Report

Python Programming and Its Engineering Applications

Contents

Topic

Development of Blog system using Django framework

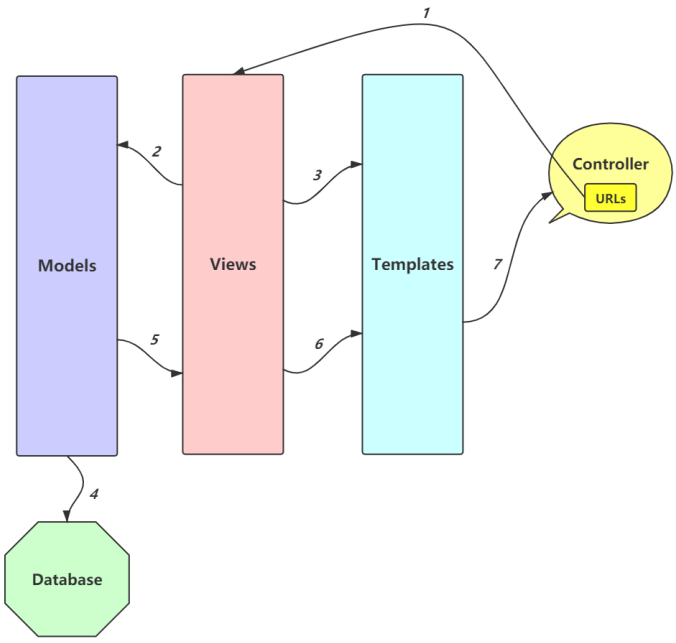
Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

You should design your own blog system with Django framework and the support of Sqilte3 database, and deploy your system.

Project Details

Implementation logic

Django implements concept of **Model-View-Template (MVT). MVT** is slightly different from **MVC**. In fact the main difference between the two patterns is that **Django** itself takes care of the Controller part (Software Code that controls the interactions between the Model and View), leaving us with the template. The template is a HTML file mixed with **Django Template Language (DTL)**.



To understand MVT, think Model as a Logical data structure. As the figure shown above, **Model** layer is the middleware and data handler between **Database** and **View** layer. The Model layer provides a definition of how the data formats when coming from the View layer and stores in the Database and how the retrieving information from the database transfers to the View layer in the displayable format.

Views layer receive data as well as request method ("POST","GET") from client side and accordingly formats the data via model so that it can be stored in database. It also communicates to the database for retrieving data which transfer to the template for viewing.

Django needs a convenient way to generate HTML dynamically. The most common approach relies on templates. A template contains the **static** parts of the desired HTML output as well as some special syntax describing how **dynamic** content will be inserted. One template can be used by different views to show various formats of data. It keeps all the content that is rendered by the browser. This part is what is visible to the client side. Model and Views layer reside on the server side.

When users click a page, front end will send a URL address to the Views layer. Views layer receive data and request, store data and retrieve data from database via Model layer, and finally transfer data to the Template layer. Static part and the inserted dynamic content of HTML template will be shown on the web page.

My Blog system

I designed the blog system using Django framework and Sqlite3 database that comes with Python. The web page template is modified from the Bootstrap Library for new developers.

The basic functions of my blog system include

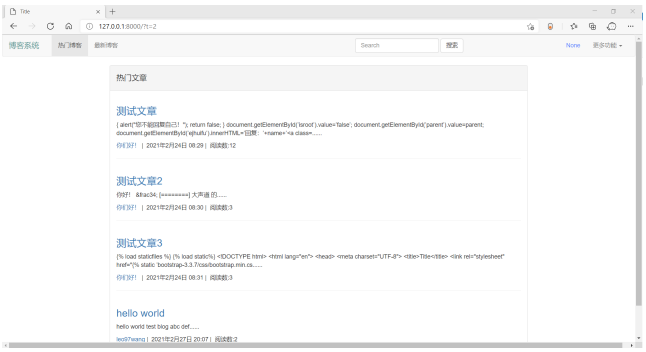
- User registration and login
- User information modification
- Blog posting, search, modification and deletion
- Read blog contents and browse all blogs.

