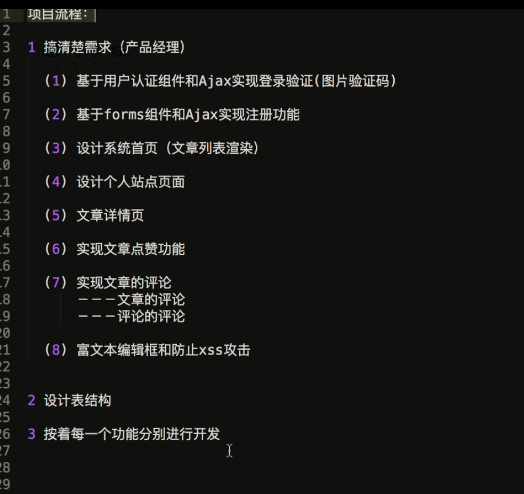
利用花生壳将个人电脑做服务器：

1.设置花生壳内网穿透

2.设置django 网点：0.0.0.0 端口

3.将个人电脑内网ip加入到setting中ALLOWED\_HOSTS

或者ALLOWED\_HOSTS=[‘\*’]



## 一、表结构设计



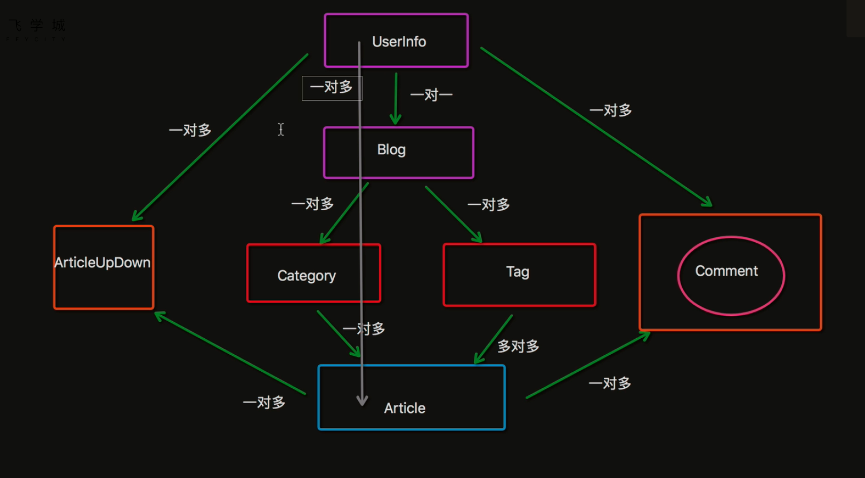
联合唯一

verbose\_name为指定字段名称。

自关联：



并且允许为空。



对于django中的用户认证（AUTH\_USER）表，如果觉得其字段不够用，可以新建一个用户表，继承该表即可。

但是注意：

1.现在建立表结构的文件中，导入AbstractUser模块。

2.在setting中声明

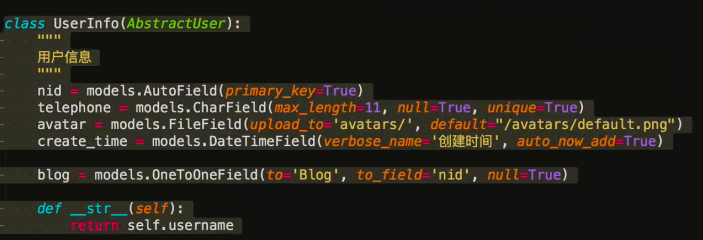
AUTH\_USER\_MODEL=‘blog.UserInfo’

