文件上传和富文本编辑

django文件上传下载

上传

配置settings.py

```
# 设定文件的访问路径，如：访问http://127.0.0.1:8000/media/就可以获取文件
MEDIA_URL = '/media/'
# 设置文件的存储路径，全部存储在media目录下，会和model类中的upload_to属性值拼接
MEDIA_ROOT = os.path.join(BASE_DIR,'media')
```

models.py

```
class Img(models.Model):
    name = models.CharField(max_length=32)
    # upload_to拼接在MEDIA_ROOT后面，../media/img/article/，内容均存在该目录下
    img = models.ImageField(upload_to='img/article/',verbose_name='图片')
```

upload.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<!--enctype属性值修改成"multipart/form-data"-->
<form action="" method="post" enctype="multipart/form-data">
    {% csrf_token %}
    <!--type类型要改成file，文件类型-->
    <input type="text" name="name">
    <input type="file" name="img">
    <button>上传</button>
</form>
</body>
</html>

<!--多文件上传-->
<input type="file" name="myfiles" multiple="">
```

views.py

```
# 获取上传文件单个插入数据
def upload(request):

    if request.method == 'POST':
        # 获取文件名称
        img_url = request.FILES.get("img")
```

```
        name = request.POST.get("name")
        # 存到数据库中，并保存到指定的目录下
        img = models.Media(name=name,img=img_url)
        img.save()

    return render(request,"youhua.html")

# 获取上传文件插入批量数据
def upload(request):
    if request.method == 'POST':
        img_list = request.FILES.getlist('img')
        name = request.POST.get("name")
        querysetlist = []
        for img in img_list:
            querysetlist.append(models.Media(name=name,img=img))
        models.Media.objects.bulk_create(querysetlist)

        return HttpResponse("上传成功")

    return render(request, "youhua.html")
```

ajax上传图片

FormData上传

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script src="https://cdn.staticfile.org/jquery/1.10.2/jquery.min.js">
</script>
</head>
<body>
<input type="file" name="img" id="img">
<input type="button" value="上传" onclick="showImg();">
<script type="text/javascript">
    function showImg() {
        var formdata = new FormData();// 创建一个空的FormData对象，可以使用
formdata.append(key,value)来添加数据。
        formdata.append('file', document.getElementById('img').files[0]);
        $.ajax({
            url: '/upload/',
            type: 'post',
            headers :{
                'x-csrftoken': '{{ csrf_token }}', // 为了通过csrf校验，所以必须带这
个过去。
            },
            data: formdata,
            // 默认值为true，默认会将发送的数据序列化以适应默认的内容类型。不想转换信息，需要
设置为false
            // 此数据传输的是对象，所以不需要序列化
            processData: false,
            // 不写默认为application/x-www-form-urlencoded只能上传文本，上传文件需要用
multipart/form-data类型。
            contentType: false, // 不设置内容类型
            success: function (data) {
```

```
                alert(data)
            }
        });
    }
</script>
</body>
</html>
```

views.py

```
def upload(request):
    if request.method == 'POST':
        img_url = request.FILES.get('file')
        img = models.Media(name='hello',img=img_url)
        img.save()
        ret = '上传成功'
        return HttpResponse(ret)

    return render(request,"youhua.html")
```

页面展示图片

如果涉及到下载/显示资源，就需要添加url

```
from django.contrib import admin
from django.urls import path
from imgTest.views import uploadImg, showImg
from django.conf.urls.static import static
from django.conf import settings

# 写法一
urlpatterns = [
    path('admin/', admin.site.urls),
    path('upload/', upload),
    path('showImg/', showImg)
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

# 写法二
from django.views.static import serve
urlpatterns = [
    ...
    path('showImg/', showImg),
    url('^media/(?P<path>.*)', serve, {'document_root': settings.MEDIA_ROOT})
]
```

views.py

```
def showImg(request):
    obj_list = models.Img.objects.all()
    print(obj_list)

    return render(request,'showImg.html',{"obj_list":obj_list})
```

showImg.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
{% for obj in obj_list %}
    <!--必须要带url，不然路径会错误-->
    <img src="{{ obj.img.url }}" alt="">
{% endfor %}
</body>
</html>
```

文件下载

自定义编写视图下载方法，主要是为了限制用户的下载内容，**一定要注意限制用户的下载内容**，不然知道路径连代码和数据库都下载了。
urls.py

```
from app_youhua import views

urlpatterns = [
    # ......
    url('^download/(?P<pk>\d+)/$', views.download, name='download'),
]
```

html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    {% load static %}
</head>
<body>
{% for obj in obj_list %}
    <img src="{{ obj.img.url }}" alt="">  <!--自动将数据库中的图片全部显示，可以自定制拼接路径-->
    <a href="{% url 'download' obj.pk %}">下载</a>  <!--url反向解析-->
{% endfor %}

</body>
</html>
```

