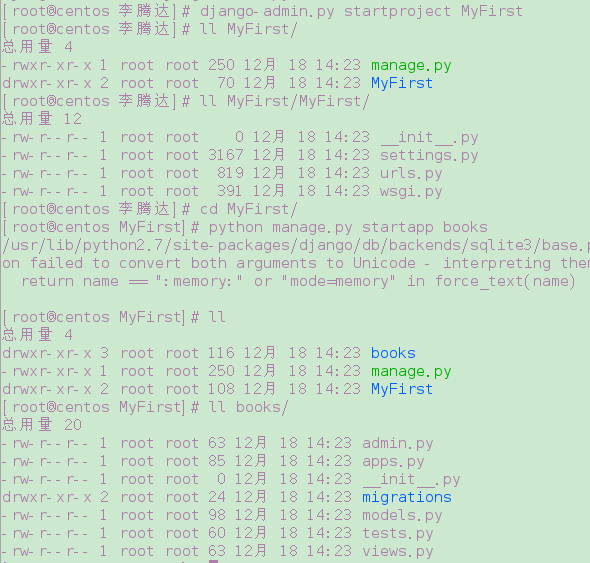
技术上，project的作用是提供配置文件，比方说哪里定义数据库连接信息, 安装的app列表， TEMPLATE\_DIRS ，等等。一个app是一套Django功能的集合，通常包括模型和视图，按Python的包结构的方式存在。例如，Django本身内建有一些app，例如注释系统和自动管理界面。 app的一个关键点是它们是很容易移植到其他project和被多个project复用。

在任意一个目录下，创建一个名为MyFirst的project，在MyFirst下创建一个名为books的app

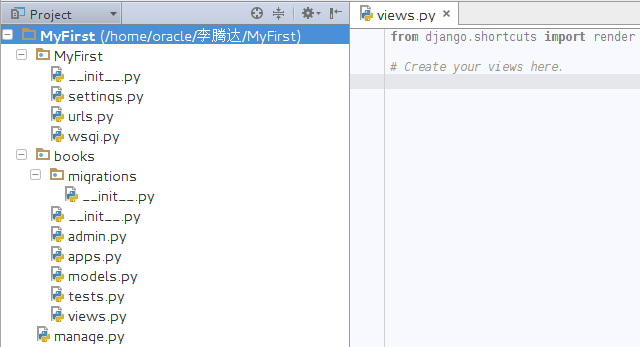
django-admin.py startproject MyFirst

cd MyFirst/

python manage.py startapp books



我们用pycharm打开这个project



models

在settings.py配置mysql数据库连接

DATABASES = {

**'default'**: {

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

**'NAME'**:**'guanye'**,

**'USER'**:**'root'**,

**'PASSWORD'**:**'123qweASDzxc!'**,

**'HOST'**:**'localhost'**,

**'PORT'**:**'3306'**,

*# 'OPTIONS':{*

*# 'init\_command':'SET storage\_engine=INNODB',*

*# }*

*# 'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),*

}

}

在models.py代码如下

*#--coding=utf8--*

*#from \_\_future\_\_ import unicode\_literals*

**from** django.db **import** models

*# Create your models here.*

*#创建一个名为tb\_user的表*

**class** tb\_user(models.Model):

*#(fields.W122) 'max\_length' is ignored when used with IntegerField*

*# user\_id = models.BigIntegerField(max\_length=20,null=None,primary\_key=True,unique=True)*

user\_id = models.BigIntegerField(null=None,primary\_key=True,auto\_created=True)

unionid = models.CharField(**'微信unionid'**,max\_length=32,null=None)

mobile = models.CharField(**'手机号'**,max\_length=11,null=None,default=**'Null'**)

password = models.CharField(**'用户密码'**,max\_length=64,null=None,default=**'Null'**)

pay\_password = models.CharField(**'支付密码'**,max\_length=32,default=**'Null'**)

bank\_card\_num = models.BooleanField(**'绑定的银行卡数量'**,max\_length=1,null=None,default=0)

level = models.BooleanField(**'用户当前等级（阶段）'**,max\_length=2,null=None,default=0)

reg\_ip = models.CharField(**'注册IP'**,max\_length=80,null=None)

