

# 文字筆記網站

一、 專題目標 .....	2
二、 每人貢獻 .....	2
三、 程式功能與設計說明 .....	2
1. 模型設計 .....	2
2. 基礎樣板設計 .....	3
3. 文章功能 .....	4
4. 用戶功能 .....	11
四、 操作手冊 .....	15

## 一、 專題目標

設想為用戶提供記錄和管理文字內容等功能的網站。所有文章內容皆會按照作者陳列在首頁。提供用戶註冊、登入、登出和刪除帳號功能，登入後用戶可以新增文章或針對自己的文章做編輯以及刪除。

## 二、 每人貢獻

D0947639 洪苡慈：文章管理(含前後端)、排版與互動 (CSS)

D0948005 王瀚鍾：用戶管理(含前後端)、權限限制(要怎麼判斷哪些功能用戶可以使用)

## 三、 程式功能與設計說明

我們原先規劃是先做到能讓用戶對文章進行評論，再陸續進行前端 CSS 的部分，因此在實作的過程中我們先設計 models、URL 路徑、view、HTML，但由於時間因素我們捨棄了評論的功能並倉促地添加 CSS，因此接下來的設計與功能的說明會著重在於模型的設計於網站的功能。

### 1. 模型設計

首先模型的部分我們建立了一個文章模型，其中包含作者、標題與內文、創建時間、文章閱覽量，其中我們將作者透過 foreign key 連接 Django 內建的 user 模型，如下圖

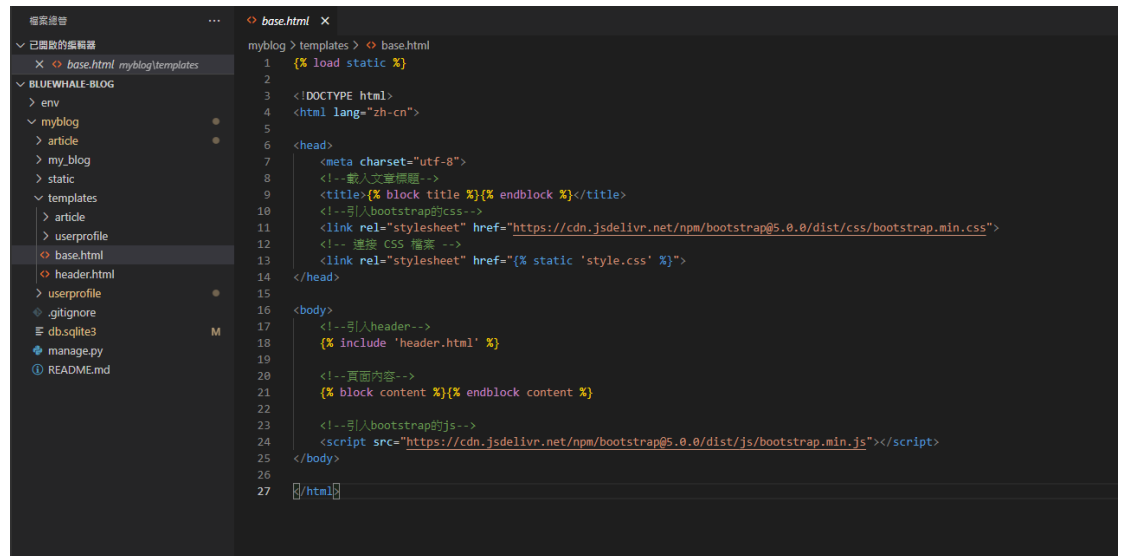
```
article > models.py > ...
1  from django.db import models
2  # Django本身內建的User模型。
3  from django.contrib.auth.models import User
4  # timezone處理時間相關
5  from django.utils import timezone
6  # class Article處理所有文章相關的數據
7  class ArticlePost(models.Model):
8      # 文章作者author透過外鍵models.ForeignKey與內建User模型連結到一起，on_delete指定資料同步刪除
9      author = models.ForeignKey(User, on_delete=models.CASCADE)
10
11      # 文章標題
12      title = models.CharField(max_length=100)
13
14      # 文章內文
15      body = models.TextField()
16
17      # 創建文章的時間
18      created = models.DateTimeField(default=timezone.now)
19
20      total_views = models.PositiveIntegerField(default=0)
21
22      # 定義model的metadata
23      class Meta:
24          # 數據以倒序排列
25          ordering = ('-created',)
26
27      # 返回文章標題
28      def __str__(self):
29          return self.title
```

