验证码、分页、文件上传

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

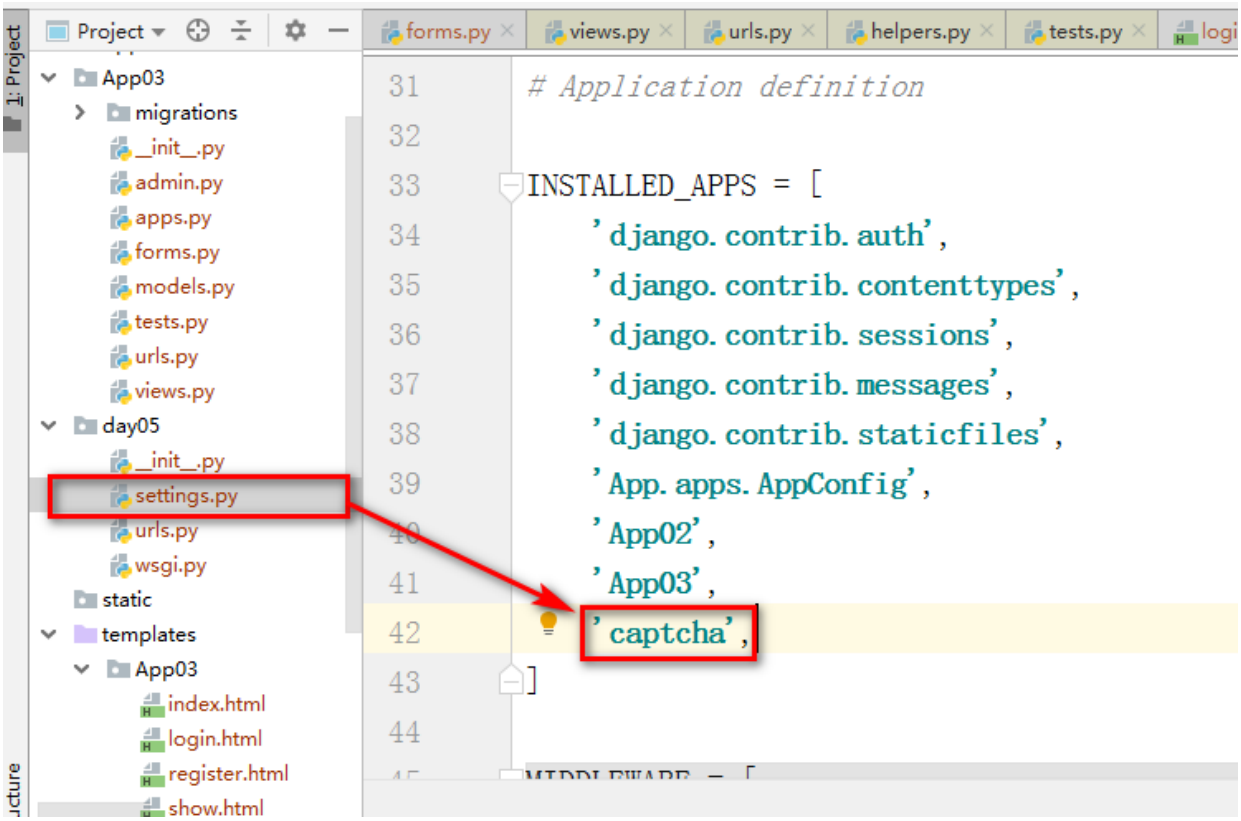

图1. 安装应用

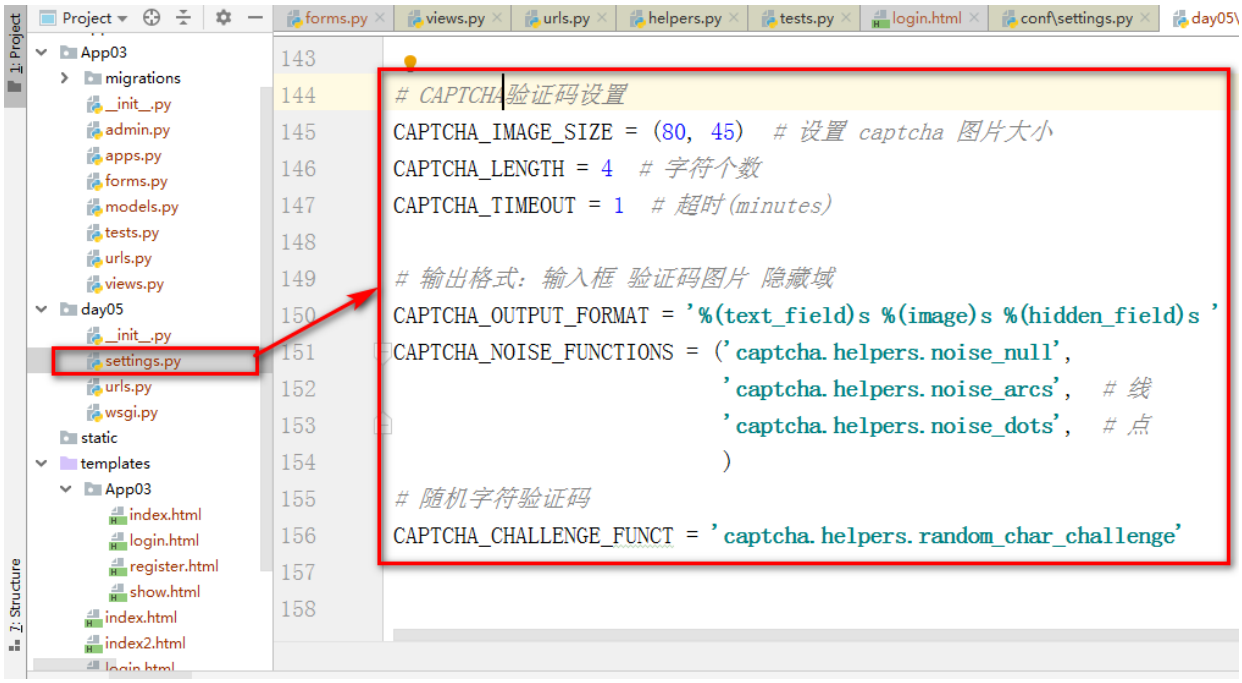

图2. 设置验证码样式

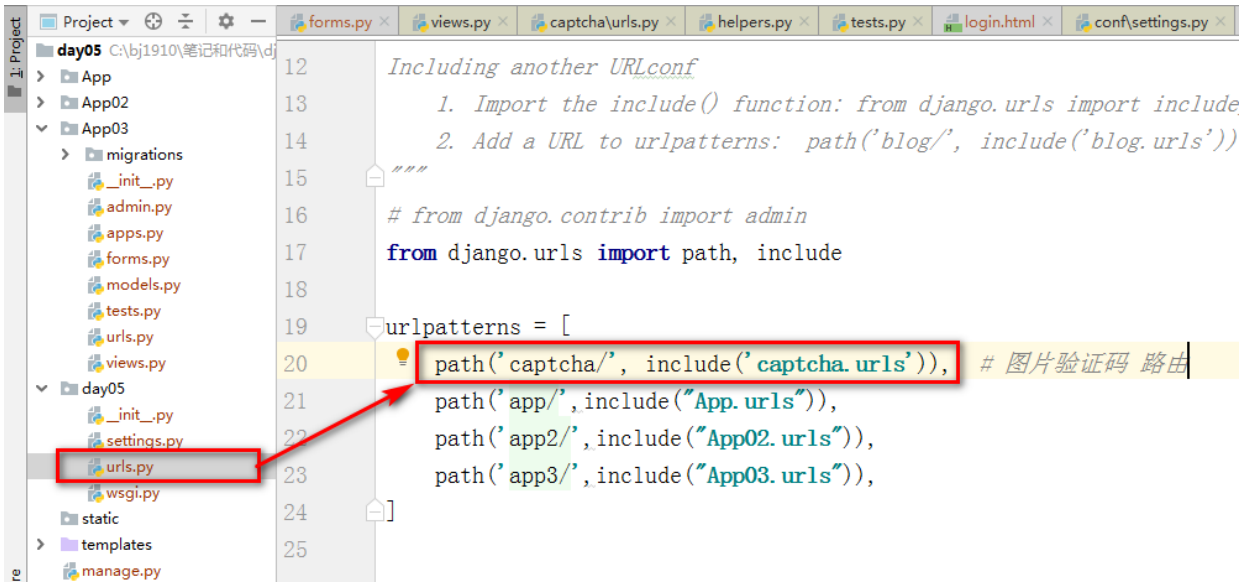

图3. 设置路由

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、短信验证码

很多业务需要使用手机验证码服务，这里以阿里云短信服务为例，说明python如何集成第三方短信服务，之所以选择阿里云短信服务，是因为阿里云短信服务针对个人，只要账号有钱就可以发送短信验证码，而很多短信平台是针对公司的，不支持个人。