*#'max\_length' is ignored when used with IntegerField*

from\_channel = models.SmallIntegerField(**'推广渠道'**,null=None,default=0)

from\_reg = models.BooleanField(**'注册来源；0为wap；1为android；2为ios'**,max\_length=1,null=None,default=0)

*#'max\_length' is ignored when used with IntegerField*

from\_activity = models.SmallIntegerField(**'活动，关联活动表'**,null=None,default=0)

realname = models.CharField(**'真实姓名'**,max\_length=20,null=True)

nick = models.CharField(**'用户昵称'**,max\_length=100,null=True)

card\_no = models.CharField(**'身份证号'**,max\_length=18,null=True)

is\_auth = models.BooleanField(**'1 认证，** **0未证'**,max\_length=1,null=None,default=0)

safety\_status = models.BooleanField(**'用户登录安全状态；1需要授权；0反之'**,max\_length=1,null=None,default=1)

*#(fields.W122) 'max\_length' is ignored when used with IntegerField*

*# login\_count = models.IntegerField(max\_length=8,null=None,default='0')*

login\_count = models.IntegerField(**' 登录次数'**,null=None,default=0)

login\_time = models.IntegerField(**'登陆时间'**,null=None,default=**'Null'**)

create\_time = models.IntegerField(**'创建时间'**,null=None,default=**'Null'**)

update\_time = models.IntegerField(**'更新时间'**,null=True,default=**'Null'**)

每个数据模型都是 django.db.models.Model 的子类。它的父类 Model 包含了所有必要的和数据库交互的方法，并提供了一个简洁漂亮的定义数据库字段的语法。每个模型相当于单个数据库表，每个属性也是这个表中的一个字段。 属性名就是字段名，它的类型（例如 CharField ）相当于数据库的字段类型 （例如 varchar ）。

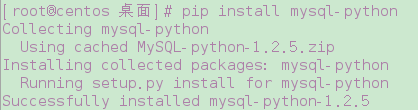
再次编辑 settings.py 文件， 找到 INSTALLED\_APPS 设置。 INSTALLED\_APPS 告诉 Django 项目哪些 app 处于激活状态。 缺省情况下如下所示：



连接mysql

yum install mysql-devel python-devel

pip install mysql-python



在PycharmProject/settings.py中修改如下：

DATABASES = {

**'default'**: {

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

**'NAME'**:**'guanye'**,

**'USER'**:**'root'**,

**'PASSWORD'**:**'123qweASDzxc!'**,

**'HOST'**:**'localhost'**,

**'PORT'**:**'3306'**,

*# 'OPTIONS':{*

*# 'init\_command':'SET storage\_engine=INNODB',*

*# }*

*# 'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),*

}

}

在settings.py的INSTALLED\_APPS 项加上APP



编写models.py如下

*#--coding=utf8--*

*#from \_\_future\_\_ import unicode\_literals*

**from** django.db **import** models

*# Create your models here.*

*#创建一个名为tb\_user的表*

**class tb\_user**(models.Model)**:**

*#(fields.W122) 'max\_length' is ignored when used with IntegerField*

*# user\_id = models.BigIntegerField(max\_length=20,null=None,primary\_key=True,unique=True)*

user\_id **=** models.BigIntegerField(null**=**None,primary\_key**=**True,unique**=**True)

open\_id **=** models.CharField(max\_length**=**32,null**=**None)

mobile **=** models.CharField(max\_length**=**11,null**=**None,default**=**Null)

password **=** models.CharField(max\_length**=**64,null**=**None,default**=**Null)

reg\_ip **=** models.CharField(max\_length**=**80,null**=**None)

realname **=** models.CharField(max\_length**=**20,null**=**True)

nick **=** models.CharField(max\_length**=**100,null**=**True)

card\_no **=** models.CharField(max\_length**=**18,null**=**True)

is\_auth **=** models.BooleanField(max\_length**=**1,null**=**None,default**=**0)

*#(fields.W122) 'max\_length' is ignored when used with IntegerField*

*# login\_count = models.IntegerField(max\_length=8,null=None,default='0')*

login\_count **=** models.IntegerField(null**=**None,default**=**0)

login\_time **=** models.IntegerField(null**=**None,default**=**Null)

create\_time **=** models.IntegerField(null**=**None,default**=**Null)

update\_time **=** models.IntegerField(null**=**True,default**=**Null)

