

边干边学 Django

目录

第一节 开发环境.....	3
第二节 项目展示.....	3
第三节 项目实现的主要功能.....	5
第四节 项目的介绍.....	5
第五节 django 入门	5
5.1 初步了解 django.....	5
5.2 开始 django 的第一步使用.....	6
5.2.1 新建一个 django project.....	6
5.2.2 新建一个 django 的 app	6
5.2.3 编写 app	6
5.3 django 进阶（使用模板与模型）.....	10
第六节 开始我们项目的制作.....	15
6.0 创建一个 django 项目.....	15
6.1 编写我们的 models 模型.....	15
6.2 在 admin 注册我们的模型.....	16
6.3 在命令行中创建数据库表.....	16
6.4 创建管理员账号（参考前面教程，现在我们可以管理员界面添加各个类的数据了，只是现在还没有在具体文件中使用）	16
6.5 分析并编写主页 class_home_page.html（更多详情可到 github 查看源代码， https://github.com/ZZKdev/class-site ）	16
6.6 编写主页视图	17
6.7 使用通用视图显示 News 类详细信息.....	20
6.8 Announcement 类的功能实现.....	21
6.9 编写文件下载功能	23
6.10 实现用户端的文件上传.....	26
6.11 生活点滴功能的实现和课程表功能实现：.....	28
第七节 运行我们的项目	30

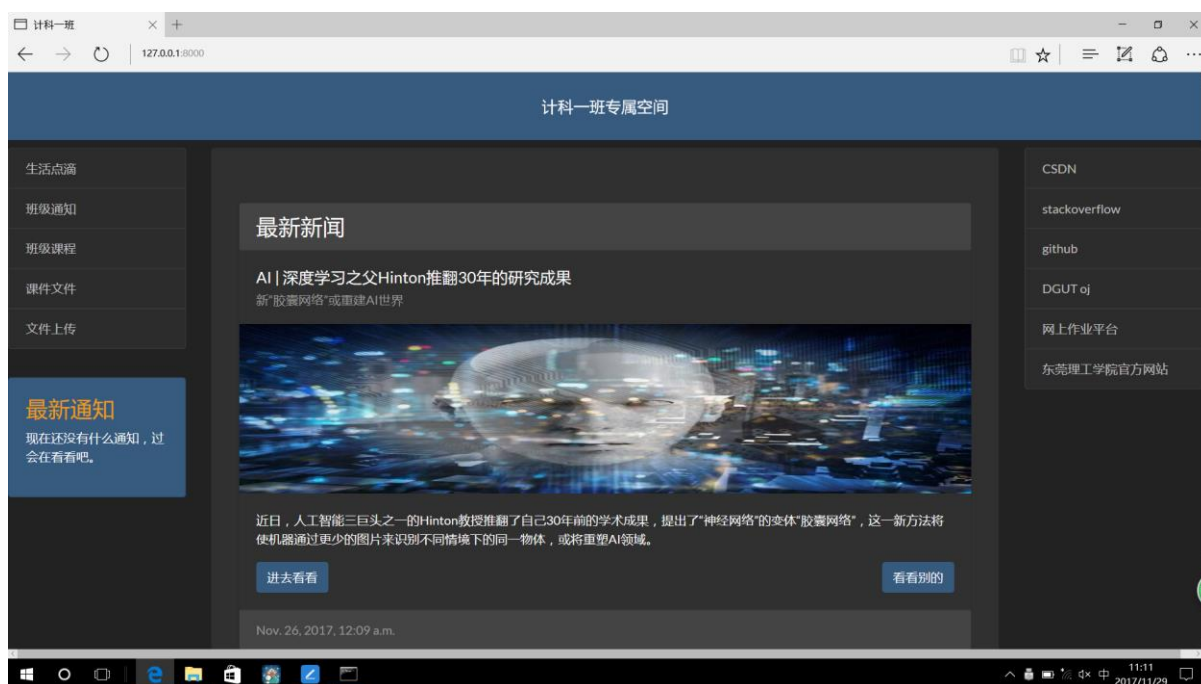
第一节 开发环境

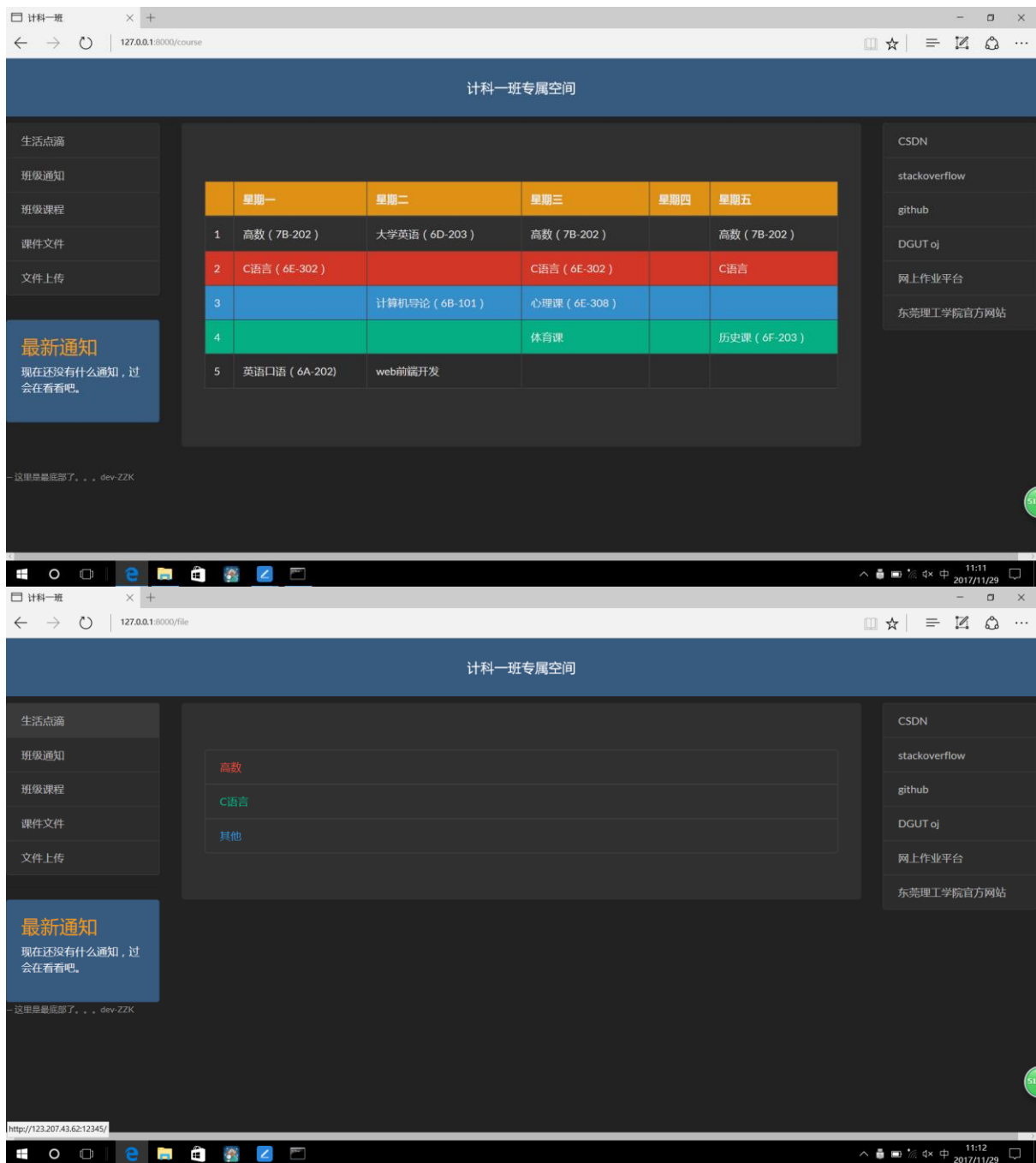
使用语言：python3

使用框架：django

使用前端 CSS 样式：bootstrap

第二节 项目展示







项目的源码已上传到 github , <https://github.com/ZZKdev/class-site> 可以查看

第三节 项目实现的主要功能

3.1 文件的分类功能

- 3.2 用户端文件的上传与下载，设置密码
- 3.3 消息通知的发布与查看，并按照时间顺序排序
- 3.4 课程安排表
- 3.5 图片的上传与显示
- 3.6 管理员界面对各个功能的管理