UserInfo为新建的表名。blog为项目名。

导入 AbstractUser模块



创建自己的表



在setting中声明：



关于models.FileField类型中的upload\_to







## 二、登录页面

### （一）ajax、表单

bs中类名 pull-right 靠右显示

form-group 分组显示，组与组之间有间隔。

form-control 表单的显示。

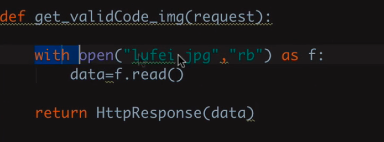
### （二）图形验证码



src地址为向改地址请求数据。

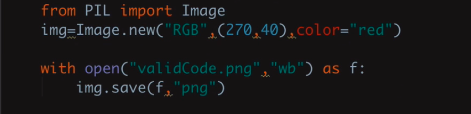
通过地址分配至视图函数。

视图函数读入图片数据，并用httpResponse返回给前端。



动态生成验证码：

安装pillow模块，导入Image模块



new里面的参数，RGB为颜色类型，（270,40）为宽高像素，red为背景色。

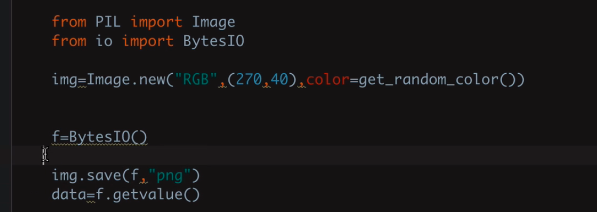
color也可以是一个元组，(255,255,255)为rgb的三元素。

然后新建一个validCode.png文件，写入。

img.save的第一个参数为写入到哪里，第二个参数为指定图片类型。

然后再将其读出来，发送给前端。

缺点：要存入磁盘，还要读出来，速度慢。



导入io模块中的BytesIO，

f=BytesIO()

直接存入内存中：img.save(f,’png’)

然后从内存中读取:f.getvalue()

速度快很多。

此时生成的是一个纯背景色的图片。

最终版：

**from** PIL **import** Image, ImageDraw, ImageFont  
**from** io **import** BytesIO  
**import** random  
  
img = Image.new(**"RGB"**, (270, 40), color=get\_random\_color()) #得到一个纯色背景的图片对象。  
  
draw = ImageDraw.Draw(img) #得到一个ImageDraw对象  
kumo\_font = ImageFont.truetype(**"static/font/kumo.ttf"**, size=32) #得到一个字体对象  
  
valid\_code\_str = **""  
for** i **in** range(5): #将验证码写入图片，并将验证码放入valid\_code\_str  
 random\_num = str(random.randint(0, 9))  
 random\_low\_alpha = chr(random.randint(95, 122))  
 random\_upper\_alpha = chr(random.randint(65, 90))  
 random\_char = random.choice([random\_num, random\_low\_alpha, random\_upper\_alpha])  
 draw.text((i \* 50 + 20, 5), random\_char, get\_random\_color(), font=kumo\_font)  
  
 *# 保存验证码字符串* valid\_code\_str += random\_char  
  
width=270  
height=40  
**for** i **in** range(10): #画线条  
 x1=random.randint(0,width)  
 x2=random.randint(0,width)  
 y1=random.randint(0,height)  
 y2=random.randint(0,height)  
 draw.line((x1,y1,x2,y2),fill=get\_random\_color())  
  
**for** i **in** range(100): #画短弧线  
 draw.point([random.randint(0, width), random.randint(0, height)], fill=get\_random\_color())  
 x = random.randint(0, width)  
 y = random.randint(0, height)  
 draw.arc((x, y, x + 4, y + 4), 0, 90, fill=get\_random\_color())  
  
print(**"valid\_code\_str"**, valid\_code\_str)  
  
request.session[**"valid\_code\_str"**] = valid\_code\_str #将生成的字符串保存在session中，便于与用户输入的验证码核对。  
  
**'''**

**保存session是的三个步骤  
1 sdajsdq33asdasd 生成一个随机字符串  
2 COOKIE {"sessionid":sdajsdq33asdasd} 在cookie中保存该字符串，键为sessionid**

**3.django-session 在数据库中django-session表中进行保存。  
 session-key session-data  
 sdajsdq33asdasd {"valid\_code\_str":"12345"}  
  
  
'''**f = BytesIO()  
img.save(f, **"png"**)  
data = f.getvalue()  
  
  
**return** HttpResponse(data)

对于点击刷新验证码的方法：

可以直接使用jqery：

<**script**>  
 **$**(**function** () {  
 **$**(**'#verifycode'**).click(**function** () {  
 **$**(**this**)[0].**src** += **'?'** })  
 })  
</**script**>

在图片的src后加个 ？ ，这样就会向服务端重新发src的请求，获取新的验证码。

#### 浏览器前端跳转：

location.herf=’url’