检查语法是否有错

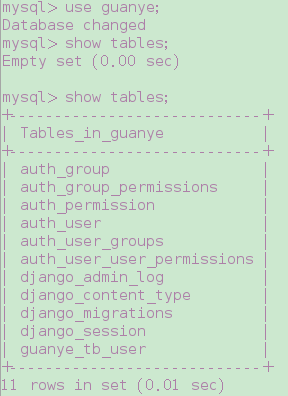
python manage.py check

python manage.py makemigrations guanye

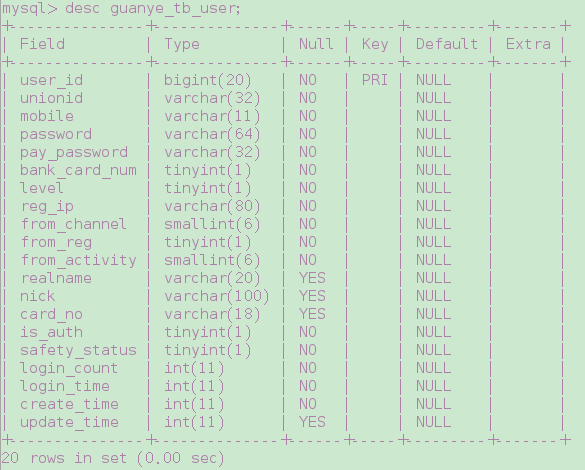
在shell环境将修改应用到数据库

python manage.py migrate

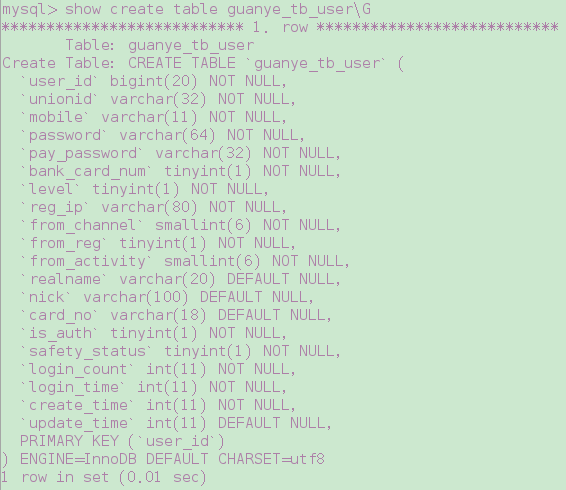
看看数据库是否有改变



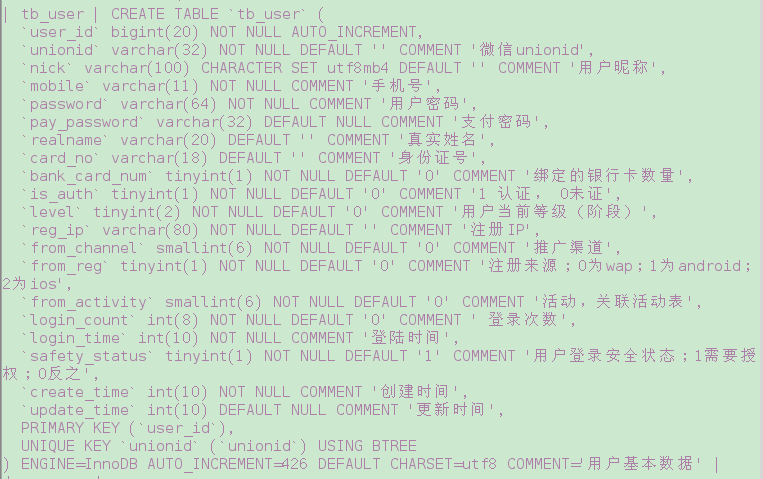
查看表结构



查看创建表的语句



这是真实的表结构



遗留问题：

1)字段无法添加描述

'描述'

2）字段无法指定默认值

is\_auth **=** models.BooleanField(max\_length**=**1,null**=**None,default**=**0)