第四节 项目的介绍

左端是各个功能的列表，左下端可以从消息列表中提取出最新消息并显示，右端是编程常用网站与学校网站，点击左边列表项即可在中间显示相应的内容，未点击时则显示新闻列表中的最新新闻。

第五节 django 入门

5.1 初步了解 django

Django 是一个开放源代码的 Web 应用框架，由 Python 写成。采用了 MVC 的框架模式，即模型 M，视图 V 和控制器 C。它最初是被开发来用于管理劳伦斯出版集团旗下的一些以新闻内容为主的网站的，即是 CMS（内容管理系统）软件。并于 2005 年 7 月在 BSD 许可证下发布。这套框架是以比利时的吉普赛爵士吉他手 Django Reinhardt 来命名的。

5.2 开始 django 的第一步使用

5.2.1 新建一个 django project

打开命令行界面，输入 `django-admin startproject learning_django`

```
C:\Users\ZZK>django-admin startproject learning_django
C:\Users\ZZK>
```

其中 `learning_django` 是项目名称，你也可以换成你喜欢的名字，现在，我们就可以在该目录下看到名为 `learning_django` 的项目文件夹了，然后使用 `cd learning_django` 进入到项目目录中

5.2.2 新建一个 django 的 app

现在，我们已经进入了项目文件夹中，在命令行继续输入
`python manage.py startapp learning_app` 创建 app

```
C:\Users\ZZK>django-admin startproject learning_django
C:\Users\ZZK>cd learning_django
C:\Users\ZZK\learning_django>
```

其中 `learning_django` 是我们新建的 app 名称，`manage.py` 则是我们在项目创建时 django 自己为我们项目创建的一个 py 文件，它可以用来管理我们的项目，比如创建 app。

5.2.3 编写 app

3.1 编写 html 文件

首先先在 `learning_django` 目录下创建一个新的文件夹，并命名为 `templates`，该文件夹用来存放我们的 html 文件，文件名不可写错，否则 django 将找不到我们的 html 文件，然后再 `templates` 文件夹中新建一个 html 文件，并命名为 `learing.html`，名字没有限制，写入下列代码：

```

1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>learing django</title>
5          <meta charset="utf-8">
6      </head>
7      <body>
8          <p>hello ZZK!!</p>
9      </body>
10 </html>
11

```

保存，如果是用记事本打开编辑的记得保存时要使用utf-8的编码模式。

3.2 编写视图函数

进入我们的 app 目录下找到 views.py 文件（记住是 app 目录下），打开并写入下列代码：

```

1  from django.shortcuts import render
2
3  # Create your views here.
4  def index(request):
5      return render(request, 'learning.html')

```

在里面我们定义了一个名为 index 的函数，并且将 request 参数传递给函数，然后调用 render 函数并返回，该函数的第一个参数是 request，第二个参数则是我们要渲染的网页。当我们调用 index 函数时我们将会显示 learning.html 网页。

3.3 编写 url

现在我们回到 learning_django 项目的主目录下，我们会发现在项目的主目录下还有一个名为 learning_djanog 的文件夹，点击进入找到 urls.py 文件，打开并编辑：

```

from django.conf.urls import url
from django.contrib import admin

from learning_app.views import index

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^$', index)
]

```

其中 `urlpatterns` 是 `url` 的匹配模式，我们在其中添加了一个 `url` 匹配模式，‘`^$`’ 是一个正则表达式（`^` 表示开始，`$` 表示结束），表示开始就结束，也可以理解为我们的首页，`index` 则是我们之前编写的视图函数，也就是说，当我们来到首页的时候，我们会调用视图函数，视图函数会帮我们渲染网页。

3.4 运行

现在我们已经将 `html` 文件与视图函数和 `url` 模式都写好了，我们可以打开命令行界面并来到我们的项目主目录下，输入 `python manage.py runserver`

```

C:\Users\ZZK\learning_django>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).

You have 13 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin,
auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
November 29, 2017 - 12:23:40
Django version 1.11.3, using settings 'learning_django.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

```

如果出现上面的界面则表示服务器运行成功，否则就代表失败，失败的话可以回到前面看看哪里出现了问题，并修正。

现在打开我们的浏览器，在地址栏地输入 127.0.0.1:8000/（这是我们的主机地址），我们会看到下面的错误：

TemplateDoesNotExist at /

learning.html

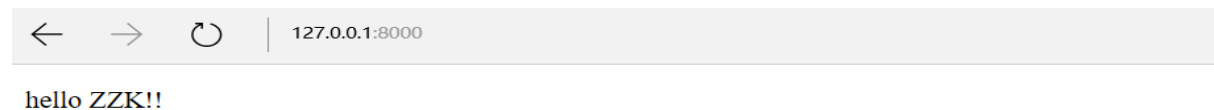
```
Request Method: GET
Request URL: http://127.0.0.1:8000/
Django Version: 1.11.3
Exception Type: TemplateDoesNotExist
Exception Value: learning.html
Exception Location: C:\Users\ZZK\Anaconda3\lib\site-packages\django\template\loader.py in get_template, line 25
Python Executable: C:\Users\ZZK\Anaconda3\python.exe
Python Version: 3.6.3
Python Path: ['C:\\Users\\ZZK\\learning_django',
'C:\\Users\\ZZK\\Anaconda3\\python36.zip',
'C:\\Users\\ZZK\\Anaconda3\\DLLs',
'C:\\Users\\ZZK\\Anaconda3\\lib',
'C:\\Users\\ZZK\\Anaconda3',
'C:\\Users\\ZZK\\Anaconda3\\lib\\site-packages',
'C:\\Users\\ZZK\\Anaconda3\\lib\\site-packages\\Babel-2.5.0-py3.6.egg',
'C:\\Users\\ZZK\\Anaconda3\\lib\\site-packages\\win32',
'C:\\Users\\ZZK\\Anaconda3\\lib\\site-packages\\win32\\lib',
'C:\\Users\\ZZK\\Anaconda3\\lib\\site-packages\\Pythonwin']
Server time: Wed, 29 Nov 2017 04:39:24 +0000
```

意思是找不到名为 learning.html 的文件，这是因为我们没有在 settings.py 里面添加我们的 learning_app，django 寻找 html 文件是在 settings 里面的 app 的 templates 文件夹中寻找 html 文件的，这也是为什么我们前面说文件夹名称一定要设为 templates。现在我们去到项目工程主目录下的 learning_django 文件夹，打开 settings.py 并在其中添加上我们的 app，代码如下：

```
19 # Quick-start development settings - unsuitable for production
20 # See https://docs.djangoproject.com/en/1.11/howto/deployment/checklist/
21
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = 'tyt*0rq@9#h2lj2!#9*-l=3ex%sqb(o+73obu4s81ki1s3*q3'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'learning_app',
35     'django.contrib.admin',
36     'django.contrib.auth',
37     'django.contrib.contenttypes',
38     'django.contrib.sessions',
39     'django.contrib.messages',
40     'django.contrib.staticfiles',
41 ]
42
43 MIDDLEWARE = [
44     'django.middleware.security.SecurityMiddleware',
```