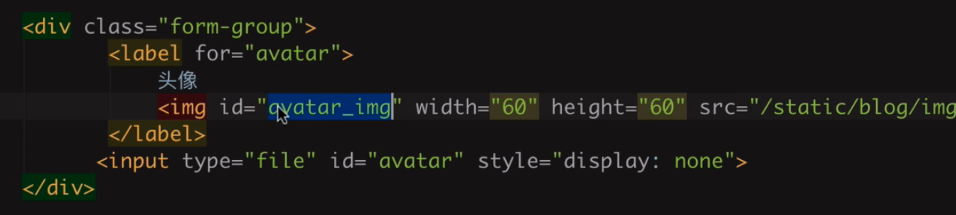
对于request中的内容，前端可以直接通过模板语法拿来用，而不用后端进行特定传值。



### （三）注册

1.利用form模块做

2.上传头像，将input type=file 隐藏，相关的label中嵌套入img，形成点击图片上传。



3.头像预览

在上传完头像后，进行预览。

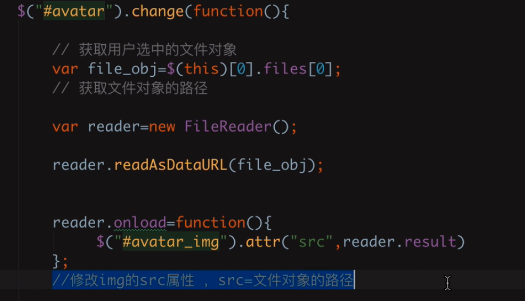
对input绑定change事件。

input 的id为avatar

则$(‘#avatar’)[0].files[0]为上传的文件对象

$(‘#avatar’)[0].files[0]

然后利用文件阅读器FileReader，读出文件的url数据。还可以读文件的二进制、文本、等数据。



得出的数据在reader.result中存放。

由于读数据的速度较慢，所以改变img地址属性要等读完之后进行。因此使用onload事件。

#### 上传的数据分用户存放：

当然，如果觉得只是避免文件路径重名，还是不能满足你，其实，django还允许你重写一个upload\_to函数，重定义上传文件的路径前缀

# models.py

#让上传的文件路径动态地与user的名字有关

def upload\_to(instance, fielname):

return '/'.join([MEDIA\_ROOT, instance.user\_name, filename])

class User(models.Model):

avatar = ImageField(upload\_to = upload\_to)

user\_name = CharField(max\_length = 250)

### （四）首页

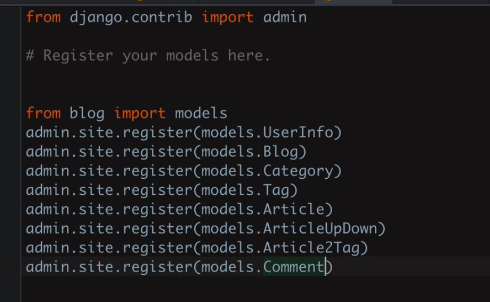
### 利用admin录入数据

利用admin 的web页面管理数据表

url为admin

登录必须为超级用户

要管理的数据表需要在项目下的admin.py文件中进行注册。



img

{

vertical-align:text-top;

}

微调图片的上下位置，可以用 负值。

#### extra



mysql中不用strftime，而是用date\_format(‘字段名’,’%%Y-%%m’)

## 评论

### 事务性操作：

即两个或者多个操作关联度高，执行时需要同时执行，如果失败，则自动回滚到所有操作未执行时的状态，而不是有的执行有的没有执行。即将这些操作看为一组操作，如果执行则必须都执行， 如果没有执行，则必须都没有执行。如转账双方一个账户减少，一个账户增加。

调用



对想进行事务性操作的语句上方写入固定语句：



即可。

### 发邮件

引入模块：

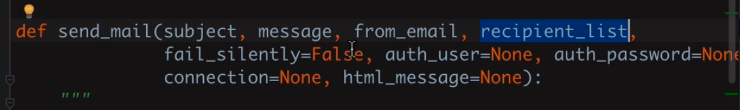


在setting中设置发送方邮箱：



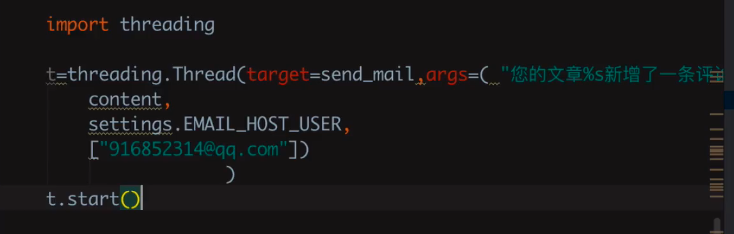
端口号中：163邮箱的端口号为25

使用send\_mail（）方法



各参数：send\_mail(主题，内容，发件邮箱，目标邮箱列表，)

