## 简介

news模块主要包含的功能是教务处、校团委、学工部新闻的信息爬取与信息公示，各个学院/教师联系方式的写入和陈列，校历、校车时刻表、学校生活作息表的公示

news模块的后端代码储存在apps\news文件夹下，前端代码储存在templates\news文件夹下

## 模块

django的数据库模型在models.py 中构建，本模块的模型有四个，分别为TestScrapy（测试用例，也是教务通知的存储模型）、Xuegongnews（学工部通知的存储模型）、Qingchunnews（校团委通知的存储模型）、ContactPost（通讯录的存储模型）分别如下：

**class** TestScrapy(models.Model):*#教务处通知* title = models.CharField(max\_length=255) *#爬取通知的标题* text = models.TextField() *# 爬取通知的内容* source = models.CharField(max\_length=30) *# 爬取通知的来源页* time = models.CharField (max\_length=30) *# 爬取通知的日期* **class** Meta:  
 app\_label = **'news'** db\_table = **'test\_scrapy'**

**class** Xuegongnews(models.Model):*#学工部通知* title = models.CharField(max\_length=255) *#爬取通知的标题* text = models.TextField() *# 爬取通知的内容* source = models.CharField(max\_length=255) *# 爬取通知的来源页* time = models.CharField (max\_length=255) *# 爬取通知的日期* **class** Meta:  
 app\_label = **'news'** db\_table = **'xuegong\_news'**

**class** Qingchunnews(models.Model):*#校团委通知* title = models.CharField(max\_length=255) *#爬取通知的标题* text = models.TextField() *# 爬取通知的内容* source = models.CharField(max\_length=255) *# 爬取通知的来源页* time = models.CharField (max\_length=255) *# 爬取通知的日期* **class** Meta:  
 app\_label = **'news'** db\_table = **'qingchun\_news'**

**class** ContactPost(models.Model):*#通讯录* name = models.CharField(max\_length=100) *#姓名* email = models.CharField(max\_length=100)*# email。* faculty = models.CharField(max\_length=100) *# 所属院系。* tel = models.CharField(max\_length=100) *# 内部类 class Meta 用于给 model 定义元数据* **class** Meta:  
 app\_label = **'news'** db\_table = **'contact'  
  
 def** \_\_str\_\_(self): *# 函数 \_\_str\_\_ 定义当调用对象的 str() 方法时的返回值内容* **return** self.name *# return self.name 将姓名返回*

通过python manage.py makemigrations 和python manage.py migrate 进行数据库迁移

## 视图

Django的视图在views.py中构建，本模块的视图有以下12个，通过views完成前后端数据的传值和操作。其中数据的操作通过django自带的数据处理器完成。

因为前端所有页面都会继承一个用户信息页面，因此所有的方法里都有获取当前用户-获取当前用户相关数据的代码段

**def** inform(request): *# 定义通知类* user = request.user *#获取当前用户* **if** user **is not** None:*#判定是否存在* user\_informs = UserProfile.objects.filter(username=user)*#运用筛选器获取当前用户信息* informs = models.TestScrapy.objects.all()*#获取表中的所有对象* **return** render(request, **'news/inform.html'**, locals())  
  
  
**def** inform\_detail(request, inform\_id): *# 定义通知详情页，从前端获取传值* user = request.user *#获取当前用户* **if** user **is not** None:*#判定是否存在* user\_informs = UserProfile.objects.filter(username=user)*#运用筛选器获取当前用户信息* inform = get\_object\_or\_404(models.TestScrapy, id=inform\_id)*#获取表中所有对象* **return** render(request, **'news/inform\_detail.html'**, locals())  
  
  
  
  
**def** xiaoche(request): *# 定义校车页* user = request.user *#获取当前用户* **if** user **is not** None:*#判定是否存在* user\_informs = UserProfile.objects.filter(username=user)*#运用筛选器获取当前用户信息* **return** render(request, **'news/inform\_service/xiaoche.html'**, locals())

**def** zxsj(request): *# 定义作息时间页* user = request.user *#获取当前用户* **if** user **is not** None:*#判定是否存在* user\_informs = UserProfile.objects.filter(username=user)*#运用筛选器获取当前用户信息* **return** render(request, **'news/inform\_service/zxsj.html'**, locals())  
  
  
**def** xiaoli(request): *# 定义校历页* user = request.user *#获取当前用户* **if** user **is not** None:*#判定是否存在* user\_informs = UserProfile.objects.filter(username=user)*#运用筛选器获取当前用户信息* **return** render(request, **'news/inform\_service/xiaoli.html'**, locals())  
  
  
**def** xuegong(request): *# 定义学工通知类* xuegongs = models.Xuegongnews.objects.all()*#获取表中所有对象* user = request.user *#获取当前用户* **if** user **is not** None:*#判定是否存在* user\_informs = UserProfile.objects.filter(username=user)*#运用筛选器获取当前用户信息* **return** render(request, **'news/xuegong.html'**, locals())  
  
  
**def** xuegong\_detail(request, xuegong\_id): *# 定义学工通知详情页，从前端获取传值* user = request.user *#获取当前用户* **if** user **is not** None:*#判定是否存在* user\_informs = UserProfile.objects.filter(username=user)*#运用筛选器获取当前用户信息* xuegong = get\_object\_or\_404(models.Xuegongnews, id=xuegong\_id)  
 **return** render(request, **'news/xuegong\_detail.html'**, locals())  
  
  
**def** qingchun(request): *# 定义青春川大通知类* user = request.user *#获取当前用户* **if** user **is not** None:*#判定是否存在* user\_informs = UserProfile.objects.filter(username=user)*#运用筛选器获取当前用户信息* qingchuns = models.Qingchunnews.objects.all()  
 **return** render(request, **'news/qingchun.html'**, locals())  
  
  
**def** qingchun\_detail(request, qingchun\_id): *# 定义青春川大通知详情页，从前端获取传值* user = request.user *#获取当前用户* **if** user **is not** None:*#判定是否存在* user\_informs = UserProfile.objects.filter(username=user)*#运用筛选器获取当前用户信息* qingchun = get\_object\_or\_404(models.Qingchunnews, id=qingchun\_id)  
 **return** render(request, **'news/qingchun\_detail.html'**, locals())  
  
**def** daily(request):  
 user = request.user *#获取当前用户* **if** user **is not** None:*#判定是否存在* user\_informs = UserProfile.objects.filter(username=user)*#运用筛选器获取当前用户信息* contacts = models.ContactPost.objects.all()  
 **return** render(request, **'news/daily.html'**,locals())  
  
**def** contact\_create(request): *#通讯录添加处0  
 # 判断用户是否提交数据* **if** request.method == **"POST"**:  
 *# 将提交的数据赋值到表单实例中* contact\_post\_form = ContactPostForm(data=request.POST)  
 *# 判断提交的数据是否满足模型的要求* **if** contact\_post\_form.is\_valid():  
 *# 保存数据，但暂时不提交到数据库中* new\_contact = contact\_post\_form.save(commit=False)  
 *# 将新文章保存到数据库中* new\_contact.save()  
 *# 完成后返回到文章列表* **return** redirect(**"news:daily"**)  
 *# 如果数据不合法，返回错误信息* **else**:  
 **return** HttpResponse(**"表单内容有误，请重新填写。"**)  
 *# 如果用户请求获取数据* **else**:  
 *# 创建表单类实例* contact\_post\_form = ContactPostForm()  
 *# 赋值上下文* context = { **'contact\_post\_form'**: contact\_post\_form }  
 **return** render(request, **'news/contact\_create.html'**, context)  
  
  
**def** dashboard(request):  
 user = request.user *#获取当前用户* **if** user **is not** None:*#判定是否存在* user\_informs = UserProfile.objects.filter(username=user)*#运用筛选器获取当前用户信息* lostfound = LostfoundPost.objects.count()*#失物招领计数* salebuy = SalebuyPost.objects.count()*#二手交易计数* seekhelp = SeekhelpPost.objects.count()*#信息求助计数* advpart = AdPost.objects.count()*#广告兼职计数* jiaowu = models.TestScrapy.objects.count()*#教务通知计数* xuegong = models.Xuegongnews.objects.count()*#学工通知计数* qingchun =models.Qingchunnews.objects.count()*##校团委通知计数* lost\_rate = round(lostfound / (lostfound + salebuy + seekhelp + advpart) \* 100)*#失物招领占比* sale\_rate = round(salebuy / (lostfound + salebuy + seekhelp + advpart) \* 100)*#二手交易占比* seek\_rate = round(seekhelp / (lostfound + salebuy + seekhelp + advpart) \* 100)*#信息求助占比* adv\_rate = round(advpart / (lostfound + salebuy + seekhelp + advpart) \* 100)*#广告兼职占比* **return** render(request, **'news/dashboard.html'**, locals())

## 爬虫

本项目使用scrapy和django进行联用完成爬虫的编写和数据存储，分为爬虫配置和代码编写两部分。

#### 爬虫配置

django和scrapy联用的关键是一个组件scrapy-djangoitem，可以使用pip指令进行安装：

pip install scrapy-djangoitem

然后在item.py中引入django模型，

import scrapy

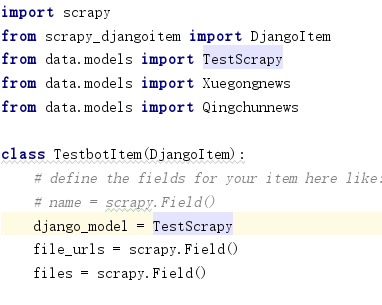
from scrapy\_djangoitem import DjangoItem

from models import ‘模型‘

class TestbotItem(DjangoItem):

django\_model = 模型

如下图所示：



#### 爬虫代码

爬虫代码有三份，基本逻辑相似，运用了replace方法进行数据清洗。



上图为对教务通知的爬取，运用css方法获取块对象，利用replace方法对转义符号进行了转换，完成数据清洗



上图为对学工通知的爬取，运用path方法获取块对象，利用replace方法对转义符号进行了转换，完成数据清洗；extract()[0]则是获取单个数据而非list的切片方法



上图为对学工通知的爬取，运用path方法获取块对象，利用replace方法对转义符号进行了转换，完成数据清洗