图文混排

> 在某些富文本编辑器中，我们可以支持文件与图片的同时编写，将文字与图片混合排列
>
> 在`django`中可以使用`TimyMce`富文本编辑器进行图文混排功能的实现

TinyMce

> `TinyMCE`：是一个轻量级的基于浏览器的所见即所得编辑器，支持目前流行的各种浏览器，由`JavaScript`写成
>
> 功能配置灵活简单，两行代码就可以将编辑器嵌入网页中，并且支持`AJAX`，加载速度非常快
>
> 最重要的是，`TinyMCE`是一个根据`LGPL license`发布的自由软件，你可以把它用于商业应用。下图是此编辑器的界面

配置上传路径

> 首先配置整个项目的上传文件路径，为了区别与自身的`static`静态目录
>
> 这里的上传文件我们将另外保存至`upload`文件夹

```
#settings.py
UPLOAD_ROOT = os.path.join(BASE_DIR,'upload')
```

配置模板页面

> 首先需要在使用到`tinymce`富文本编辑器的`html`页面下导入必备`js`文件

```
<script src="{% static 'js/jquery-1.10.2.min.js' %}" ></script>
<script src="{% static 'js/tinymce_setup.js' %}"></script>

<script src="{% static 'tinymce/js/tinymce/tinymce.min.js' %}"></script>
```

> 接下来，在模板页面中加入一个`id`为`content`的输入表单，这里我们以一个文章数据为例

- 首先是模型层文件定义

```
#models.py
class Article(models.Model):
    title = models.CharField(max_length=100,verbose_name='标题')
    author = models.CharField(max_length=100,verbose_name='作者')
    content = models.TextField(verbose_name='内容')
```

> 这里的`content`内容要用到富文本编辑器

- 接着是模板页面的主要部分

```
<form method="POST" action='/' enctype="multipart/form-data">
    {% csrf_token %}
    <input type="text" placeholder="文章标题">
    <br>
    <input type="text" placeholder="文章作者">
    <br>
    <input id="rich_content" name="content" value=" ">
    <br>
    <button type="submit">提交</button>
</form>
```

修改插件配置

> 这里搭配了一个`tinymce_setup.js`文件，用来控制富文本编辑器所使用的插件等
>
> 全文配置如下

```
tinymce.init({
    // 选择id为content的标签作为编辑器
    selector: '#rich_content',
    // 方向从左到右
    directionality:'ltr',
    // 语言选择中文
    language:'zh_CN',
    // 高度为400 宽度为一半
    height:300,
    width: '50%',
    // 工具栏上面的补丁按钮
    plugins: [
            'advlist autolink link image lists charmap print preview hr anchor pagebreak
spellchecker',
            'searchreplace wordcount visualblocks visualchars code fullscreen
insertdatetime media nonbreaking',
            'save table contextmenu directionality template paste textcolor',
            'codesample imageupload',
    ],
    // 工具栏的补丁按钮
     toolbar: 'insertfile undo redo | \
     styleselect | \
     bold italic | \
     alignleft aligncenter alignright alignjustify | \
     bullist numlist outdent indent | \
     link image | \
     print preview media fullpage | \
     forecolor backcolor emoticons |\
     codesample fontsizeselect fullscreen |\
     imageupload',
    // 字体大小
    fontsize_formats: '10pt 12pt 14pt 18pt 24pt 36pt',
    // 按tab不换行
    nonbreaking_force_tab: true,
    imageupload_url: "/upload_img/",
```

```
    // 上传后图片保存为绝对路径
    relative_urls : false,
});
```

> 注释已经很清晰
>
> 要注意的是`imageupload_url`配置用来确定当前图片上传所对应的视图路由

上传视图配置

> 接下来编写富文本编辑器的上传图片路由函数及对应的路由配置
>
> 路由映射`tinymce_setup.js`与中的`imageupload_url`配置路由相同
>
> 这里还要注意，由于此时`tinymce`的上传图片表单并不是和本身所容纳的`form`表单一起上传，所以并不会具备`csrf_token`值，需要我们将上传图片的函数额外进行装饰器装饰，取消`csrf_token`验证

```
#views.py
from django.views.decorators.csrf import csrf_exempt
def md5(str_):
    import time
    m = hashlib.md5()
    m.update(str(time.time()).encode())
    filename = m.hexdigest()
    return filename + '.' + str_.split('.')[-1]

@csrf_exempt
def upload_img(request):
    if request.method == 'POST':
        img = request.FILES.get('file')
        if img:
            file_name_md5 = md5(img.name)
            with open(os.path.join(UPLOAD_ROOT,file_name_md5), 'wb') as fp:
                for buf in img.chunks():
                    fp.write(buf)
                # 迭代读取文件并写入到本地
        response = {}
        response['path'] = '/upload/' + file_name_md5
        response['error'] = False
        return HttpResponse(json.dumps(response))
```