2)*IntegerField*无法指定*max\_length*值

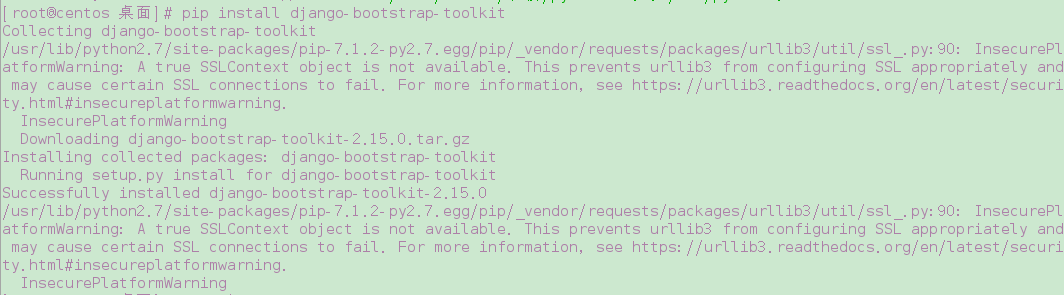
*login\_count = models.IntegerField(max\_length=8,null=None,default='0')*

*(fields.W122) 'max\_length' is ignored when used with IntegerField*

django-bootstrap

1.安装django-bootstrap-toolkit

pip install django-bootstrap-toolkit



2.在django使用forms

git

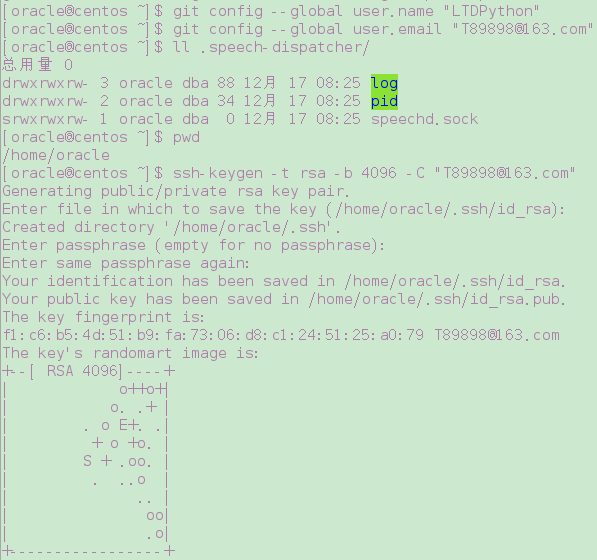
在github.com注册,用户名LTDPython,邮箱T89898@163.com,密码123qweasdzxc后，配置本地

git config --global user.name "LTDPython"

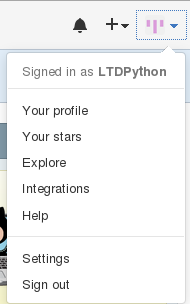
git config --global user.email "T89898@163.com"

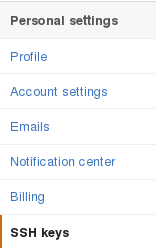


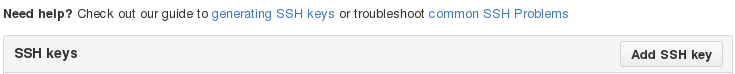
ssh-keygen -t rsa -b 4096 -C "T89898@163.com" #密码123qweasdzxc