在这里添加，注意后面不要少了个逗号

现在刷新页面，我们就可以看到我们的网页成功运行了。



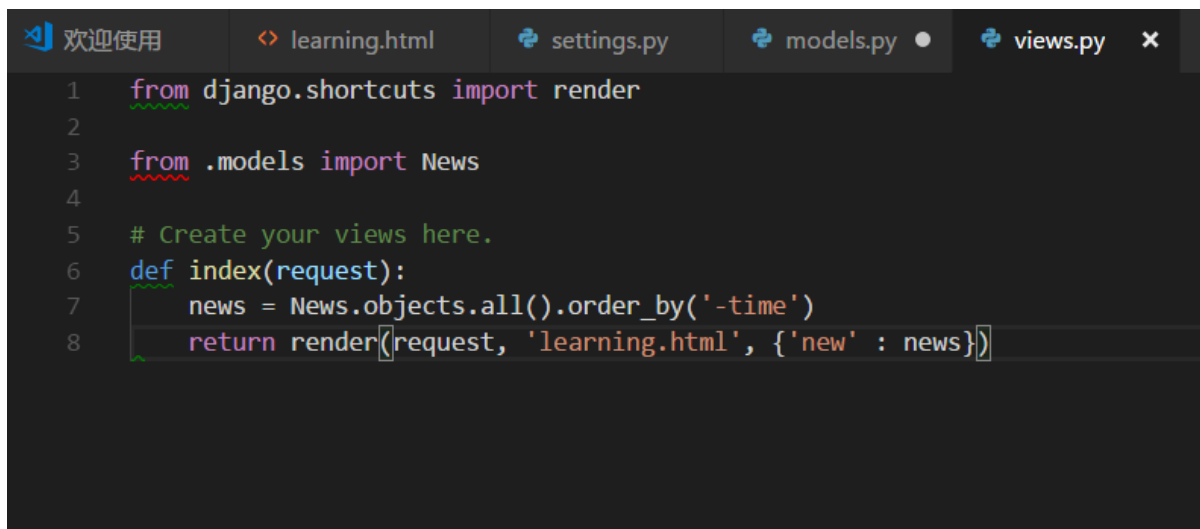
5.3 django 进阶（使用模板与模型）

之前我们已经做了一个小的网页，现在我们这节将学习如何使用模板与模型，首先我们先到我们的 app 目录下找到 models.py 文件打开并编辑：

```
1  from django.db import models
2
3  # Create your models here.
4  class News(models.Model):
5      title = models.CharField(max_length = 30)
6      time = models.DateTimeField(auto_now_add = True)
7      body = models.TextField()
8
```

我们在其中定义了一个名为 News（新闻）的模板类，该类继承了 models.Model（所有的模板类都要继承该类），在里面定义了 title, time, body，其中 title（用来存放标题）是 models.CharField 类，并为其设置了最大长度，time（用来存放时间）是 models.DateTimeField 类（auto_now_add = True 表示添加是自动为其添加时间），body（用来存放文章内容）则是 models.TextField 类，该类无字数限制，一般用来做长文本的输入。

现在回到我们的视图函数，修改其中代码：

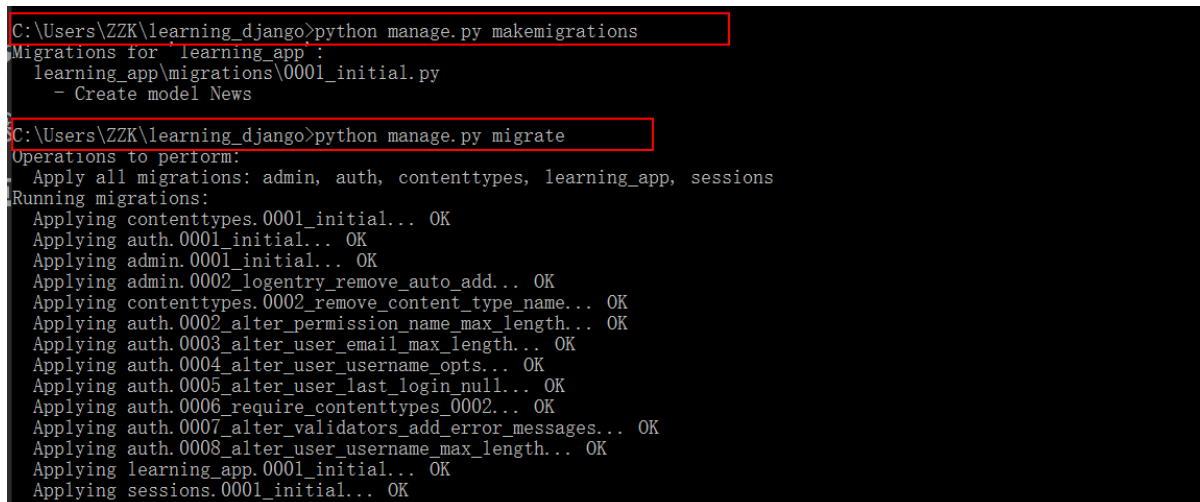


```
1 from django.shortcuts import render
2
3 from .models import News
4
5 # Create your views here.
6 def index(request):
7     news = News.objects.all().order_by('-time')
8     return render(request, 'learning.html', {'new' : news})
```

首先我们从当前目录的 models 中引入 News 类，然后在 index 中获得 News 类的所有对象并按时间倒排赋值给 news，我们可以观察到现在我们给 render 多传递了一个参数，这个参数是一个字典，我们可以在后面的 html 中使用字典的索引来使用它们。

现在我们要实现在管理员界面中添加数据：

打开命令行界面，输入下面代码：



```
C:\Users\ZZK\learning_django>python manage.py makemigrations
Migrations for 'learning_app':
  learning_app\migrations\0001_initial.py
  - Create model News

C:\Users\ZZK\learning_django>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, learning_app, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying learning_app.0001_initial... OK
  Applying sessions.0001_initial... OK
```