## 2. 基礎樣板設計

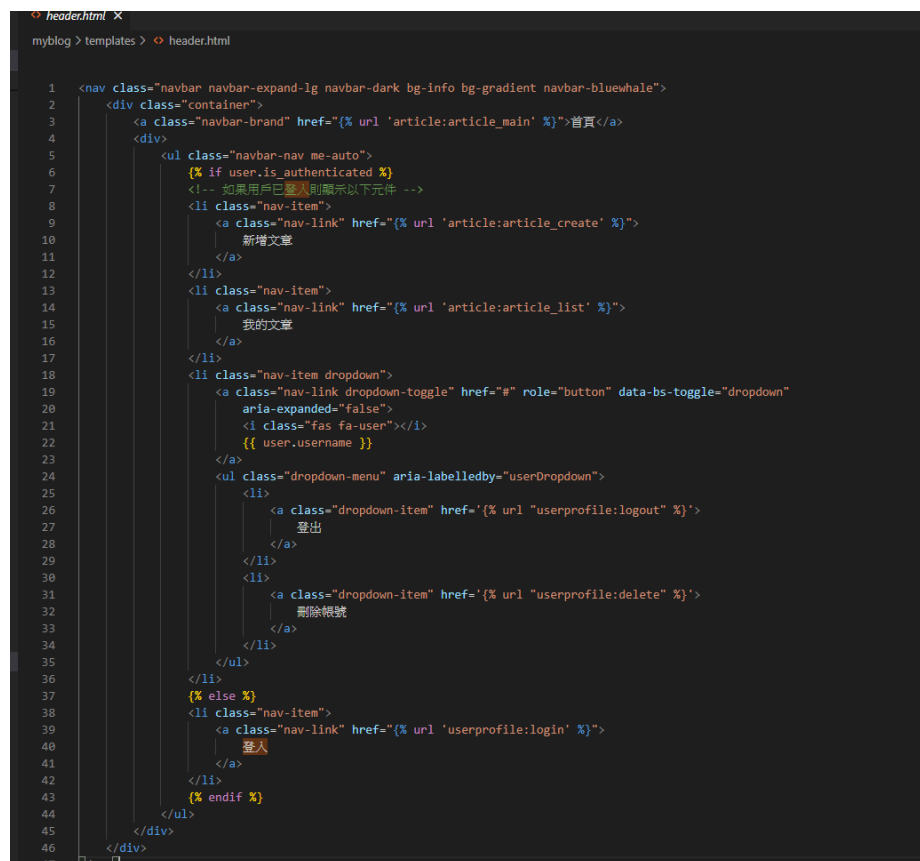
設計一個 base.html 加載所有所需靜態文件：style.css、bootstrap  
header.html 保持在頁面上方，網頁內容可以替換填充，並在登入功能、文章功能設計完成後加入條件顯示的 html 與 label a 的網址跳轉連結。

限定登入者可使用：新增文章、我的文章、登出、刪除帳號

未登入者可以使用：登入、首頁



```
1  {% load static %}
2
3  <!DOCTYPE html>
4  <html lang="zh-cn">
5
6  <head>
7      <meta charset="utf-8">
8      <!-- 登入文章標題 -->
9      <title>{% block title %}{% endblock %}</title>
10     <!-- 引入bootstrap的css -->
11     <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/css/bootstrap.min.css">
12     <!-- 連接 CSS 檔案 -->
13     <link rel="stylesheet" href="{% static 'style.css' %}">
14 </head>
15
16 <body>
17     <!-- 引入header -->
18     {% include 'header.html' %}
19
20     <!-- 頁面內容 -->
21     {% block content %}{% endblock content %}
22
23     <!-- 引入bootstrap的js -->
24     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/js/bootstrap.min.js"></script>
25 </body>
26
27 </html>
```