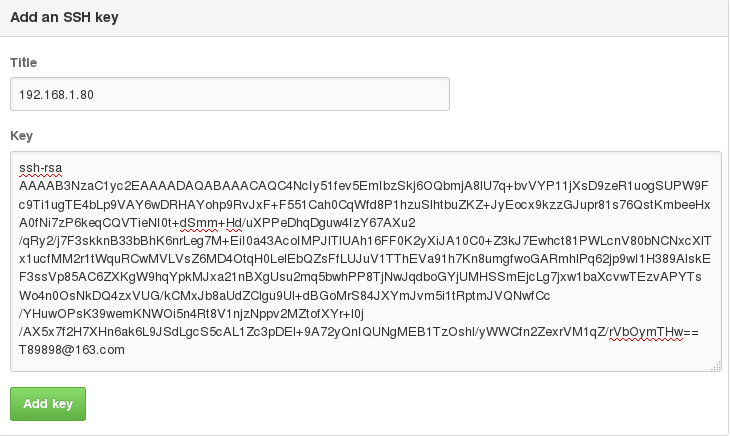
cat .ssh/id\_rsa.pub





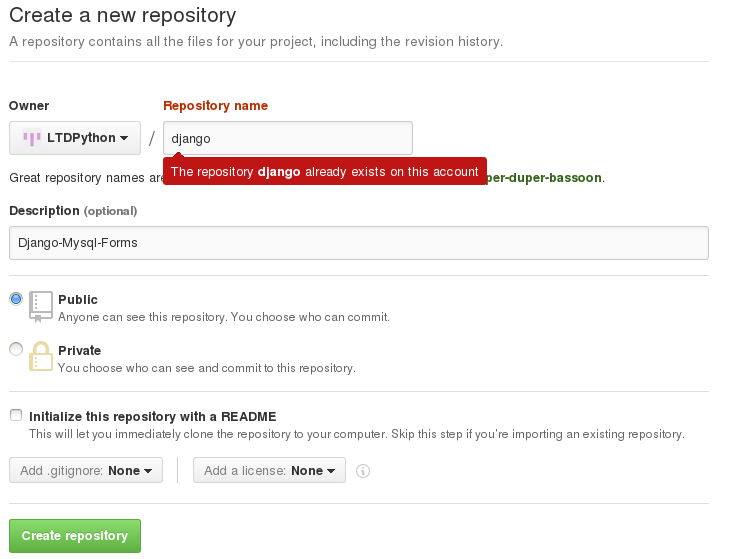


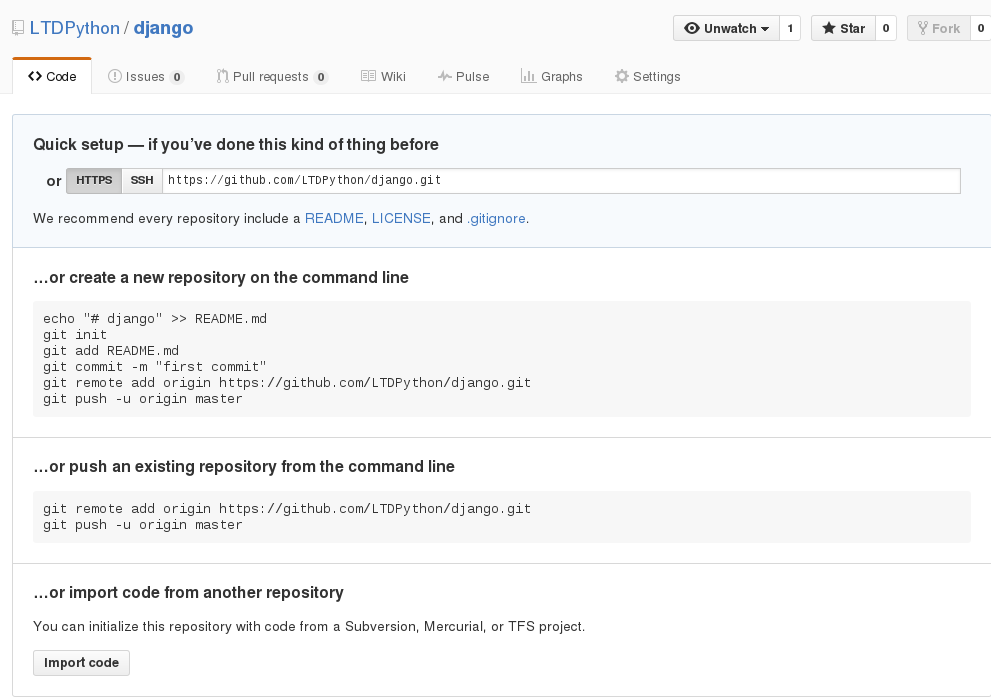
把~/.ssh/id-rsa.pub的内容粘贴进去

然后用ssh -T git@github.com命令验证

ssh -T git@github.com

在github账户上创建名为django的repository

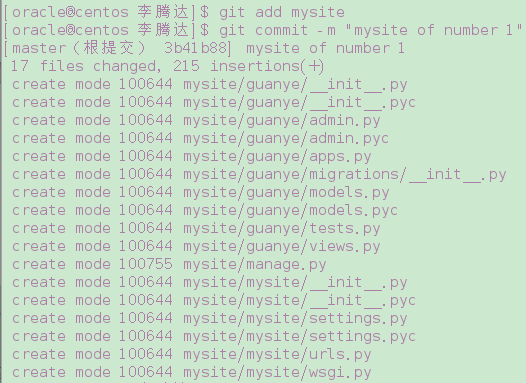
按照说明



把mysite文件夹添加到git缓冲区，并提交

git add mysit

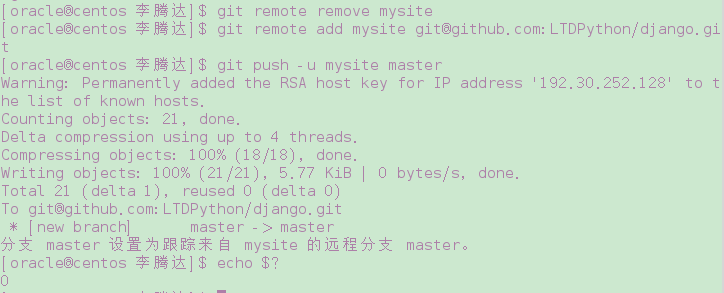
git commit -m "mysite of number 1