1.短信验证码设置

首先登陆阿里云，到控制台，在"产品和服务"中选择：

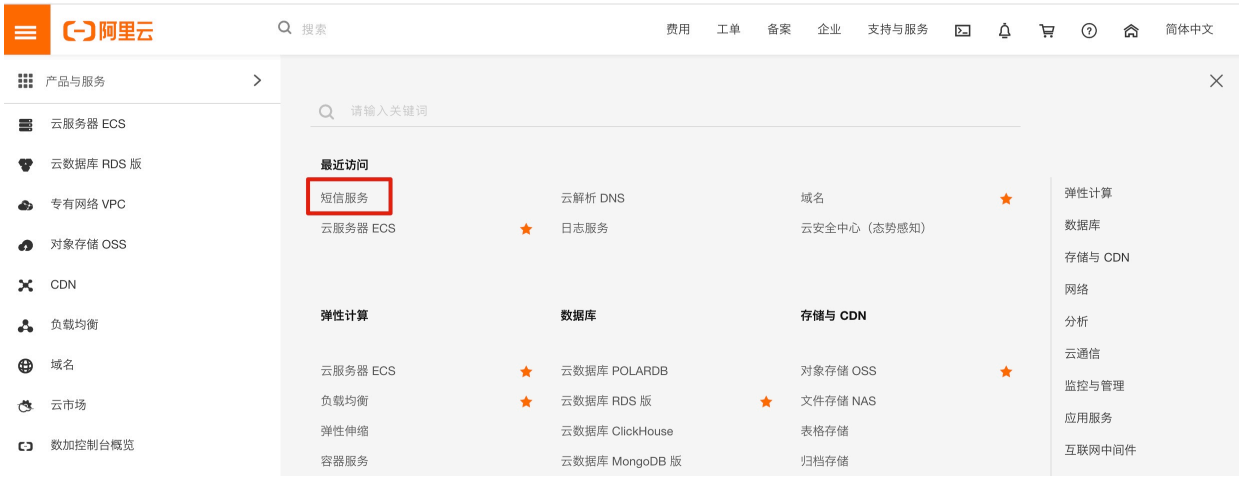

图2.1 访问阿里云短信服务

- 获取key和sceret

图2.2 短信服务的key

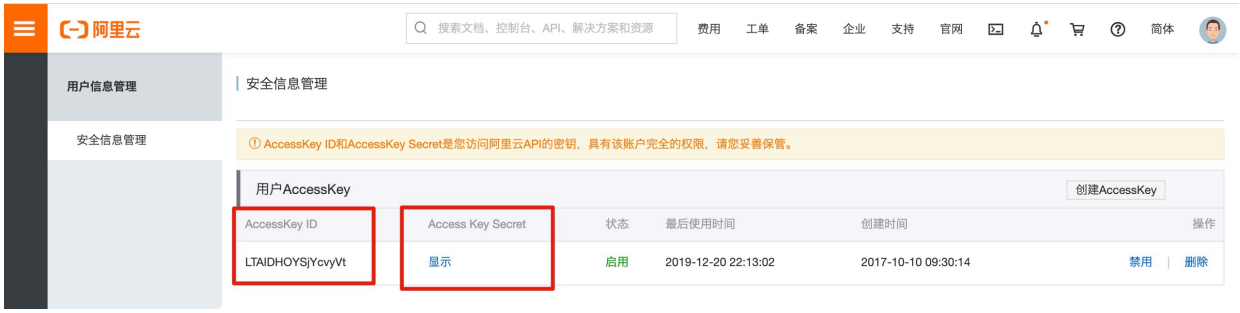

图2.3 获取短信服务的key

- 获取签名

图2.4 获取短信服务的签名

- 获取模板代码

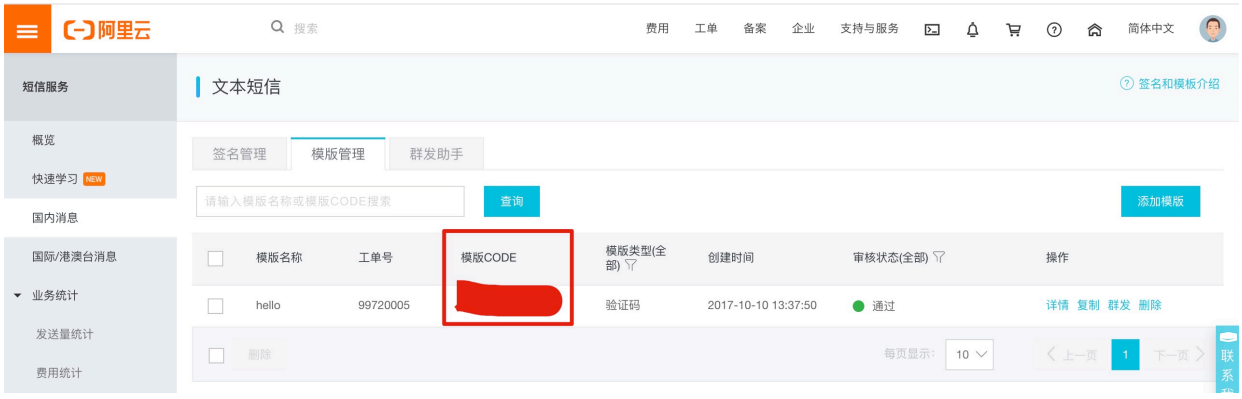

图2.5 获取短信服务的模板代码

2.安装阿里云短信验证码库

安装阿里云SDK核心库，在虚拟开发环境里，执行：

```
pip install aliyun-python-sdk-core
```

在安装完成后，您可以使用OpenAPI Explorer来生成相关API的Demo并应用在您的项目中。

- 短信验证码类

```
import random
from aliyunsdkcore.client import AcsClient
from aliyunsdkcore.request import CommonRequest

ACCESS_KEY_ID = "xxx"  #用户AccessKey  需要根据自己的账户修改
ACCESS_KEY_SECRET = "xxxxxx"  #Access Key Secret  需要根据自己的账
户修改

class SMS:
    def __init__(self,signName,templateCode):
        self.signName = signName  #签名
        self.templateCode = templateCode  #模板code
        self.client = client = AcsClient(ACCESS_KEY_ID,
ACCESS_KEY_SECRET, 'cn-hangzhou')

    def send(self,phone_numbers,template_param):
        request = CommonRequest()
        request.set_accept_format('json')
        request.set_domain('dysmsapi.aliyuncs.com')
        request.set_method('POST')
        request.set_protocol_type('https')  # https | http
        request.set_version('2017-05-25')
        request.set_action_name('SendSms')

        request.add_query_param('RegionId', "cn-hangzhou")
        request.add_query_param('PhoneNumbers', phone_numbers)
        request.add_query_param('SignName', self.signName)
        request.add_query_param('TemplateCode',
self.templateCode)