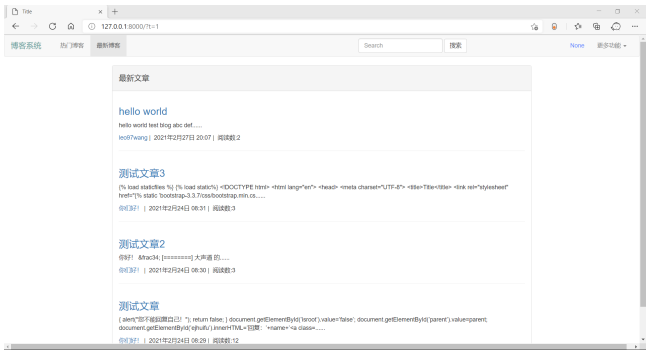
In addition, by registering super administrator accounts, backend developers have the right to manage information of all the users and blogs.

Function modules

Home page

Browse all the blogs, including reading in order of view numbers(Hot blogs) and reading in order of post time(Latest blogs).





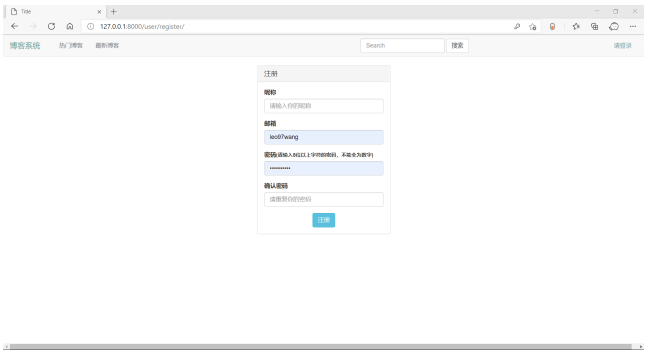
Function realization method

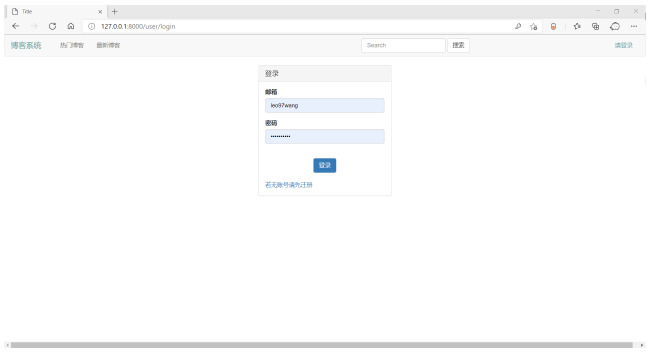
When users click button hot-blogs or latest-blogs, Views layer receive a data 't' from frontend. When t=1, Views layer retrieve data from database in order of post time and when t=2, it retrieve data in order of view numbers. Then those data will be inserted into template "home.html" and the web page will be displayed to clients.

```
def home(请求):
    t=请求.GET.get('t',-1)
    hotblog = Blog.objects.order_by('-viewnum')[0:20]
    if t == '1':
        hotblog = Blog.objects.order_by('-time')[0:20]
    return render(请求, 'home.html', { 'hotblog': hotblog,'t':t})
```

User registration & login

All the clients could use their email address and set a password to register a blog account and login via email and password.





Function realization method

Use clients login module as an example. When users click login button, controller will send the url: 127.0.0.1:8000/user/login to Views layer. 127.0.0.1:8000/user is the primary address

```
path('user/',include('user.urls')), #code in untitled/urls.py
```

/login is the secondary address

```
path('login',views.userlogin,name='登录'), #code in blog/urls.py
```

The function ***userlogin*** in Views layer will be called and the web page "login.html" will be displayed