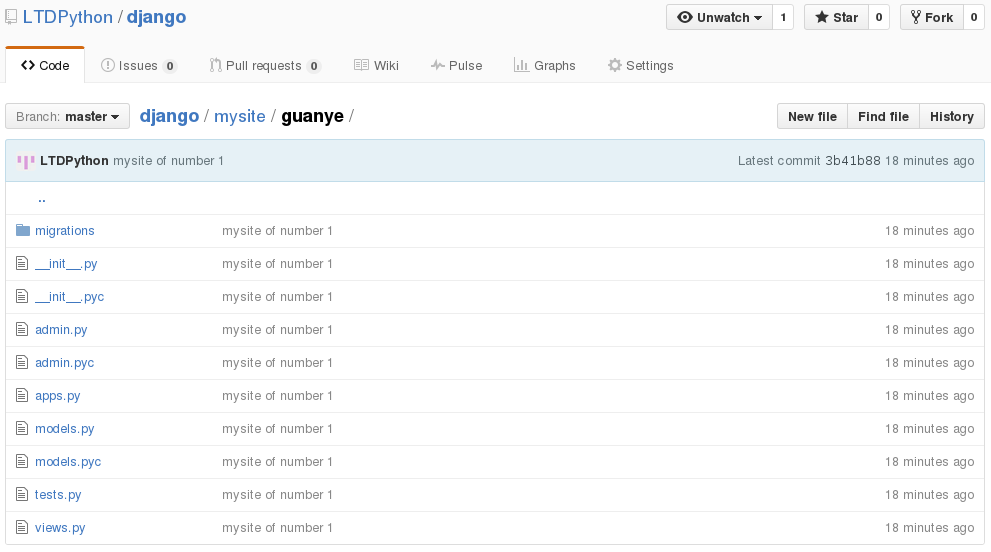
在本地remote add然后push

git remote add mysite git@github.com:LTDPython/django.git

git push -u mysite master

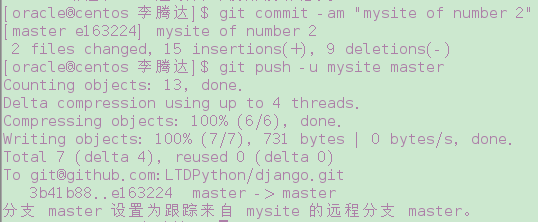


提交成功

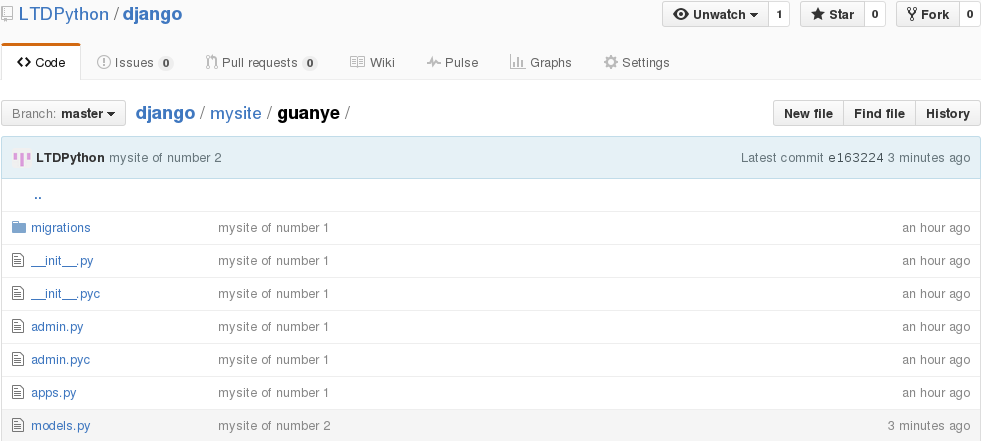
修改了些内容，先commit再push

git commit -am "mysite of number 2"

git push -u mysite master

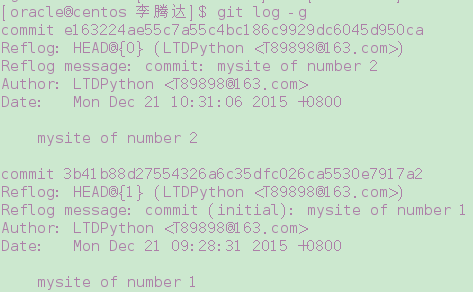


查看github上的改变

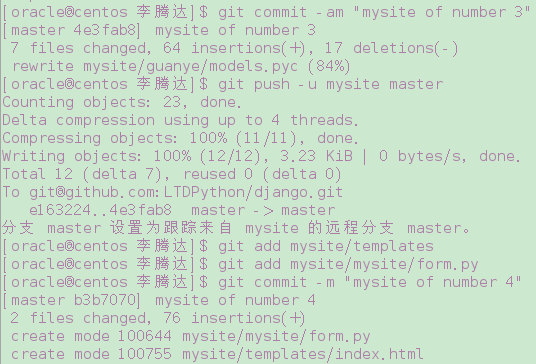