这两行代码帮我们创建并同步了数据库表，用来存储我们的数据。

现在我们来创建一个管理员账户：

继续在命令行输入下面代码：

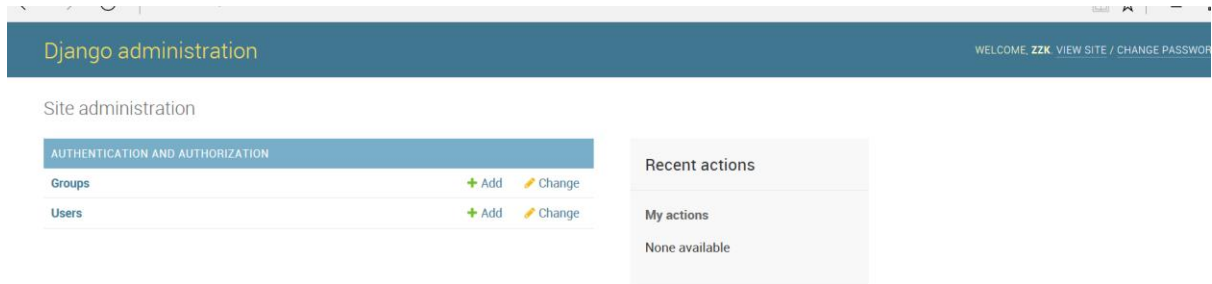


```
C:\Users\ZZK\learning_django>python manage.py createsuperuser
Username (leave blank to use 'zzk'): ZZK
Email address: 2657530327@qq.com
Password:
Password (again):
Superuser created successfully.

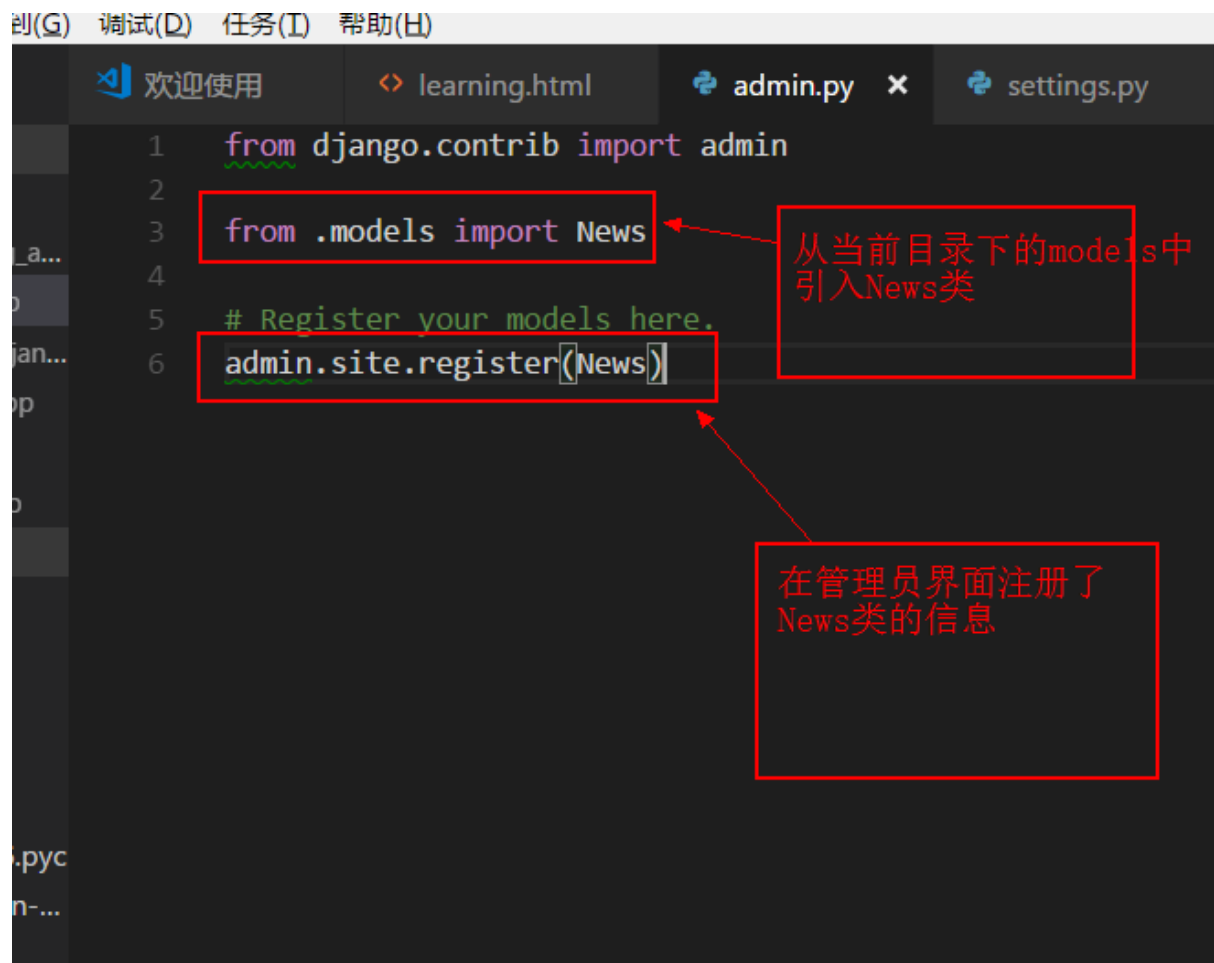
C:\Users\ZZK\learning_django>
```

按要求输入就可以创建管理员账户了

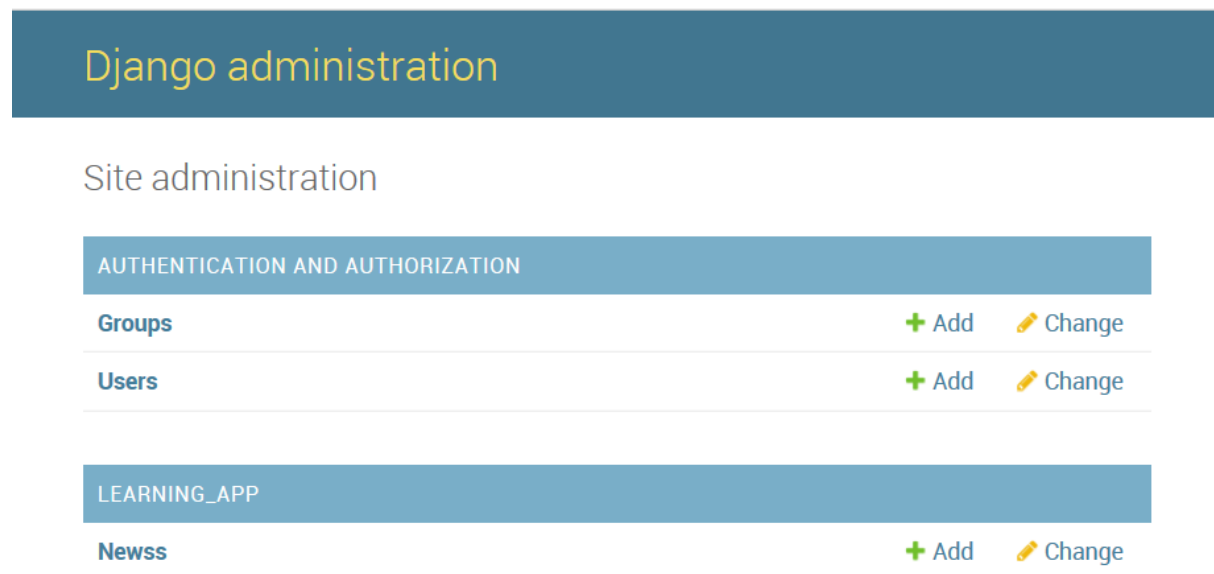
现在再次运行服务器，并打开 `http://127.0.0.1:8000/admin` 就可以看到我们的管理员登陆界面了，输入之前填写的用户名和密码即可登陆，界面如下：



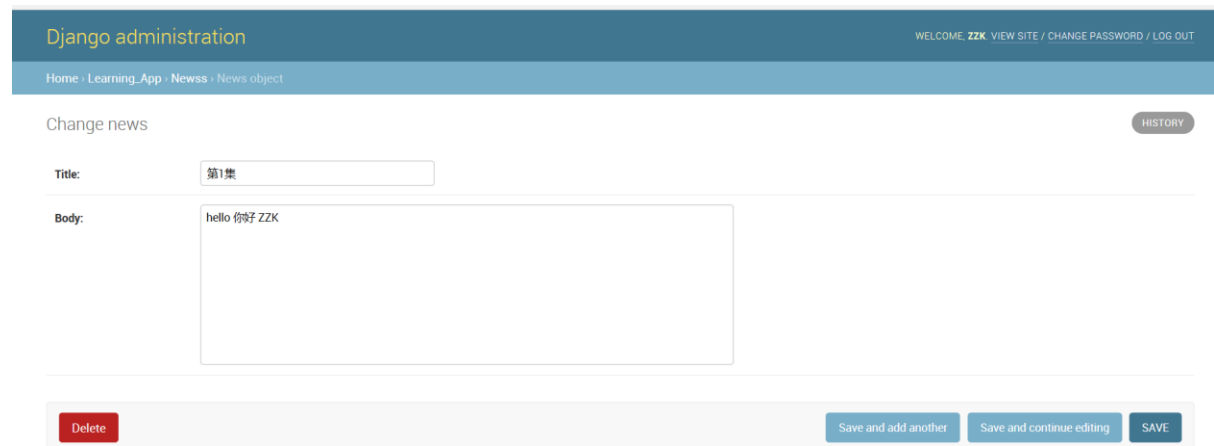
我们现在要实现在管理员界面添加数据，我们可以来到 `app` 目录下的 `admin.py` 文件打开并编辑：