        request.add_query_param('TemplateParam', template_param)
        response = self.client.do_action_with_exception(request)
        return response


if __name__ == "__main__":
    # 使用自己的签名（xxx）和模板代码(xxxxxx)
    sms = SMS("xxx","xxxxxx")
    phone = input("请输入手机号：")
    #验证码
    num = random.randint(10000,99999)
    # 设置模板参数
    para = "{'number':'%d'}"%num
```

```
    # 发送验证码
    res = sms.send(phone,para)
    print(res.decode('utf-8'))
```

3.实现

在应用的urls中设置

```
from django.urls import path

from App01 import views
app_name = "App01"
urlpatterns = [
    path('sms/',views.send,name='sms'),
]
```

视图函数send的实现

```
@csrf_exempt
def send(request):
    if request.method == 'POST':
        if request.is_ajax():
            phone = request.POST.get('phone')
            sms = SMS("成少雷", "SMS_102315005")
            code = randint(1000, 9999)
            para = "{'number':'%d'}" % code
            request.session['code'] = code
            res = sms.send(phone,para)
            return JsonResponse({'ok':1})
    return render(request,"app01/message.html")
```

前端页面实现

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>发送短信</title>
```

```
</head>
<body>
<form action="{% url 'App01:sms' %}" method="post">
    {% csrf_token %}
    <input type="hidden" name="hello" value="123">
    电话号码：  <input type="text" name="phone" id="phone">
    <input type="button" id="yzm" value="发送验证码">
    <input type="submit">
</form>
</body>
</html>
<script src="{% static 'jquery.min.js' %}"></script>
<script>
    var second = 30;