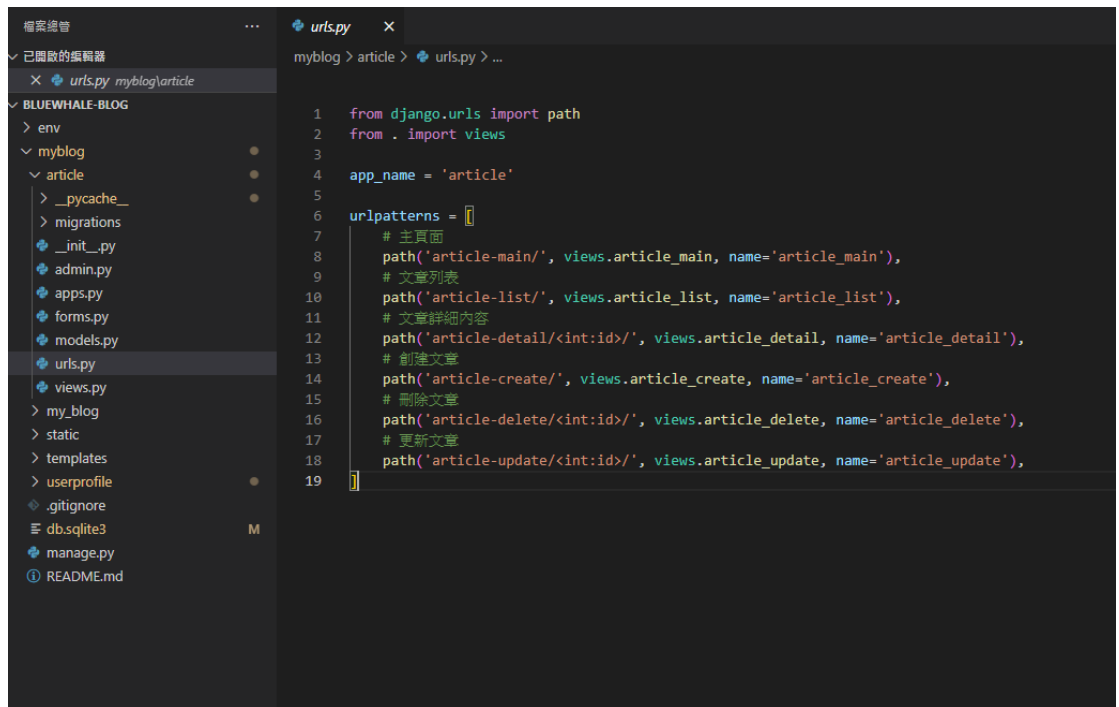


```
1  <nav class="navbar navbar-expand-lg navbar-dark bg-info bg-gradient navbar-bluewhale">
2      <div class="container">
3          <a class="navbar-brand" href="{% url 'article:article_main' %}">首頁</a>
4          <div>
5              <ul class="navbar-nav me-auto">
6                  {% if user.is_authenticated %}
7                  <!-- 如果用戶已登入則顯示以下元件 -->
8                  <li class="nav-item">
9                      <a class="nav-link" href="{% url 'article:article_create' %}">
10                          新增文章
11                      </a>
12                  </li>
13                  <li class="nav-item">
14                      <a class="nav-link" href="{% url 'article:article_list' %}">
15                          我的文章
16                      </a>
17                  </li>
18                  <li class="nav-item dropdown">
19                      <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown"
20                          aria-expanded="false">
21                          <i class="fas fa-user"></i>
22                          {{ user.username }}
23                      </a>
24                      <ul class="dropdown-menu" aria-labelledby="userDropdown">
25                          <li>
26                              <a class="dropdown-item" href="{% url 'userprofile:logout' %}">
27                                  登出
28                              </a>
29                          </li>
30                          <li>
31                              <a class="dropdown-item" href="{% url 'userprofile:delete' %}">
32                                  刪除帳號
33                              </a>
34                          </li>
35                      </ul>
36                  </li>
37                  {% else %}
38                  <li class="nav-item">
39                      <a class="nav-link" href="{% url 'userprofile:login' %}">
40                          登入
41                      </a>
42                  </li>
43                  {% endif %}
44              </ul>
45          </div>
46      </div>
47 </nav>
```

### 3. 文章功能

在登入功能新增後我們利用 Django 內建的 `login_required` 來修飾文章列表、新增、修改、刪除的功能，確保需要在有用戶登入的情況下才能操作這些功能，此外刪除與修改的部分我們需要在近一步判斷作者是否與用戶是同一個人，用來避免有人直接輸入網址(Url)對網頁文章進行操作