使用HttpResponse

```
from django.http import HttpResponse, Http404

def download(request, pk=None):
    obj = models.Media.objects.get(pk=pk)
    filename = str(obj.img)
```

```
    filepath = os.path.join(settings.MEDIA_ROOT, filename)
    with open(filepath, 'rb') as f:
        try:
            response = HttpResponse(f)
            response['content_type'] = "application/octet-stream"
            response['Content-Disposition'] = 'attachment; filename=%s' %
filename
            return response
        except Exception:
            raise Http404
```

HttpResponse有个很大的弊端，其工作原理是先读取文件，载入内存，然后再输出。如果下载文件很大，该方法会占用很多内存。对于下载大文件，Django更推荐StreamingHttpResponse和FileResponse方法，这两个方法将下载文件分批(Chunks)写入用户本地磁盘，先不将它们载入服务器内存。

使用StreamingHttpResponse和FileResponse

```
from django.http import FileResponse, StreamingHttpResponse

def download(request, pk=None):
    obj = models.Media.objects.get(pk=pk)
    filename = str(obj.img)
    # filename = request.GET.get('file')  # 如果文件名直接通过页面传回
    filepath = os.path.join(settings.MEDIA_ROOT, filename)
    # 文件将会自动关闭，所以不需要使用with语句打开文件
    fp = open(filepath, 'rb')
    response = StreamingHttpResponse(fp)
    # response = FileResponse(fp) # 默认一次下载4096字节
    response['Content-Type'] = 'application/octet-stream'
    response['Content-Disposition'] = 'attachment;filename="%s"' % filename
    return response
```

文件私有化的两种方法

如果你想实现只有登录过的用户才能查看和下载某些文件。

- 上传文件放在media文件夹，文件名使用很长的随机字符串命名(uuid), 让用户无法根据文件名猜出这是什么文件。视图和模板里验证用户是否已登录，登录或通过权限验证后才显示具体的url。- 简单易实现，安全性不高，但对于一般项目已足够。
- 上传文件放在非media文件夹，用户即使知道了具体文件地址也无法访问，因为Django只会给media文件夹里每个文件创建独立url资源。视图和模板里验证用户是否已登录，登录或通过权限验证后通过自己编写的下载方法下载文件。- 安全性高，但实现相对复杂。
- 我们定义的下载方法可以下载所有文件，不仅包括.py文件，还包括不在media文件夹里的文件(比如非用户上传的文件)。比如当我们直接访问127.0.0.1:8000/file/download/file_project/settings.py/时，你会发现我们连file_project目录下的settings.py都下载了。

```
# 简单举例，不让用户下载.py等结尾的文件
from django.http import Http404,FileResponse, StreamingHttpResponse
def download(request, pk=None):
    obj = models.Media.objects.get(pk=pk)
    filename = str(obj.img)
    filepath = os.path.join(settings.MEDIA_ROOT, filename)
```

```
    ext = os.path.basename(file_path).split('.')[-1].lower()
    # cannot be used to download py, db and sqlite3 files.
    if ext not in ['py', 'db',  'sqlite3']:
        # 文件将会自动关闭，所以不需要使用with语句打开文件
        fp = open(filepath, 'rb')
        response = StreamingHttpResponse(fp)
        response['content_type'] = "application/octet-stream"
        response['Content-Disposition'] = 'attachment;filename="%s"' % filename
        return response
    else:
        raise Http404
```

回到顶部

django富文本编辑框

下载

```
pip install django-ckeditor
```

注册

```
INSTALLED_APPS = [
    ...
    'ckeditor',
    'ckeditor_uploader',
]
```

settings中配置

```
CKEDITOR_UPLOAD_PATH = 'ckeditor/'
```

配置urls.py

```
from ckeditor_uploader import views
from django.views.static import serve

urlpatterns = [
    url(r'^media/(?P<path>.*)', serve, {'document_root':settings.MEDIA_ROOT}),
    # 上传文件
    url(r'^ckeditor/upload/', views.upload),
    url(r'^ckeditor/', include('ckeditor_uploader.urls')),
]
```

models.py使用富文本编辑框字段

```
from ckeditor_uploader.fields import RichTextUploadingField
class Article(models.Model):
    title = models.CharField(max_length=32)
    detail = models.OneToOneField('ArticleDetail',on_delete=models.CASCADE)

class ArticleDetail(models.Model):
    content = RichTextUploadingField(verbose_name='文章详情')
```

模板中使用

```
{{ field }} 使用ModelForm富文本编辑框的字段，
# 导入js样式，只要需要显示富文本编辑框就要导入
<script src="{% static 'ckeditor/ckeditor/ckeditor.js' %}"></script>
<script src="{% static 'ckeditor/ckeditor-init.js' %}"></script>
```