打开 <http://127.0.0.1:8000/admin>，现在我们可以管理员界面中添加 News 类的数据了：

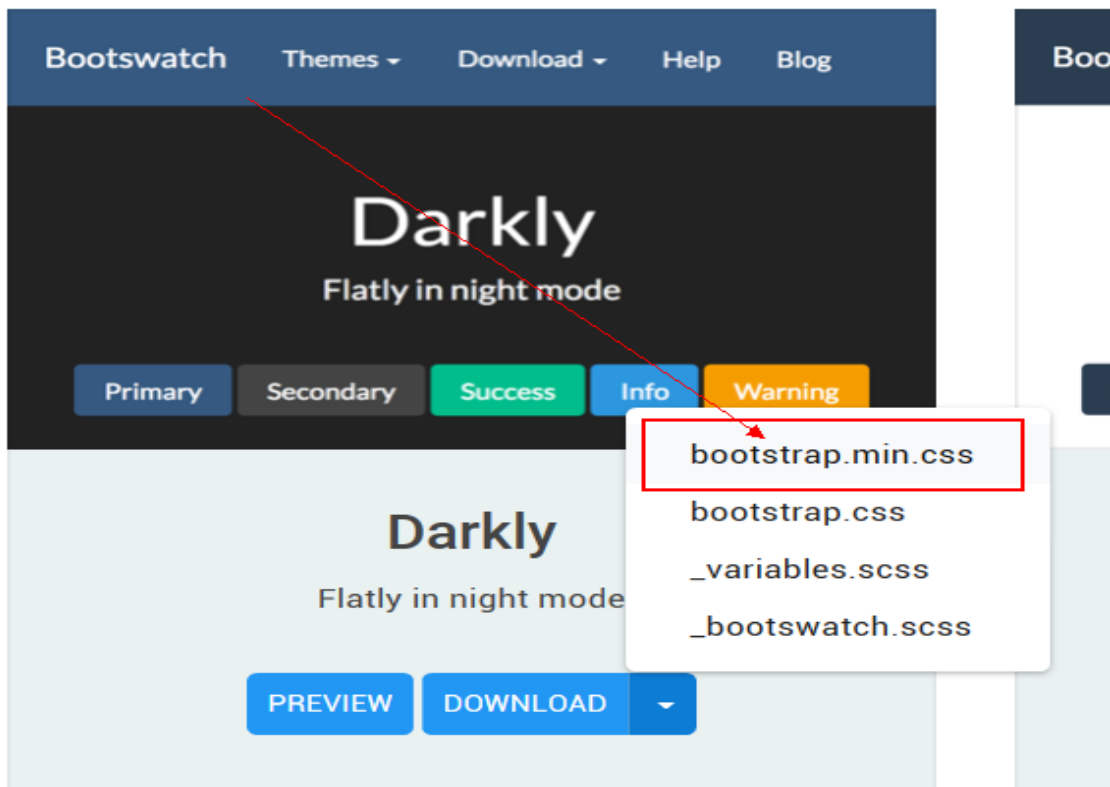


点击 Add 添加数据（Save 保存，按照这样添加三个 News 类数据）：



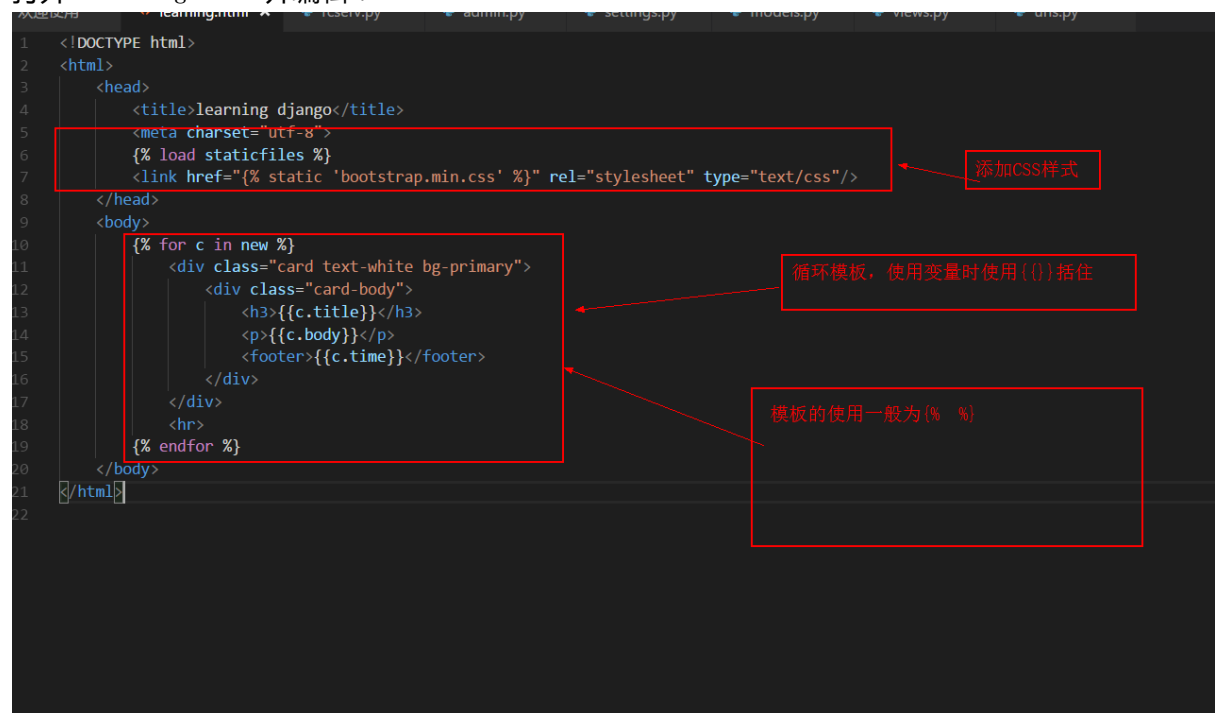
编写 html：

首先因为我们要引入 bootstrap，所以我们先在 app 目录下创建一个 static 的文件夹（名字一定要为 static，不然将会找不到我们的 css 样式），并到 <https://bootswatch.com/> 网站上下载我们的 bootstrap 主题（本教程使用的是 Darkly 主题，下载的文件是 bootstrap.min.css 文件），然后将 bootstrap.min.css 文件存放到 static 文件夹中。



进入 settings.py 文件中查看是否有下面的一行代码，一般 django 会自动帮我们加上，没有的话就加加上。

打开 learning.html 并编辑：



现在我们打开 <http://127.0.0.1:8000/> 可以看到存放在数据库被我们使用模板循环打印出来了，而且是按照时间顺序排列：



在 django 中我们不止有 for 循环的模板，还有许多其他的模板，比如 if 模板，这些我们会在后面讲到。

第六节 开始我们项目的制作

前面我们已经对 django 有了一个大概的了解，现在我们开始通过一个项目来逐步熟练 django。

6.0 创建一个 django 项目

命名为 classsite，并添加一个 app 为 classpage。

6.1 编写我们的 models 模型

我们一共需要下面五个类分别为：News（新闻类），Announcement（通知类），File_math（数学文件），File_language（语言文件类），File_other（其他文件类）。代码如下（省略了 File_others 类）：

```

# Create your models here.
class News(models.Model):
    titlemax = models.CharField(max_length = 30)
    titlemin = models.CharField(max_length = 40)
    introduction = models.CharField(max_length = 150)
    time = models.DateTimeField(auto_now_add = True)
    body = models.TextField()
    photot = models.ImageField()

class Announcement(models.Model):
    title = models.CharField(max_length = 30)
    body = models.TextField()
    time = models.DateTimeField(auto_now_add = True)
    Author = models.CharField(max_length = 20)

class File_math(models.Model):
    name = models.CharField(max_length = 30)
    file_ppt = models.FileField()
    time = models.DateTimeField(auto_now_add = True)
    people = models.CharField(max_length = 30)

class File_language(models.Model):
    name = models.CharField(max_length = 30)
    file_ppt = models.FileField()
    time = models.DateTimeField(auto_now_add = True)
    people = models.CharField(max_length = 30)