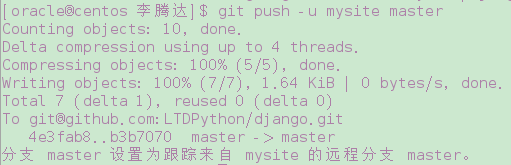
注意：-am是将add和commit -m两条命令合并了，只适合已经存在的文件被修改后提交。如果要提交新增的文件，需要先add 文 件，再comit -m

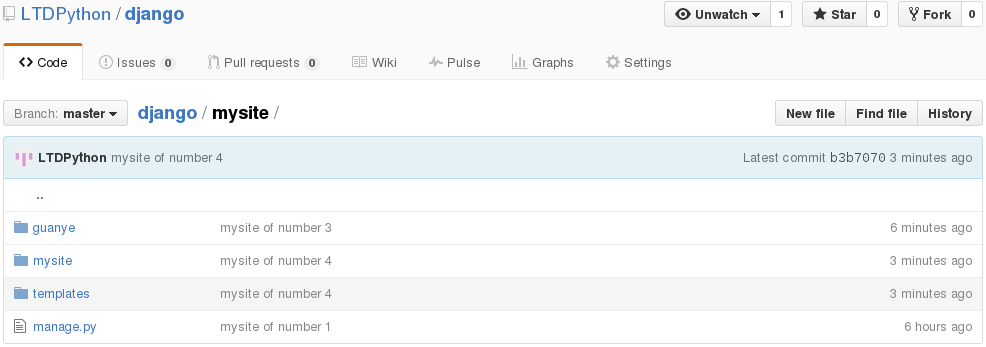
git log -g 查看提交记录

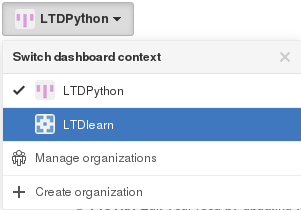


如果新建了文件，必须要add，才能commit





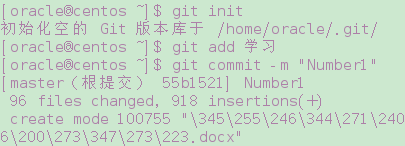
新建organizations



git init

git add 学习

git commit -m "Number1"



git remote add learn git@github.com:LTDlearn/LTD.git

git push -u learn master