关联表同时显示富文本编辑框

比如：编辑内容时，肯定是连同关联的详情内容一同填写。

```
# 视图函数
def article_add(request):
    # 对两个form表单进行实例化
    form_obj = ArticleForm()
    detail_form = ArticleDetailForm()
    if request.method == 'POST':
        # 先校验并保存被关联方
        detail_form = ArticleDetailForm(request.POST)
    if detail_form.is_valid():
        detail_form.save()
        qd = request.POST.copy() # request.POST是有序字典，默认是不可编辑，所以进行深拷贝后编辑
        # 找到被关联方提交此条数据的pk
        qd['detail'] = detail_form.instance.pk
        # 校验并保存被关联方
        form_obj = ArticleForm(data=qd, files=request.FILES) # 如存在文件类，需单独传参
    if form_obj.is_valid():
        form_obj.save()
        return redirect(reverse('backend:article_list'))
    # 如果被关联方未通过校验，且关联方通过校验并保存，则删除关联方保存的数据
    if detail_form.is_valid() and detail_form.instance:
        detail_form.instance.delete()
        title = '新增文章'
    return

 render(request,'backend/article_form.html'{'form_obj':form_obj,'title':title,'detail_form':detail_form})
from django import forms
from repository import models

class ArticleForm(forms.ModelForm):
    class Meta:
        model = models.Article
        fields = '__all__'
```

```
    def __init__(self,*args,**kwargs):
        super(ArticleForm, self).__init__(*args,**kwargs)

        for field in self.fields.values():
            # 用来控制不使用'form-control'样式的
            if isinstance(field.widget,forms.ClearableFileInput):
                continue
            field.widget.attrs['class'] = 'form-control'

class ArticleDetailForm(forms.ModelForm):
    class Meta:
        model = models.ArticleDetail
        fields = '__all__'
```

给Django后台富文本编辑器添加上传文件的功能

使用富文本编辑器上传的文件是要放到服务器上的，所以这是一个request。既然是一个request，就需要urls.py进行转发请求views.py进行处理。views.py处理完了返回一个文件所在的路径给富文本编辑器，富文本编辑器通过HTML来渲染文件，如果文件是图片，就显示图片。

以下以simditor富文本编辑器为例。它上传文件的api是这样的：

```
#upload要么为false 要么为对象
upload:{
        url: '',
        params: null,
        fileKey: 'upload_file',
        connectionCount: 3,
        leaveConfirm: 'Uploading is in progress, are you sure to leave this
page?'
    }
```

需要返回的JSON格式：

```
{
  "success": true/false,
  "msg": "error message", # optional
  "file_path": "[real file path]"
}
```

第1步：在settings.py建立MEDIA的全局变量

```
#settings.py
MEDIA_URL='/uploads/'
MEDIA_ROOT=os.path.join(BASE_DIR,'uploads')
```

第2步：配置富文本编辑器JS文件

```
upload:{
        url:'/myadmin/upload/files', /* 注意myadmin前面的斜杠不能省掉，这是相对于根目录的*/
        filekey:'upload_file', /* 相当于html标签里面的name值 */
    }
```

第3步：配置urls.py

```
#urls.py
from blog.upload_proc import upload_file

urlpatterns+=[
   url(r'^myadmin/upload/(?P<dir_name>)',upload_file)
]
```

第4步：撰写upload_file处理函数

```
#upload_proc.py

from django.http import HttpResponse
from django.views.decorators.csrf import csrf_exempt
from django.conf import settings
import json
import os
import datetime

@csrf_exempt  #取消csrf验证，否则会有403错误
def file_upload(request,dir_name):
    files=request.FILES.get('upload_file') #得到文件对象
    today=datetime.datetime.today()

    file_dir=settings.MEDIA_ROOT+dir_name+'/%d/%d/%d/'%
(today.year,today.month,today.day)
    if not os.path.exists(file_dir):
        os.makedirs(file_dir)
    file_path=file_dir+files.name

    open(file_path,'wb+').write(files.read()) #上传文件

    #得到JSON格式的返回值
    upload_info={"success":True,'file_path':settings.MEDIA_URL+files.name}
    upload_info=json.dumps(upload_info)

    return HttpResponse(upload_info,content_type="application/json")
```

第5步：再次配置urls.py

为什么需要再次配置 `urls.py` 呢？因为文本编辑器虽然返回了已经上传的文件的地址，但是要显示在页面上，实际上又是一次request。凡是 `request`，都需要 `urls.py` 转发给 `views.py` 进行处理。

```
#urls.py
from django.conf import settings

#django.views.static.serve是内置的，但是只能在开发中使用，生产环境中需要交给服务器来处理，因为上传的文件已经属于静态文件了。
urlpatterns+=[
   url(r'^uploads/(?P<path>.*)$',diango.views.static.serve,
{'document_root':settings.MEDIA_ROOT})
]
```

扩展：一次性上传多个文件，如上传多张图片

1. simditor采用的方法是利用ajax请求多次，有几个文件就请求几次。
2. 有没有其他的方法呢?