```

大标题和小标题

简要的介绍

正文和图片，图片我们这里使用了一个新的类为models.ImageField(),用来存放图片

标题，正文，时间，发布人

文件名，文件，时间，发布人

与上面的类一样，下面的File_others类也是

6.2 在 admin 注册我们的模型

```

from django.contrib import admin
from classpage.models import News
from classpage.models import Announcement
from classpage.models import File_language, File_math, File_others

# Register your models here.

admin.site.register(News)
admin.site.register(Announcement)

admin.site.register(File_language)
admin.site.register(File_math)
admin.site.register(File_others)

```

6.3 在命令行中创建数据库表

```

C:\Users\ZZK\classsite>python manage.py makemigrations

C:\Users\ZZK\classsite>python manage.py migrate_

```

6.4 创建管理员账号（参考前面教程，现在我们可以管理员界面添加各个类的数据了，只是现在还没有在具体文件中使用）

6.5 分析并编写主页 class_home_page.html（更多详情可到 github 查看源代码，<https://github.com/ZZKdev/class-site>）



首先我们将主页分为八个部分，1 为一个浮动在顶部的导航栏，2，5，7 为三个大框将下面分为三个部分，2 中放置了一个列表（3）和一个 div（4），5 中放置了一个 div（6），6 其中主要放置了图片和文本和按钮。7 中放置了一个列表（8）。

其中 6 这个盒子是我们经常改变的地方，当我们切换画面时，变换的就只有 6 这个部分，django 中提供了 html 的继承功能，继承完之后 6 这个盒子是我们要修改的地方，所以我们在 6 这个框外加上两行代码：



6.6 编写主页视图

代码如下：

```

from django.shortcuts import render
from django.http import HttpResponse
from .models import News, Announcement, File_math, File_language, File_others
from django.core.paginator import Paginator, PageNotAnInteger, EmptyPage
# Create your views here.

def index(request):
    news_list = News.objects.all().order_by('-time')
    paginator = Paginator(news_list, 1)

    page = request.GET.get('page')
    try:
        news = paginator.page(page)
    except PageNotAnInteger:
        news = paginator.page(1)
    except EmptyPage:
        news = paginator.page(paginator.num_pages)
    First = Announcement.objects.last()
    return render(request, 'class_home_page.html', {'news' : news, 'First' : First})

```

需求：我们需要将新闻类的所有对象进行分页，实现下一页的功能。

首先引入 News 类，再引入 django 中自带的分页管理器，在 index 函数中，第一行首先是获取 News 类的所有对象并按时间顺序倒排，第二行是用 news_list 实例化一个分页对象，Paginator 的第二个参数是每页对象个数，在这里我们设为 1，第三行用来获取页码。

try:

news = paginator.page(page) 获取当是页数的对象

except PageNotAnInteger: 如果页数不是为整数，则跳转到第一页

news = paginator.page(1)

except EmptyPage: 如果页数过大则跳转到最后一页

news = paginator.page(paginator.num_pages)

First = Announcement.objects.last() 这一行我们暂时不管。

最后一行返回一个字典。

现在回到我们 class_home_page.html 中：

```

{% block content %}
    {% for new in news %}
        <div class="card">
            <h3 class="card-header">最新新闻</h3>
            <div class="card-body">
                <h5 class="card-title">{{ new.titlemax }}</h5>
                <h6 class="card-subtitle text-muted">{{ new.titlemin }}</h6>
            </div>
            <img style="height: 200px; width: 100%; display: block;" src= {{ new.photot.url }}>
            <div class="card-body">
                <p class="card-text">{{ new.introduction }}</p>
                <a class="btn btn-primary" type="button" href="/news/{{ new.id }}">进去看看</a>
                {% if news.has_next %}
                    <a class="btn btn-primary" type="button" href="?page={{ news.next_page_number }}" style="float: right">看看别的</a>
                {% else %}
                    <a class="btn btn-primary disabled" type="button" href="#" style="float: right">好像已经没有了</a>
                {% endif %}
            </div>
            <div class="card-footer text-muted">
                {{ new.time }}
            </div>
        </div>
    {% endfor %}
{% endblock %}
iv>

```

首先是一个 for 循环（因为 news 获得的是一个列表，但由于我们每页的对象个数为 1，所以只循环一次），输出 new 的各个属性值，然后是一个 if 模板，判断 news 页有没有下一页，有的话则看到一个可点击的按钮，可以带我们到下一页（注意 href 路径，它将带我们跳转到?page={news.next_page_number}），这里不需要我们再写 url 的匹配模式，django 的 Paginator 类已经帮我们处理好了，news.next_page_number 是下一页的页码），没有的话则显示一个不可点击的按钮。为了使我们新闻类的上传的图片可以正常地显示出来，我们还要在 settings.py 更改一些东西，如下代码：

```
9 # Static files (CSS, JavaScript, Images)
10 # https://docs.djangoproject.com/en/1.11/howto/static-files/
11
12 MEDIA_URL = '/uploads/'
13 MEDIA_ROOT = os.path.join(BASE_DIR, 'uploads')
```

在 settings.py 中添加这两行代码，并在项目主目录下创建 uploads 文件夹。

再来到 urls.py 文件，添加下面这行代码：

```
23 from classpage.views import uploads_file_view, uploads_success_view
24
25 urlpatterns = [
26     url(r'^admin/', admin.site.urls),
27     url(r'^$', index),
28     url(r'^news/(?P<pk>\d+)$', DetailView.as_view(
29         model = News,
30         template_name = "newspage.html",
31     )),
32     url(r'^course/$', course_view),
33     url(r'^announce/$', Announce_view),
34     url(r'^file/$', file_view),
35     url(r'^file/math/$', file_view_math),
36     url(r'^file/language/$', file_view_language),
37     url(r'^file/others/$', file_view_others),
38     url(r'^uploads/$', uploads_file_view),
39     url(r'^uploads/success$', uploads_success_view)
40 ]+static(settings.MEDIA_URL, document_root = settings.MEDIA_ROOT)
41
```

回到 class_home_page.html 中，我们要查看图片的话不能仅仅使用{{ new.photot }}，这个返回的仅仅是图片的名字，要使图片正常显示，我们还得在他后面添加个 url 获取图片路径。

```

{% block content %}
    {% for new in news %}
        <div class="card">
            <h3 class="card-header">最新新闻</h3>
            <div class="card-body">
                <h5 class="card-title">{{ new.titlemax }}</h5>
                <h6 class="card-subtitle text-muted">{{ new.titlemin }}</h6>
            </div>
            <img style="height: 200px; width: 100%; display: block;" src= {{ new.phototot.url }}>
            <div class="card-body">
                <p class="card-text">{{ new.introduction }}</p>
                <a class="btn btn-primary" type="button" href="/news/{{ new.id }}">进去看看</a>
                {% if news.has_next %}
                    <a class="btn btn-primary" type="button" href="?page={{ news.next_page_number }}" style="float: right">下一页</a>
                {% else %}
                    <a class="btn btn-primary disabled" type="button" href="#" style="float: right">好像已经没有了</a>
                {% endif %}
            </div>
            <div class="card-footer text-muted">
                {{ new.time }}
            </div>
        </div>
    {% endfor %}
{% endblock %}