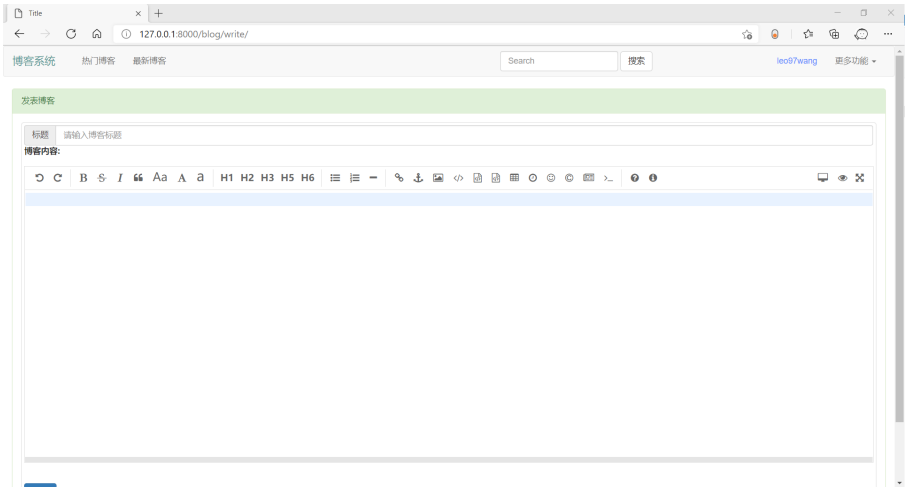
```
#code in blog/views.py
def userlogin(请求):
    if 请求.method =='POST':
        user=authenticate(请求,username=请求.POST['username'],password=请求.POST['password'])#如果有该用户则返回该用户，没有则返回空
        if user is None:
            return render(请求,'login.html',{'erro':'密码错误或用户不存在！'})
        else:
            login(请求,user)
            return redirect('主页')
    elif 请求.user.is_authenticated:
        return redirect('主页')
    else:
        return render(请求, 'login.html')
```

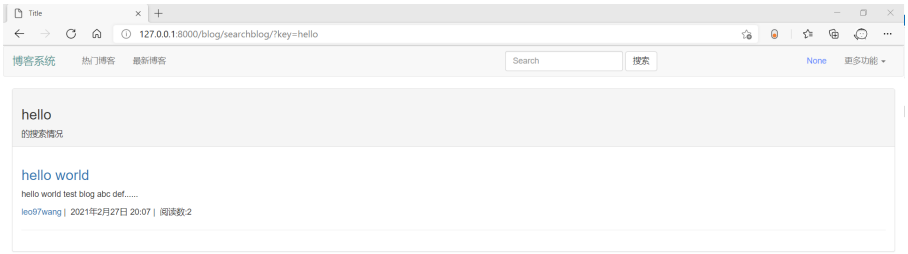
Then Views layer will get username and password from clients and match them in database. If users login successfully, they will see the home page.

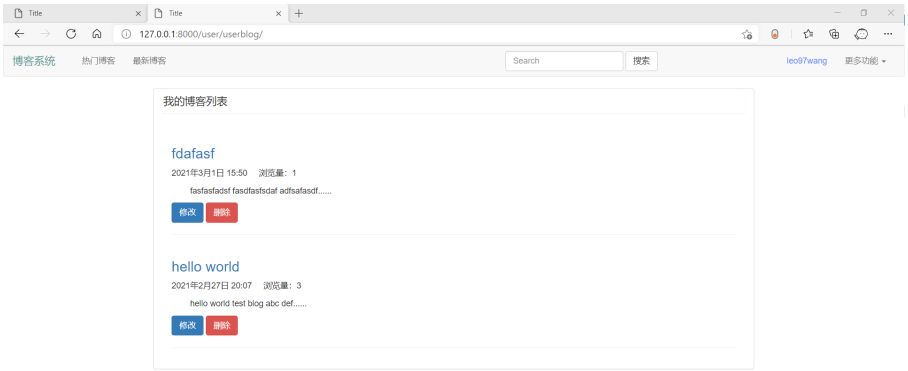
```
url('',views.home,name='主页') #code in untitled/urls.py

def home(请求): #code in untitled/views.py
    t=请求.GET.get('t',-1)
    hotblog = Blog.objects.order_by('-viewnum')[0:20]
    if t == '1':
        hotblog = Blog.objects.order_by('-time')[0:20]
    return render(请求, 'home.html', { 'hotblog': hotblog,'t':t})
```

Blog posting, search, modification and deletion

Users could write blog in markdown language, search blog with keywords, modify their blogs and delete their blogs in user blog page.