    $("#yzm").click(function () {
        $(this).prop('disabled',true);
        _this = this;
        let oTimer = setInterval(function () {
            $(_this).prop('value',''+second+"后发送");
            second--;
            if (second <= 0){
                clearTimeout(oTimer);
                oTimer = null;
                $(_this).prop('value',"发送验证码")
            }
        },1000);
        let sPhone = $('#phone').val();
        console.log(sPhone)
        $.post("{% url 'App01:sms' %}","phone="+sPhone,function
(data) {
            console.log(data)
        })
    })
</script>
```

三、分页

1 Paginator 分页器

Paginator用于分页，但Paginator并不具体管理具体的页的处理，而是使用Page对象管理具体页面

- 创建分页器对象

格式： Paginator(<query_set查询集>,每页显示数据的条数)

- 对象的属性

  count 分页对象的个数

  num_pages 总页数

  page_range 页码的列表

- 方法

  page(num) 返回page对象 如果给定的页码不存在 则抛出异常

2 page 对象

page对象具体负责每页的处理，包括每页的数据，当前页的页码，是否有上一页或下一页等。

类别	名称	说明
属性	object_list	当前页码上的所有数据
属性	number	当前页码值
属性	paginator	返回Paginator的对象
方法	has_next	是否有下一页
方法	has_previous	是否有上一页
方法	has_other_pages	是否有上一页 或者下一页
方法	next_page_number	返回下一页的页码
方法	previous_page_number	返回上一页的页码
方法	len	返回当前页数据的个数

- 实例

```
#路由
url(r'^userlist/$',views.userlist,name='userlist'),
url(r'^userlist/(\d+)/$',views.userlist,name='userlist1'),
```

```
# views.py
def userlist(request,page=1):
    users = User.objects.all()
    # 实例化分页对象，一页两条记录
    pagination = Paginator(users,2)
    page = pagination.page(page)  #某一页的分页对象

    return render(request,'userlist.html',context={
        'data':page.object_list,  #当前页的数据(列表)
        'page_range':pagination.page_range,#页码范围
        'page':page
    })
```

```
#模板文件
{% load static %}
<!DOCTYPE html>
<html lang="zh-CN">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <!-- 上述3个meta标签*必须*放在最前面，任何其他内容都*必须*跟随其后！ -
->
    <title>用户列表</title>
    <!-- Bootstrap -->
    <link href="{% static 'bootstrap/css/bootstrap.min.css' %}"
rel="stylesheet">
  </head>
  <body>
    <div class="container-fluid">
        <div class="row">
            <div class="bs-example" data-example-id="bordered-table">
                <table class="table table-bordered">
                    <thead>
                    <tr>
                        <th>#</th>
```

```
                        <th>用户名</th>
                        <th>密码</th>
                        <th>年龄</th>
                    </tr>
                    </thead>
                    <tbody>
                    {% for user in data %}
                        <tr>
                            <th scope="row">{{ forloop.counter }}
</th>
                            <td>{{ user.username }}</td>
                            <td>{{ user.password }}</td>
                            <td>{{ user.age }}</td>
                        </tr>
                    {% endfor %}



                    </tbody>
                </table>
            </div>
        </div>
    </div>
    <div class="container-fluid">
        <div class="row">
            <div class="col-md-4"></div>
            <div class="col-md-5">
                <nav aria-label="Page navigation">
                    <ul class="pagination">
                        {% if page.has_previous %}
                           <li>
                                <a href="{% url 'app:userlist1'
page.number|add:'-1' %}" aria-label="Previous">
                                    <span aria-hidden="true">&laquo;
</span>
                                </a>
                            </li>
                        {% else %}
                             <li class="disabled">
                            <span href="{% url 'app:userlist1'
page.number|add:'-1' %}" aria-label="Previous">
```

```
                                <span aria-hidden="true">&laquo;
</span>
                            </span>
                            </li>

                        {% endif %}
                        {% for num in pagerange %}
                            {% ifequal num page.number %}
                                <li class="active"><a href="{% url
'app:userlist1' num %}">{{ num }}</a></li>
                            {% else %}
                                <li><a href="{% url 'app:userlist1'
num %}">{{ num }}</a></li>
                            {% endifequal %}

                        {% endfor %}
                        {% if page.has_next %}
                            <li>
                                <a href="{% url 'app:userlist1'
page.number|add:'1' %}" aria-label="Next">
                                    <span aria-hidden="true">&raquo;
</span>
                                </a>
                            </li>
                        {% else %}
                             <li class="disabled">
                                <span href="#" aria-label="Next">
                                    <span aria-hidden="true">&raquo;
</span>
                                </span>
                            </li>
                        {% endif %}
                    </ul>
                </nav>

            </div>
            <div class="col-md-3"></div>
        </div>
    </div>
    <!-- jQuery (Bootstrap 的所有 JavaScript 插件都依赖 jQuery，所以必
须放在前边) -->
```

```
    <script src="{% static 'bootstrap/js/jquery-1.12.4.min.js' %}">
</script>
    <!-- 加载 Bootstrap 的所有 JavaScript 插件。你也可以根据需要只加载单
个插件。 -->
    <script src="{% static 'bootstrap/js/bootstrap.min.js' %}">
</script>
  </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())
```