由于发邮件比较慢，所以一般会再开一个线程来发邮件



## 后台管理

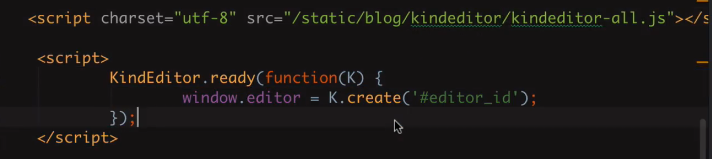
### 富文本编辑器

使用kindeditor编辑器：

1.去官网下载文件

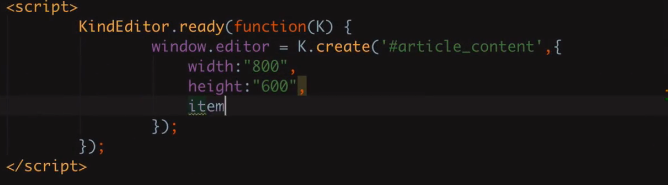
2.将文件放在static目录中

3.在html中引入。



此处 #editor\_id 为需要放入编辑器的文本框的id。也可以是class等选择器

create后还可以跟参数，对编辑器进行设置。



[kniteditor的使用方法](http://kindeditor.net/doc.php)

### 编辑器图片上传

#### 1.在编辑参数中先进行设置：

（1）uploadjson：指定上传文件的url。

（2）[extraFileUploadParams](http://kindeditor.net/docs/option.html#id111)：由于上传是post操作，所以需要传入csrf参数，也可以传入你想传入的任意参数。

上传图片、Flash、视音频、文件时，支持添加别的参数一并传到服务器。

数据类型: Array

默认值: {}

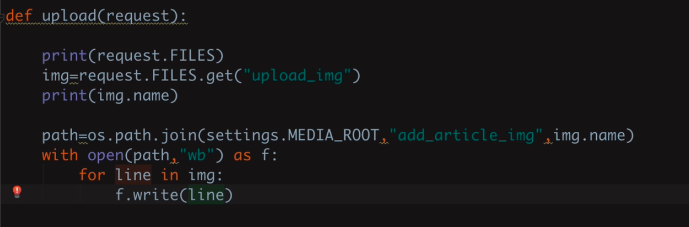
（3）[filePostName](http://kindeditor.net/docs/option.html#id112) 默认值为imgFile 可以缺省。在服务端取文件时，用的就是这个键值。request.FILES[‘imgFile’]或者request.FILES,get(‘imgFile’)

指定上传文件form名称。

数据类型: String

默认值: imgFile

#### 2.在后端进行保存



此时设置的filePostName为upload\_img

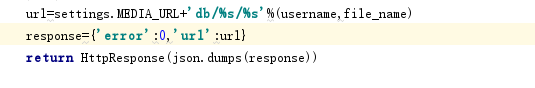
img.name 是上传的文件名。

#### 3.在前端进行显示

将保存的图片的url返回至前端，即可显示。

由于media路径是已经在settings设置好，所以将后续的地址接上就可以。

以字典形式返回，键值为url。



前端会自动将这个地址填入到img标签中的src中，显示出图片。

#### 文章简介

由于文章正文里含有各种html标签，所以简介不能从正文中直接截取。

引入



BeautifulSoup对标签进行解析，获得文本部分。



第一行为提取文本，第二行为文本截断。

### 基于bs4模块防御xss攻击



find\_all() 是提取soup中所有的标签对象，是[obj,obj,…]的对象放入列表的形式。

对象有name属性，为标签名称。

对象有decompose()方法，为删除掉soup对象中该标签及其所含的内容。

注意，虽然是tag.decompose()，但是soup对象中该内容是被删除掉了

写入数据库时，要使用拦截后的信息，即str(soup)

从5月22日到7月6日之间跌幅最大的，

从7月22日到8月6日之间跌幅最大的

DIFF:= EMA(CLOSE,SHORT) - EMA(CLOSE,LONG);

DEA := EMA(DIFF,M);

MACD:= 2\*(DIFF-DEA);

T:=BARSLAST(MACD=0);

S:=REF(BARSLAST(MACD=0),T+1)+T+1;

k:=MA(CLOSE,5);

j:=MA(CLOSE,10);

k<j AND T<3 AND MACD<0;