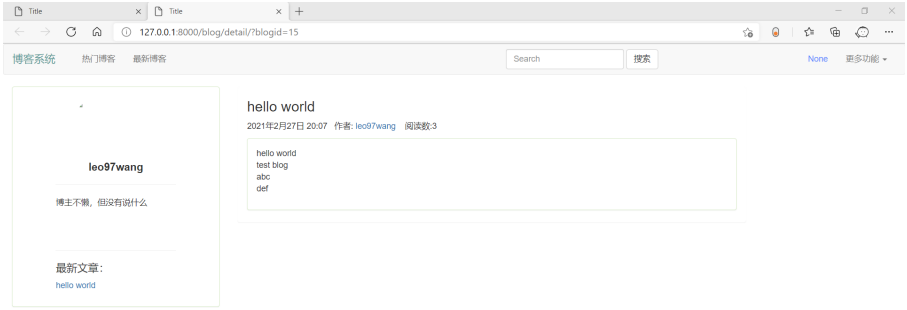
Function realization method

The package *django-mdeditor* allows developers to integrate markdown language when they write blogs. Each blog, when created, will be assigned with a unique number called blogid. Backend will use blogid receiving from frontend to retrieve the information of blog from database. These functions are all developed based on the retrieval of blog using blogid.

```python
#code in blog/views.py
#写博客
def writeblog(请求):
    if(请求.user.is_authenticated==False):
        return redirect('user:登录')
    if(请求.method == 'POST'):
        #获取前端页面填写在md编译器的博客内容
        str1=请求.POST['id_博客内容-wmd-wrapper-html-code']
        #用来去除前端md编译器的标签
        bloginfo=mdtotex(str1)[0:200]+'......'
        blog=Blog.objects.create(bloguser=请求.user,
                                 title=请求.POST['title'],
                                 bloginfo=bloginfo,
                                 blogtext=请求.POST['博客内容'],
                                 viewnum=1)
        return redirect('user:登录')
    else:
        #返回博客表单，用于填写博客内容
        form=AddBlogForm()
        return render(请求, 'writeblog.html',{'form':form})
#搜索博客
def searchblog(请求):
    keyword=请求.GET.get('key')#获取关键词
    if(keyword==""):
        return redirect('主页')#判断关键词是否为空
    blogall=Blog.objects.order_by('-viewnum')
    bloglist=[]
    for blog in blogall:
        if(keyword in blog.title or keyword in blog.bloginfo): #查找符合条件的博客
            bloglist.append(blog)
    return render(请求,'seachblog.html',{'bloglist':bloglist,'keyword':keyword})
#修改博客
def updateblog(请求):
    if (请求.user.is_authenticated == False):
        return redirect('user:登录')
    if (请求.method == 'POST'):
        blogid=请求.POST['blogid']
        blog = Blog.objects.get(pk=blogid, bloguser=请求.user)
        str1 = 请求.POST['id_博客内容-wmd-wrapper-html-code']
        bloginfo = mdtotex(str1)
        blog.title=请求.POST['title']
        blog.bloginfo=bloginfo[0:200]+'......'
        blog.blogtext=请求.POST['博客内容']
        blog.save()
        return redirect('user:用户博客')
    else:
        blogid=请求.GET.get('blogid')
        blog=Blog.objects.get(pk=blogid,bloguser=请求.user)
        form = AddBlogForm(initial={'博客内容':blog.blogtext})
        return render(请求, 'updateblog.html', {'form': form,'blog':blog})
#删除博客
def deleteblog(请求):
    if (请求.user.is_authenticated == False):
        return redirect('user:登录')
    blogid=请求.GET.get('blogid')
    Blog.objects.get(pk=int(blogid),bloguser=请求.user).delete()
    return redirect('user:用户博客')
```

Read blog contents

Users could click on any blogs they take interests in and read the details of those blog.