> 视图函数还是老样子去接收上传文件并保存即可
>
> 这里还使用了`md5`的方式进行文件名保存，避免重名文件上传互相覆盖

- 路由配置

```
#urls.py
path('upload_img/',views.upload_img),
```

表单接收视图

> 以上的视图函数只能处理上传图片的内容接收

> 我们的模板页面中还有作者及标题两样表单内容会被`POST`提交到后台
>
> 并且要主要的是，富文本编辑器里除了图片的内容，还有文字等其他内容，这里也需要我们保存下来

- 这里还需要一个视图函数去接管处理

```
def index(request):
    if request.method == 'GET':
        return render(request, 'index.html')
    if request.method == 'POST':
        title = request.POST.get('title')
        author = request.POST.get('author')
        content = request.POST.get('content')
        models.Article.objects.create(
            title = title,
            author = author,
            content = content,
        )
        return redirect('/show/')
```

- 表单路由

```
#urls.py
path('',views.index),
```

- 这里上传之后，保存在后台的`content`数据是这个样子

```
<p><img src="/upload/ec8f8c9a56ed32464a6727741fd58d8d.png" /></p><p>今天<em>一切</em>都是
<strong>美丽</strong>的，哈哈哈</p>
```

图片访问路由

> 当有了上传图片的视图函数及所有内容的接收视图函数
>
> 这里还有一个特殊的问题，现在如果在富文本编辑器中选择上传图片，你会发现图片已经在选择时就已经存储到了后台`upload`文件夹下，这也是为什么我们的图片上传视图函数要单独编写，并且还需要取消`csrf_token`的验证的原因

> 除了这个问题你还会发现，在富文本编辑器中，上传的图片是看不到的，是一个坏掉的图片；
>
> 使用`F12`开发者工具你可以看到，这里的图片展示为一个`img`标签，而标签的`src`属性内容正是通过上传视图函数返回的`response`中的`path`值
>
> 想让这个`path`值在访问时，能获取到实际的图片效果，需要我们在路由文件中继续配置，配置专门的`upload/xxxx.jpg`的图片路由访问，让图片真正展示出来

```
#urls.py
from django.views.static import serve
re_path('^upload/(?P<path>.*)/$',serve,{'document_root':UPLOAD_ROOT}),
```

> 上传后图片的访问路径是`/upload/`，那么这里的路由也是`upload/`，之后通过静态文件映射函数`serve`查找`upload`文件夹下的同路径同名图片资源
>
> 当有了这条路由配置之后，再次再富文本编辑器中加入图片，你就会发现图片出现啦

上传并展示

> 最后，我们将测试上传图片及文字
>
> 并且上传成功之后，重定向到`show`视图函数

- 这个视图函数用来提取当前最新的上传数据并返回到模板页面

```
#views.py
def show(request):
    if request.method == 'GET':
        article = models.Article.objects.all().last()
        return render(request,'show.html',locals())
```

- 展示的模板页面

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>展示文章</title>
</head>
<body>
    {% if article %}
        <h3>{{ article.title }}</h3>
        <h4>{{ article.author }}</h4>
        <div>{{ article.content|safe }}</div>
    {% endif %}
</body>
</html>
```

> 其实本身富文本编辑器上传的文本内容就已经在一个`p`标签中，所以这里没有用段落标签
>
> 另外由于保存在数据库的文本为`html`格式，而后台传递来的模板变量`django`出于安全考虑会自动进行转义，直接观看到的效果不会`html`样式，只是一些普通字符串；
>
> 这里可以使用`safe`过滤器将内容认定为安全，展示为原始的`html`效果，其实还不错

总结

> 富文本编辑器其实就是将用户输入的内容变为`html`代码
>
> 这里的图文混排，只是在图片加入时，单独将图片上传保存，并且回调获取到上传的路径
>
> 之后只需要服务端后台配置好相关的上传图片访问路由配置即可