```

6.7 使用通用视图显示 News 类详细信息

我们先来到 class_home_page.html 页面将“进去看看”按钮的 href 的地址换成这样，把 new 对象的 id 值放到地址中，让后面的正则表达式匹配。

```

<a class="btn btn-primary" type="button" href="/news/{{ new.id }}">进去看看</a>
{% if news.has_next %}
    <a class="btn btn-primary" type="button" href="?page={{ news.next_page_number }}" style="float: right">下一页</a>
{% else %}
    <a class="btn btn-primary disabled" type="button" href="#" style="float: right">好像已经没有了</a>
{% endif %}
</div>

```

再来到 urls.py 文件

```

from django.conf.urls import url
from django.contrib import admin
from django.conf import settings
from django.conf.urls.static import static
from django.views.generic import DetailView
from classpage.models import News
from classpage.views import index, course_view, Announce_view, file_view, file_view_math, file_view_la
from classpage.views import uploads_file_view, uploads_success_view

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^$', index),
    url(r'^news/(?P<pk>\d+)$', DetailView.as_view(
        model = News,
        template_name = "newspage.html",
    )),
]

```

首先引入一个通用视图类和 News 类，url 模式中添加该模式，其中 '^news/(?P<pk>\d+)\$' 是一个正则表达式，从 news/ 开始后面表示捕获一个值（这个值为 News 类对象的 id 值，id 值是创建数据库表时赋予 News 类对象的一个属性值）并传递给 DetailView.as_view 函数，在 DetailView.as_view 函数中我们传递了两个参数给它，分别是我们要使用的模型和 html 文件，我们在 html 引用该对象的方法是使用该类的小写来引用。

下面是我们的 newspage.html 文件：

```

1 {% extends "class_home_page.html" %}
2
3 {% block content %}
4     <h3 style="text-align: center">{{news.titlemax}}</h3>
5     <p class="text-muted" style="text-align: center">{{ news.time }}</p>
6     <hr>
7     <img style="height: 200px; width: 100%; display: block;" src= {{ news.photot.url }}>
8     <hr>
9     <p style="text-indent: 2em">{{ news.body|linebreaksbr }}</p>
10    <br><hr>
11    <a class="btn btn-primary btn-lg btn-block" type="button" href="/">带我回主页</a>
12
13 {% endblock %}
14 {% block texterror %}
15 <h4>看那边----></h4>
16 <footer class="blockquote-footer text-warning" style="float:right;">( ' ' ) ^ _ _ </footer>
17 {% endblock %}

```

首先我们先继承了 class_home_page.html 文件，然后重写了在原来 class_home.page.html 文件中被

```
{% block content %}
{% endblock %}
```

括住的部分，在里面我们引用对象时我们使用的是该类的小写来引用，

使用{{ news.body | linebreaksbr }}表示在 news.body 中文本的换行和空格可以被正常显示。

下面的 {% block texterror %}

{% endblock %}重写了 class_home_page.html 中被它标识的另一个部分。

6.8Announcement 类的功能实现

前面我们已经将 News 类的所有功能都已经写好了，现在我们开始写 Announcement 类（通知类）的功能，之前我们已经将 Announcement 类的模型写好并在 admin 上注册了，现在我们只要将 Announcement 类的视图函数、url 模板匹配和 html 文件写好就可以了，首先来到 veiws.py，下面是我们的视图代码。

```

1
2
3 def Announce_view(request):
4     Announce_list = Announcement.objects.all().order_by('-time')
5
6     First = Announcement.objects.last()
7
8     return render(request, 'Announce.html', {'announce_list' : Announce_list, 'First' : First})
9
10
11
12

```

第一行是获取所有 Announcement 类对象并按时间顺序倒排，第二行是获取最新的 Announcement 对象，最后放到字典中传递给 Announce.html 文件。

现在开始写 Announce.html 文件：

```

1 {% extends "class_home_page.html" %}
2
3 {% block content %}
4     {% if announce_list %}
5         {% for announce in announce_list %}
6             <div class="card text-white bg-primary">
7                 <div class="card-body">
8                     <h3 class="text-warning">{{ announce.title }}</h3>
9                     <p class="text-muted">{{ announce.time }}</p>
10                    <hr>
11                    <p style="text-indent: 2em;">{{ announce.body | linebreaksbr }}</p>
12                    <footer class="blockquote-footer" style="float:right;">作者: <cite title="Source Title">{{ announce.Author }}</cite></footer>
13                </div>
14            </div>
15            <hr>
16        {% endfor %}
17    {% else %}
18        <div class="card text-white bg-primary">
19            <div class="card-body">
20                <h3 class="text-warning">无通知</h3>
21                <hr>
22                <p>现在还没有什么通知，过会再看看吧。</p>
23            </div>
24        </div>
25    {% endif %}
26 {% endblock %}

```

这里要对传进来的 `announce_list` 先进行判断，看它是否为空。在这里我们还传递了一个 `First` 给 `Announce.html` 那么它在哪里被使用了呢，我们可以来到 `class_home_page.html` 文件看看：

```

<div class="card-body">
  <blockquote class="card-blockquote">
    {% block texterror %}
      {% if First %}
        <h3 class="text-warning">最新通知</h3>
        <p>{{ First.body | linebreaksbr }}</p>
        <footer class="blockquote-footer" style="float:right;">作者: <cite title="Source Title">{{ First.Author }}</cite></footer>
        <footer class="blockquote-footer" style="float:right;"><cite title="Source Title">{{ First.time }}</cite></footer>
      {% else %}
        <h3 class="text-warning">最新通知</h3>
        <p>现在还没有什么通知，过会再看看吧。</p>
      {% endif %}
    {% endblock %}
  </blockquote>
</div>

```

`First` 就在这里被引用了。而且之前我们也有看到其他的视图函数传递了 `First`，它们都会在这里被使用，我们还可以看到这里可以被重写，前面的 `newspage.html` 就重写了它。至于其他的细节可以参考前面的文档。

最后就是我们的 `url` 匹配了：

```

from django.contrib import admin
from django.conf import settings
from django.conf.urls.static import static
from django.views.generic import DetailView
from classpage.models import News
from classpage.views import index, course_view, Announce_view, file_view, file_view_math, f
from classpage.views import uploads_file_view, uploads_success_view

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^$', index),
    url(r'^news/(?P<pk>\d+)$', DetailView.as_view(
        model = News,
        template_name = "newspage.html",
    )),
    url(r'^course/$', course_view),
    url(r'^announce/$', Announce_view),
    url(r'^file/$', file_view),
    url(r'^file/math/$', file_view_math),

```

引入视图函数并写上url匹配模式，' ^ announce/\$' 表示announce/页面。

6.9 编写文件下载功能

首先将下面的点击课程文件显示分类的功能写出来：



编写 File_list.html 文件：

```
{% extends "class_home_page.html" %}

{% block content %}
    <div class="list-group">
        <a class="list-group-item list-group-item-action text-danger" href="/file/math">
            高数
        </a>
        <a class="list-group-item list-group-item-action text-success" href="/file/language">
            C语言
        </a>
        <a class="list-group-item list-group-item-action text-info" href="/file/others">
            其他
        </a>
    </div>
{% endblock %}
```

编写 url 匹配模式：

```
urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^$', index),
    url(r'^news/(?P<pk>\d+)$', DetailView.as_view(
        model = News,
        template_name = "newspage.html",
    )),
    url(r'^course/$', course_view),
    url(r'^announce/$', Announce view),
    url(r'^file/$', file_view),
    url(r'^file/math/$', file_view_math),
    url(r'^file/language/$', file_view_language),
```

现在我们开始编写点击高数按钮后的界面（C语言和其他按钮同它一样）

编写 url 匹配模式：


```

from classpage.views import index, course_view, Announce_view, f
from classpage.views import uploads_file_view, uploads_success_v

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^$', index),
    url(r'^news/(?P<pk>\d+)$', DetailView.as_view(
        model = News,
        template_name = "newspage.html",
    )),
    url(r'^course/$', course_view),
    url(r'^announce/$', Announce_view),
    url(r'^file/$', file_view),
    url(r'^file/math/$', file_view_math),
    url(r'^file/language/$', file_view_language),
    url(r'^file/others/$', file_view_others),
    url(r'^uploads/$', uploads_file_view),
    url(r'^uploads/success$', uploads_success_view)
]
+static(settings.MEDIA_URL, document root = settings.MEDIA ROOT

```

编写视图函数：

```

7 def file_view_math(request):
8     File_list = File_math.objects.all().order_by('-time')
9     return render(request, 'File.html', {'File_list' : File_list})
0

```

编写 File.html 文件：

```
{% extends "class_home_page.html" %}

{% block content %}
    {% if File_list %}
        {% for File in File_list %}
            <div class="card text-white bg-primary">
                <div class="card-body">
                    <h3 class="text-warning">{{ File.name }}</h3>
                    <p class="text-muted">{{ File.time }}-----{{ File.people }}</p>
                    <hr>
                    <a class="btn btn-info btn-lg btn-block" href="{{ File.file_ppt.url }}">下载</a>
                </div>
            </div>
        {% endfor %}
    {% else %}
        <div class="card text-white bg-primary">
            <div class="card-body">
                <h3 class="text-warning">暂无文件</h3>
            </div>
        </div>
    {% endif %}
    <a class="btn btn-success btn-lg btn-block" type="button" href="/file/">带我回上一层</a>
{% endblock %}
```

注意{{ File.file_ppt.url }}后面记得要加上 url（原理同之前上传图片的问题），现在我们文件的下载功能就实现好了，我们可以在管理员界面中先传文件，然后在点击课件文件，找到相应的目录点击下载。

6.10 实现用户的文件上传

编写 uploads_file.html:

```
{% extends "class_home_page.html" %}

{% block content %}
    <form action="/uploads/success" method="post" enctype="multipart/form-data">
        {% csrf_token %}
        <fieldset>
            <div class="form-group">
                <label for="Select_project">选择科目</label>
                <select class="form-control" id="Select_project" name="project">
                    <option value="1">数学</option>
                    <option value="2">C语言</option>
                    <option value="3">其他</option>
                </select>
            </div>
            <div class="form-group">
                <label for="file">选择文件</label>
                <input class="form-control-file" id="file" type="file" name="file_ppt">
            </div>
            <div class="form-group">
                <label for="passkey">输入密码</label>
                <input class="form-control" id="passkey" type="password" placeholder="密码" name="passkey">
            </div>
            <div class="form-group">
                <label for="input_file_name">输入文件名称</label>
                <input class="form-control" id="input_file_name" type="text" name="name">
            </div>
            <div class="form-group">
                <label for="input_name">输入你的名字</label>
                <input class="form-control" id="input_name" type="text" name="people">
            </div>
            <button class="btn btn-primary" type="submit">确认上传</button>
        </fieldset>
    </form>
{% endblock %}
```

在这里我们有一个表单，在表单里面加入 `{% csrf_token %}` 来防止 csrf 攻击（不加的话 django 会给我们报 403 错误信息），在每一个输入框中都有 name 属性用来让我们在视图函数中找到它们提交的信息。form 中的 action 属性为我们提交表单时跳转到的页面，method 为我们提交表单的方式，enctype 属性要设置为 multipart/form-data 才会将文件的数据发送出去。

编写文件上传的视图函数和 url 匹配模式：

```
def uploads_file_view(request):
    return render(request, 'uploads_file.html')

url(r'^file/others/$', file_view_others),
url(r'^uploads/$', uploads_file_view),
```

编写处理文件视图函数：

```
return render(request, 'uploads_file.html')

def uploads_success_view(request):
    if request.POST['passkey'] == '2017414021' and request.POST['name'] and request.POST['people']:
        file = request.FILES.get('file_ppt', None)
        if file is None:
            return render(request, 'uploads_fail.html')
        else:
            with open('uploads/%s' % file.name, 'wb+') as f:
                for chunk in file.chunks():
                    f.write(chunk)
            if request.POST['project'] == '1':
                File_math.objects.create(
                    name = request.POST['name'],
                    people = request.POST['people'],
                    file_ppt = 'uploads/' + file.name
                )
            elif request.POST['project'] == '2':
                File_language.objects.create(
                    name = request.POST['name'],
                    people = request.POST['people'],
                    file_ppt = 'uploads/' + file.name
                )
            else:
                File_others.objects.create(
                    name = request.POST['name'],
                    people = request.POST['people'],
                    file_ppt = 'uploads/' + file.name
                )
            return render(request, 'uploads_success.html')
        else:
            return render(request, 'uploads_fail.html')
```

第一行代码首先判断密码是否正确，文件名和上传人是否为空，为空或密码错误则跳转到文件上传失败界面（uploads_fail.html），第二行则是获取文件如果为空则置为 None，然后判断文件是否为空，为空则跳转到失败界面，不为空的话就进行文件操作将文件分块写入 uploads 目录中。后面就是判断文件类型，然后将数据保持到数据库中。（file_ppt 实际上存储的是文件的地址）。

之后就将上传成功和失败的 html 文件写好就可以了，代码如下：

```
uploads_success.html x File.html File_list.html
1 {% extends "class_home_page.html" %}
2
3 {% block content %}
4     <p>成功</p>
5 {% endblock %}
```

```
1 {% extends "class_home_page.html" %}
2
3 {% block content %}
4 <div class="card text-white bg-danger">
5     <div class="card-body">
6         <blockquote class="card-blockquote">
7             <p>上传失败!!</p>
8             <p>请输入正确密码和信息</p>
9         </blockquote>
10    </div>
11 </div>
12 <hr>
13 <a class="btn btn-primary btn-lg btn-block" href="/uploads">返回上一层</a>
14 {% endblock %}
```

6.11 生活点滴功能的实现和课程表功能实现：

生活点滴功能则外链到了 <http://123.207.43.62:12345/>，这是另外一个项目，项目的代码在 (<https://github.com/POTION4/Pic-Ey>)

课程表功能则是一个 html 表格构成，代码如下：

```

欢迎使用    <> course.html x    <> uploads_file.html    <> class_home_page.html
1  {% extends "class_home_page.html" %}
2
3  {% block content %}
4  <table class="table table-striped table-hover table-bordered" style="
5      <thead class="table-warning">
6          <tr>
7              <th></th>
8              <th>星期一</th>
9              <th>星期二</th>
10             <th>星期三</th>
11             <th>星期四</th>
12             <th>星期五</th>
13         </tr>
14     </thead>
15     <tbody>
16         <tr>
17             <td>1</td>
18             <td>高数 (7B-202) </td>
19             <td>大学英语 (6D-203) </td>
20             <td>高数 (7B-202) </td>
21             <td></td>
22             <td>高数 (7B-202) </td>
23         </tr>
24         <tr class="table-danger">
25             <td>2</td>
26             <td>C语言 (6E-302) </td>
27             <td></td>
28             <td>C语言 (6E-302) </td>
29             <td></td>
30             <td>C语言</td>
31         </tr>
32         <tr class="table-info">

```

url 匹配模式和视图函数如下：

```

)),
url(r'^course/$', course_view),
url(r'^announce/$', Announce_view),

```

```
def course_view(request):
    First = Announcement.objects.last()
    return render(request, 'course.html', {'First' : First})
```

第七节 运行我们的项目

现在我们的项目的所有功能都已经实现好了，现在打开命令行界面，来到我们的工程目录下输入 `python manage.py runserver` 运行服务器，在浏览器中打开 `http://127.0.0.1:8000/` 就可以

看到我们的项目啦！

打开结果：