神级程序员在使用Django时写下的富文本编辑器及上传的详细介绍!

什么是富文本编辑器

百度百科有详细介绍

KindEditor是一套开源的在线HTML编辑器，主要用于让用户在网站上获得所见即所得编辑效果，开发人员可以用 KindEditor 把传统的多行文本输入框(textarea)替换为可视化的富文本输入框。 KindEditor 使用 JavaScript 编写，可以无缝地与 Java、.NET、PHP、ASP 等程序集成，比较适合在 CMS、商城、论坛、博客、Wiki、电子邮件等互联网应用上使用。

Django配置

1 、配置static静态资源

KindEditor是用JavaScript编写的，这属于static files，因此需要为Django设置static路径。 首先在工程目录下新建static文件夹，这里要注意的是千万不要在my_app/下创建static文件夹作为static文件存放的目录, 这会导致Django无法搜索到自己的static 文件。创建后好，在settings中配置static文件目录。添加以下代码

```
STATIC_URL = '/static/'

STATICFILES_DIRS = (
    os.path.join(BASE_DIR,  'static'),
)

MEDIA_URL = '/uploads/'

MEDIA_ROOT = os.path.join(BASE_DIR,  'uploads')
```

2 、下载

> http://kindeditor.net/down.php

3 、解压并复制到项目目录下

4 、定义Media类并编辑kindeditor配置

```
class ArticleAdmin(admin.ModelAdmin):
    list_display = ('title', 'desc','click_count','date_publish')#显示列
    list_display_links = ('title', 'desc',)#显示列上的链接效果
    list_editable = ('click_count',)#可编辑的列

    fieldsets = (
        (None, {
            'fields': ('title', 'desc', 'content', 'user', 'category', 'tag',)
        }),
        ('高级设置', {
            'classes': ('collapse',),#折叠状态
            'fields': ('click_count', 'is_recommend',)
        }),
    )

    class Media:
        js = (
            '/static/js/kindeditor-4.1.10/kindeditor-min.js',
            '/static/js/kindeditor-4.1.10/lang/zh_CN.js',
            '/static/js/kindeditor-4.1.10/config.js',
        )
```

config.js

```
KindEditor.ready(function (K) {
    K.create('textarea[name=content]', {/*css元素*/
        width: '800px',
        height: '200px',
        uploadJson: '/admin/upload/kindeditor',/*请求url*/
    });
});
```

5 、配置url

```
urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^admin/upload/(?P<dir_name>[^/]+)$', upload.upload_image, name='upload_image'),
]
```

upload.py

```
# -*- coding: utf-8 -*-
from django.http import HttpResponse
from django.conf import settings
from django.views.decorators.csrf import csrf_exempt
import os
import uuid
import json
import datetime as dt
```

```python
@csrf_exempt
def upload_image(request, dir_name):
    ##################
    #  kindeditor图片上传返回数据格式说明:
    # {"error": 1, "message": "出错信息"}
    # {"error": 0, "url": "图片地址"}
    ##################
    result = {"error": 1, "message": "上传出错"}
    files = request.FILES.get("imgFile", None)
    if files:
        result =image_upload(files, dir_name)
    return HttpResponse(json.dumps(result), content_type="application/json")
```

```python
#目录创建
def upload_generation_dir(dir_name):
    today = dt.datetime.today()
    dir_name = dir_name + '/%d/%d/' %(today.year,today.month)
    if not os.path.exists(settings.MEDIA_ROOT + dir_name):
        os.makedirs(settings.MEDIA_ROOT + dir_name)
    return dir_name
```

```python
# 图片上传
def image_upload(files, dir_name):
    #允许上传文件类型
    allow_suffix =['jpg', 'png', 'jpeg', 'gif', 'bmp']
    file_suffix = files.name.split(".")[-1]
    if file_suffix not in allow_suffix:
        return {"error": 1, "message": "图片格式不正确"}
    relative_path_file = upload_generation_dir(dir_name)
    path=os.path.join(settings.MEDIA_ROOT, relative_path_file)
    if not os.path.exists(path): #如果目录不存在创建目录
        os.makedirs(path)
    file_name=str(uuid.uuid1())+"."+file_suffix
    path_file=os.path.join(path, file_name)
    file_url = settings.MEDIA_URL + relative_path_file + file_name
    open(path_file, 'wb').write(files.file.read()) # 保存图片
    return {"error": 0, "url": file_url}
```

测试

文章内容: