一、图形验证码

1 安装django-simple-captcha库

在网站开发的登录页面中，经常会需要使用到图形验证码来验证。在Django中，django-simple-captcha库包提供了图形验证码的使用。

```
$ pip install django-simple-captcha

# 如果安装有依赖库问题，请执行下面的安装
apt-get -y install libz-dev libjpeg-dev libfreetype6-dev python-dev
```

2 设置

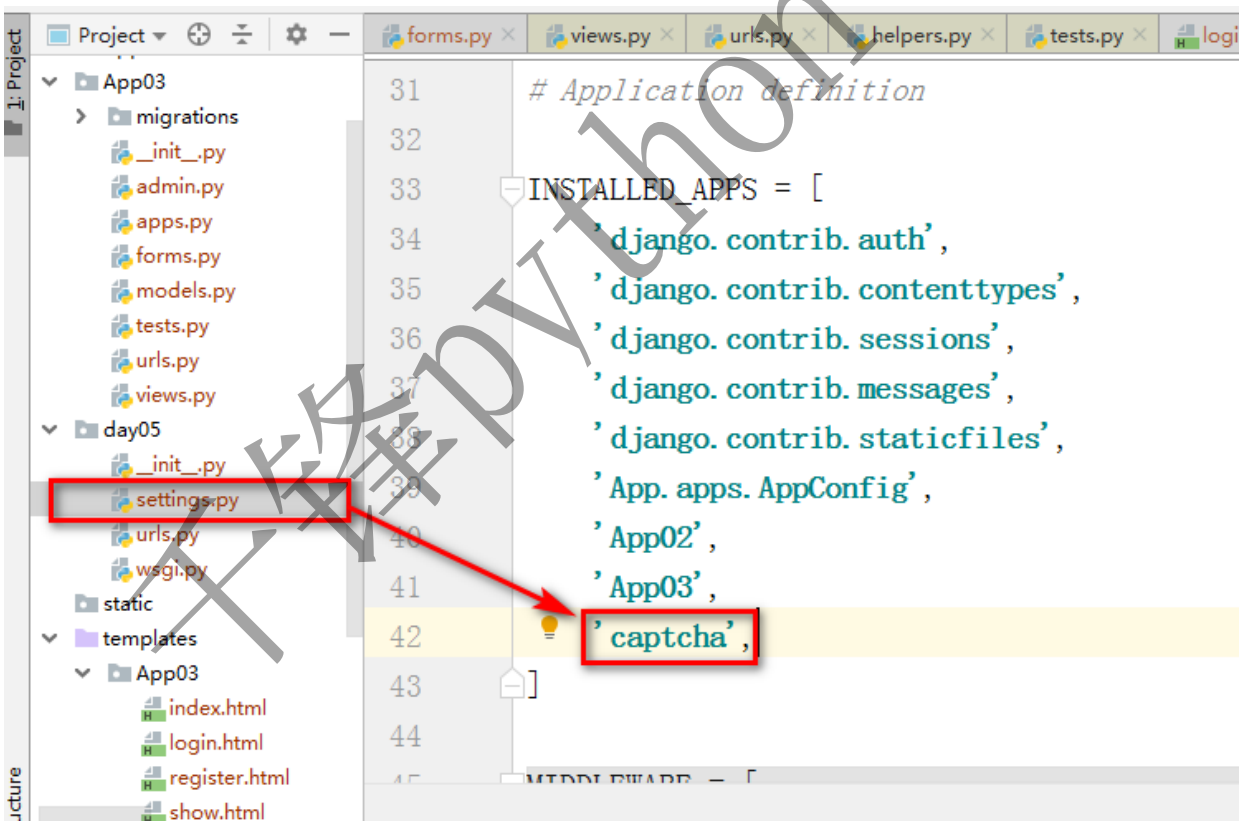

图1. 安装应用

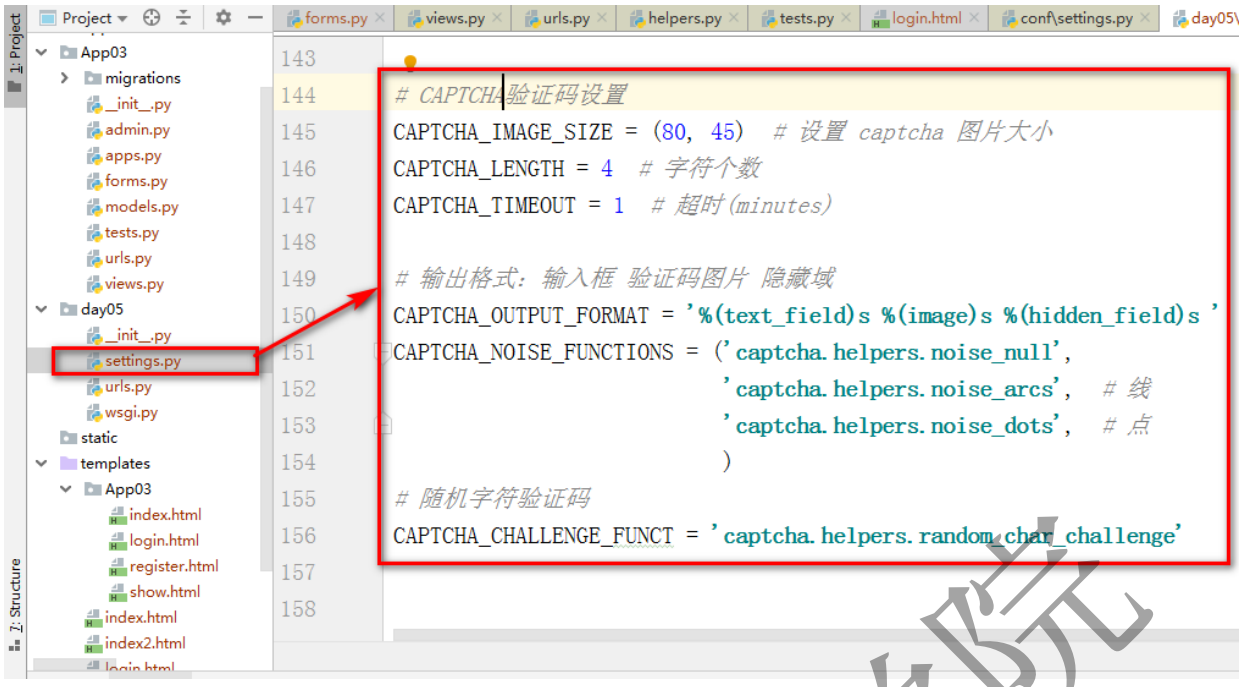

图2. 设置验证码样式

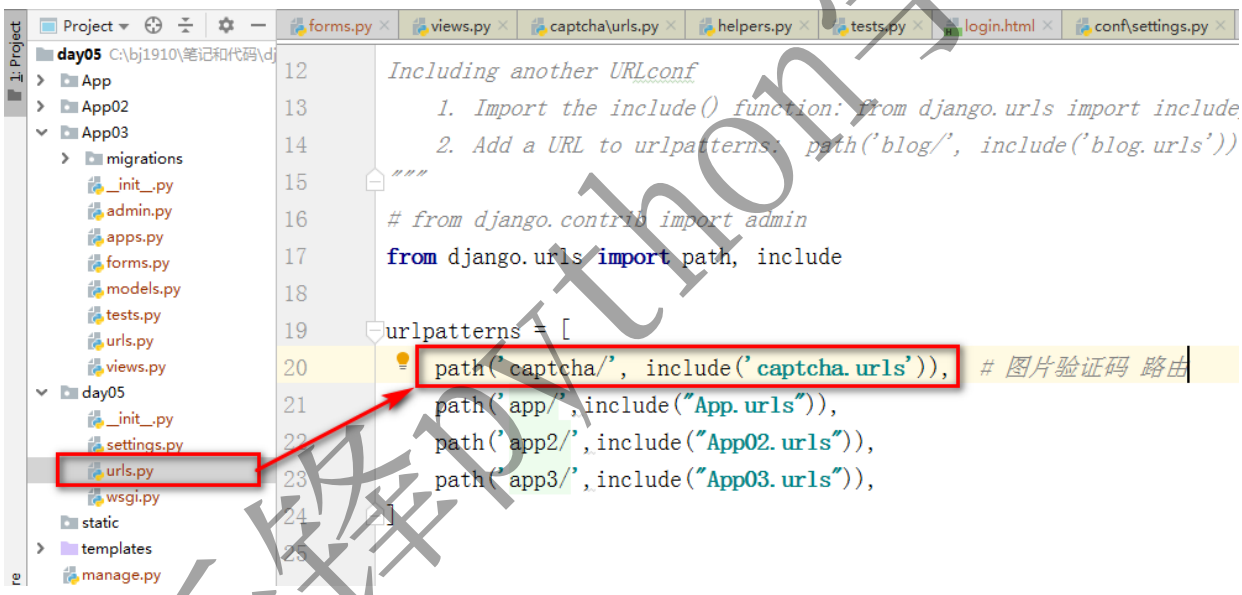

图3. 设置路由

最后要迁移数据库:

```
python manage.py migrate
```

3 建立表单

```
# forms.py
from django import forms
from captcha.fields import CaptchaField
class LoginForm(forms.Form):
    username = forms.CharField(max_length=20,min_length=3)
    password =
forms.CharField(max_length=128,widget=forms.PasswordInput())
    captcha = CaptchaField() # 验证码字段
```

4 实现

```
# 应用的urls.py
urlpatterns = [
    .....
    path('yzm/',views.user_login,name='yzm'),
]

# 前端页面
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>登录</title>
</head>
<body>
<div>{{ msg }}</div>
<form action="{% url 'App03:yzm' %}" method="post">
    {% csrf_token %}
    用户：{{ form.username }} <span>{{ form.username.errors.0 }}
</span> <br>
    密码：{{ form.password }} <span>{{ form.password.errors.0 }}
</span><br>
    验证码：{{ form.captcha }} <span>{{ form.captcha.errors.0 }}
</span><br>
    <input type="submit">
</form>
</body>
</html>
<script src="https://cdn.bootcss.com/jquery/1.12.3/jquery.min.js">
</script>
<script>
    //点击刷新验证码
```

```
    $(function () {
        $('.captcha').css({
            'cursor': 'pointer'
        });
        // ajax刷新
        $('.captcha').click(function () {
            console.log('click');
            $.get("/app3/refresh/",
                function (result) {
                    $('.captcha').attr('src', result['image_url']);
                    $('#id_captcha_0').val(result['key'])
                });
        });
    })
</script>

# views.py
import json

from captcha.helpers import captcha_image_url
from captcha.models import CaptchaStore
from django.contrib.auth import authenticate
import django.contrib.auth as auth
from django.contrib.auth.decorators import login_required
from django.http import HttpResponse, JsonResponse
from django.shortcuts import render, redirect

def user_login(request):
    if request.method == "POST":
        form = LoginForm(request.POST)
        if form.is_valid():
            username = form.cleaned_data.get('username')
            password = form.cleaned_data.get('password')
            user =
authenticate(request,username=username,password=password)
            if user:
                auth.login(request,user)
                return redirect(reverse("App03:home"))
    else:
        form = LoginForm()
    # 跳转登录页面
    return render(request,'App03/login.html',context={'form':form})
```

二、发送邮件

1.setting配置

```
# smtp服务的邮箱服务器
EMAIL_HOST = 'smtp.163.com'
# smtp服务固定的端口是25
EMAIL_PORT = 25
#发送邮件的邮箱
EMAIL_HOST_USER = 'landmark_cheng@163.com'
#在邮箱中设置的客户端授权密码
EMAIL_HOST_PASSWORD = 'q123456'
#收件人看到的发件人 <此处要和发送邮件的邮箱相同>
EMAIL_FROM = 'python<landmark_cheng@163.com>'
```

2.发送邮件

```
#一封邮件
from django.core.mail import send_mail
from django.conf import settings
def sendone(request):
    send_mail('标题', '内容', settings.EMAIL_FROM,
              ['313728420@qq.com'])
    return HttpResponse("发一封邮件")

# 发多封邮件
def sendmany(request):
    message1 = ('Subject here', '<b>Here is the message</b>',
settings.EMAIL_FROM, ['313728420@qq.com'])
    message2 = ('Subject here', '<b>Here is the message</b>',
settings.EMAIL_FROM, ['313728420@qq.com'])
    send_mass_mail((message1,message2), fail_silently=False)
    return HttpResponse('发送多封邮件')

#渲染模板进行邮件发送
def send_mail(request):
   subject, from_email, to = 'html', settings.EMAIL_FROM,
'313728420@qq.com'
```

```
    html_content =
loader.get_template('active.html').render({'username': '小花猫'})
    msg = EmailMultiAlternatives(subject, from_email=from_email, to=
[to])
    msg.attach_alternative(html_content, "text/html")
    msg.send()
    return HttpResponse('发送html的文件内容')
```

三、富文本编辑器

一般用于写文章 编辑内容自带样式

- 安装：pip install django-tinymce
- 配置

(1) 配置settings文件

在INSTALL_APPS 添加如下代码

```
INSTALLED_APPS = [
    ...
    'App',
    'tinymce',
]
```

在settings.py下添加如下代码

```
#富文本编辑器的配置
TINYMCE_DEFAULT_CONFIG = {
    'theme':'advanced',
    'width':600,
    'height':400
}
```

(2) 添加视图函数

```
def index(req):
    if req.method == 'GET':
        return render(req,'index.html')

    if req.method == 'POST':
        # print(req.POST)

Posts(title=req.POST.get('title'),content=req.POST.get('content')).sa
ve()
    return HttpResponse('index')
```

(3) 前台模板的展示

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script src="/static/tiny_mce/tiny_mce.js"></script>
    <script>
        tinyMCE.init({
            'mode':'textareas',
            'width':800,
            'height':600,
        })
    </script>
</head>
<body>
<form action="/" method="POST">
    {% csrf_token %}
    <p>标题 <input type="text" name="title" placeholder="请输入标题"
maxlength="20" required></p>
    <textarea name="content" id="" cols="30" rows="10"></textarea>
    <input type="submit">
</form>
</body>
</html>
```


四、文件上传

使用request.FILES 获取上传文件

1.表单注意

- 表单的enctype的值需要设置为：enctype="multipart/form-data
- 表单提交类型为POST

2.存储路径

在settings.py文件下添加如下代码

```
#设置上传文件路径
MDEIA_ROOT = os.path.join(BASE_DIR,'static/upload')
```

3. 文件上传对象的属性和方法

名称	说明
file.name	获取上传的名称
file.size	获取上传文件的大小（字节）
file.read()	读取全部（适用于小文件）
file.chunks()	按块来返回文件 通过for循环进行迭代，可以将大文件按照块来写入到服务器
file.multiple_chunks()	判断文件 是否大于2.5M 返回True或者False

4.创建上传文件的表单

- 模板

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<form action="/doUpload/" method="post" enctype="multipart/form-
data">
    {% csrf_token %}
    <p>文件 <input type="file" name="file"></p>
```

```
    <p><input type="submit" value="上传"></p>
</form>
</body>
</html>
```

- **views.py**

```
from django.conf import settings
import os
#文件上传处理
def doUpload(req):
    file = req.FILES.get('file')
    # print(file.name)
    # print(file.size)
    savePath = os.path.join(settings.MDEIA_ROOT,file.name)
    # print(savePath)
    with open(savePath,'wb') as f:
        # f.write(file.read())
        if file.multiple_chunks():
            for myf in file.chunks():
                f.write(myf)
            print('大于2.5')
        else:
            print('小于2.5')
            f.write(file.read())
    return HttpResponse('文件上传')
```