Function realization method

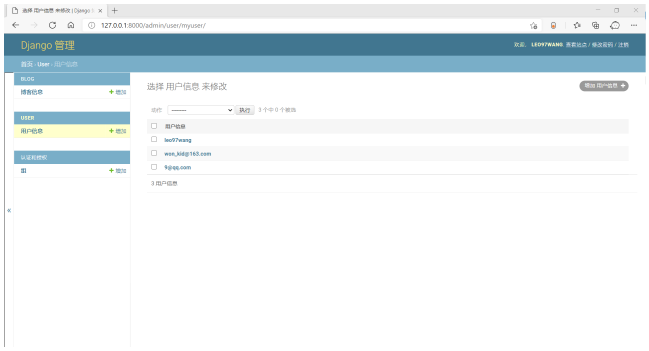
When users choose a blog, controller will send the correspondent blogid to Views layer. Views layer retrieve information from database, form it in markdown format and transfer it to template detailblog.html . Then the web page will be displayed to the clients.

Create superusers and manage all the information

Use command below to create superusers

```
python manage.py createsuperuser
```

and visit url: $http://127.0.0.1:8000/admin/$ to manage all the information



How to install

Create a New Virtual Environment and Activate it:

```
conda create -n DjangoPath python=3.7.3
conda activate DjangoPath
```

Install Project Dependencies:

```
conda install Django  #install Django
pip install django-notifications-hq
pip install django-mdeditor   #blogs are written by markdown language, like csdn
pip install markdown
```

Deploy the System:

```
python manage.py makemigrations  #migrate databases, I use sqlite3 database
python manage.py migrate
python manage.py runserver 127.0.0.1:8000
```

Use the Blog:

Open your favorite browser, visit the website: 127.0.0.1 : 8000 and enjoy the blog system!

Don't forget to register and login for gaining all functions.

Project tree

```
untitled
├─ untitled
│  ├─ asgi.py
│  ├─ settings.py
│  ├─ urls.py
│  ├─ views.py
│  ├─ wsgi.py
│  ├─ __init__.py
│  └─ __pycache__
├─ user
│  ├─ admin.py
│  ├─ apps.py
│  ├─ forms.py
│  ├─ models.py
│  ├─ tests.py
│  ├─ urls.py
│  ├─ views.py
│  ├─ __init__.py
│  ├─ migrations
│  └─ __pycache__
├─ blog
│  ├─ admin.py
│  ├─ apps.py
│  ├─ forms.py
│  ├─ models.py
│  ├─ tests.py
│  ├─ urls.py
│  ├─ views.py
│  ├─ __init__.py
│  ├─ migrations
│  └─ __pycache__
├─ static
│  ├─ bootstrap-3.3.7
│  │  ├─ css
│  │  ├─ fonts
│  │  └─ js
│  ├─ jquery-1.12.4.min.js
│  ├─ jquery.tinymce.min.js
│  └─ langs
├─ templates
│  ├─ detailblog.html
│  ├─ home.html
│  ├─ login.html
│  ├─ seachblog.html
│  ├─ updateblog.html
│  ├─ updatepassword.html
│  ├─ userblog.html
│  ├─ userregister.html
│  └─ writeblog.html
├─ media
├─ db.sqlite3
├─ manage.py
├─ requirements.txt
├─ .idea
├─ .vscode
└─ __pycache__
```

References

1. https://www.tutorialspoint.com/django/django_overview.htm

2. https://blog.csdn.net/qq_40558166/article/details/101106983

3. https://blog.csdn.net/qq_40558166/article/details/102741382

4. https://code.visualstudio.com/api/extension-guides/tree-view

5. https://www.codenong.com/43564567/

6. https://www.cnblogs.com/zydeboke/p/11557875.html