5.封装文件上传类

可以自定义一个类实现文件上传，文件上传类可以：

- 检查文件类型
- 检查文件大小
- 是否生成随机文件名

```
import os
from datetime import datetime
from random import randint


class FileUpload:
```

```
    def __init__(self,file,exts=
['png','jpg','jpeg'],size=1024*1024,is_randomname=False):
        """
        :param file: 文件上传对象
        :param exts: 文件类型
        :param size: 文件大小，默认1M
        :param is_randomname: 是否是随机文件名，默认是否
        """
        self.file = file
        self.exts = exts
        self.size = size
        self.is_randomname = is_randomname

    #文件上传
    def upload(self,dest):
        """

        :param dest: 文件上传的目标目录
        :return:
        """
        #1 判断文件类型是否匹配
        if not self.check_type():
            return -1
        #2 判断文件大小是否符合要求
        if not self.check_size():
            return -2
        #3 如果是随机文件名，要生成随机文件名
        if self.is_randomname:
            self.file_name = self.random_filename()
        else:
            self.file_name = self.file.name
        #4 拼接目标文件路径
        path = os.path.join(dest,self.file_name)
        #5 保存文件
        self.write_file(path)
        return 1

    def check_type(self):
        ext = os.path.splitext(self.file.name)
        if len(ext) > 1:
            ext = ext[1].lstrip('.')
            if ext in self.exts:
```

```
                return True
        return False

    def check_size(self):
        if self.size < 0:
            return False
        #如果文件大小于给定大小，返回True，否则返回False
        return self.file.size <= self.size

    def random_filename(self):
        filename =
datetime.now().strftime("%Y%m%d%H%M%S")+str(randint(1,10000))
        ext = os.path.splitext(self.file.name)
        #获取文件后缀
        ext = ext[1] if len(ext)>1 else ''
        filename += ext
        return  filename

    def write_file(self,path):
        with open(path,'wb') as fp:
            if self.file.multiple_chunks():
                for chunk in self.file.chunks():
                    fp.write(chunk)
            else:
                fp.write(self.file.read())
```

五、站点管理

(1) 配置admin应用

```
django.contrib.admin
```

(2) 创建管理员用户

python3 manage.py createsuperuser

依次输入用户名->邮箱->密码->确认密码

(3) 汉化

```
LANGUAGE_CODE = 'zh-Hans'
TIME_ZONE = 'Asia/Shanghai'
```

(4) 在App/admin.py 里面注册自己的模型类

```
from .models import Grade,Students

#注册模型类 在后台展示
admin.site.register(Grade)
admin.site.register(Students)
```

(5) 配置后台页面和添加数据的展示

```
#配置数据的展示
class GradeAdmin(admin.ModelAdmin):
    #设置显示哪些字段
    list_display = ['pk','gname','gboynum','ggirlnum']
    #添加搜索字段
    search_fields = ['gname']
    # 分页
    list_per_page = 5
    # 过滤字段‘
    list_filter = ['gname']

class StudentsAdmin(admin.ModelAdmin):
    list_display = ['pk','sname','ssex','sage','grade']
    search_fields = ['sname']
    #分页
    list_per_page = 5
    #过滤字段‘
    list_filter = ['sname']
    #更改添加 修改的字段属性的位置
    # fields = ['sage','ssex','sname','grade','info']
    fieldsets = [
        ("基本信息",{"fields":['sname','sage','ssex']}),
        ("其它信息",{'fields':['info','grade']}),
    ]
    #字段顺序和字段分组不能同时使用


#注册模型类 在后台展示
```

```
admin.site.register(Grade,GradeAdmin)
admin.site.register(Students,StudentsAdmin)
```

(6) 关联对象

```
#TabularInline 横着展示添加学生的布局
#StackedInline 竖着展示添加学生的布局
# class AddStudents(admin.TabularInline):
class AddStudents(admin.StackedInline):
class AddStudents(admin.TabularInline):
    model = Students #关联的模型名称
    extra = 2 #添加学生的个数

#配置数据的展示
class GradeAdmin(admin.ModelAdmin):
    inlines = [AddStudents]
```

(7) bool值的显示男女

```
def sex(self):
        if self.ssex:
            return '男'
        else:
            return '女'

    sex.short_description = '性别'  # 给字段名称添加简介（字段的中文说
明）
    # list_display = ['pk','sname','ssex','sage','grade']
    list_display = ['pk','sname',sex,'sage','grade']
```