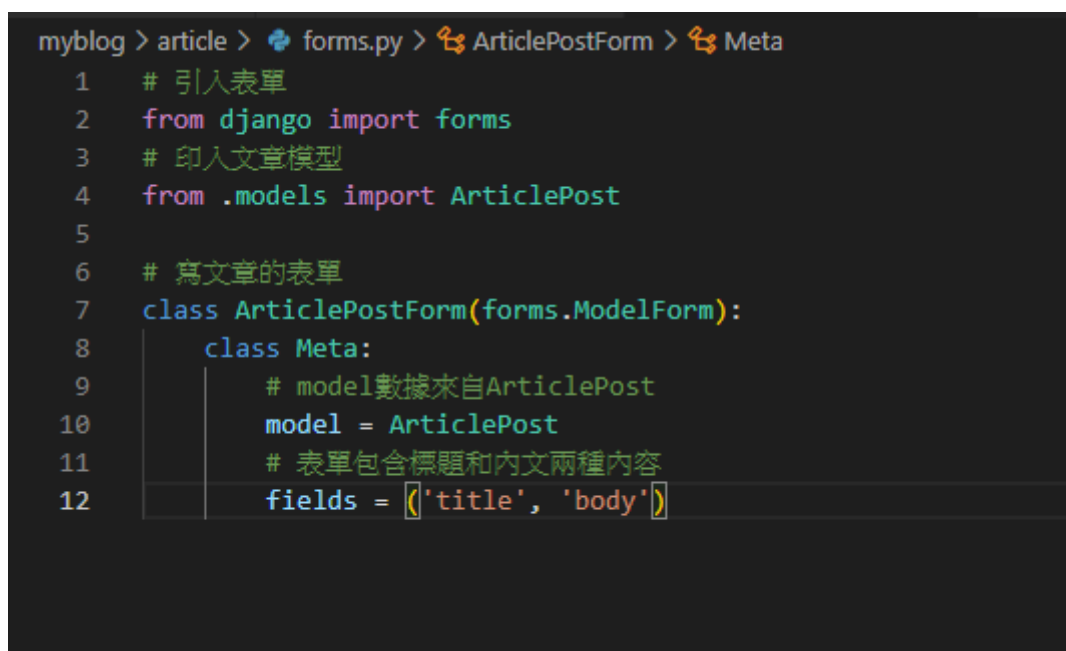
#### Urls



```
1 from django.urls import path
2 from . import views
3
4 app_name = 'article'
5
6 urlpatterns = [
7     # 主頁面
8     path('article-main/', views.article_main, name='article_main'),
9     # 文章列表
10    path('article-list/', views.article_list, name='article_list'),
11    # 文章詳細內容
12    path('article-detail/<int:id>/', views.article_detail, name='article_detail'),
13    # 創建文章
14    path('article-create/', views.article_create, name='article_create'),
15    # 刪除文章
16    path('article-delete/<int:id>/', views.article_delete, name='article_delete'),
17    # 更新文章
18    path('article-update/<int:id>/', views.article_update, name='article_update'),
19 ]
```

#### Forms

導入 ArticlePost 文章 model，指定 ArticlePostForm 表單的 metadata

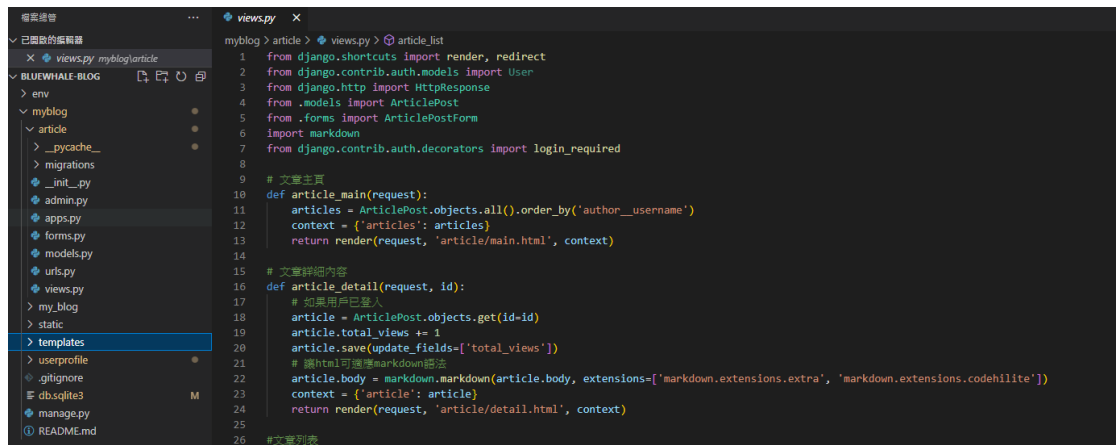


```
myblog > article > forms.py > ArticlePostForm > Meta
1 # 引入表單
2 from django import forms
3 # 引入文章模型
4 from .models import ArticlePost
5
6 # 寫文章的表單
7 class ArticlePostForm(forms.ModelForm):
8     class Meta:
9         # model數據來自ArticlePost
10        model = ArticlePost
11        # 表單包含標題和內文兩種內容
12        fields = ('title', 'body')
```

首頁

myblog\article\views.py

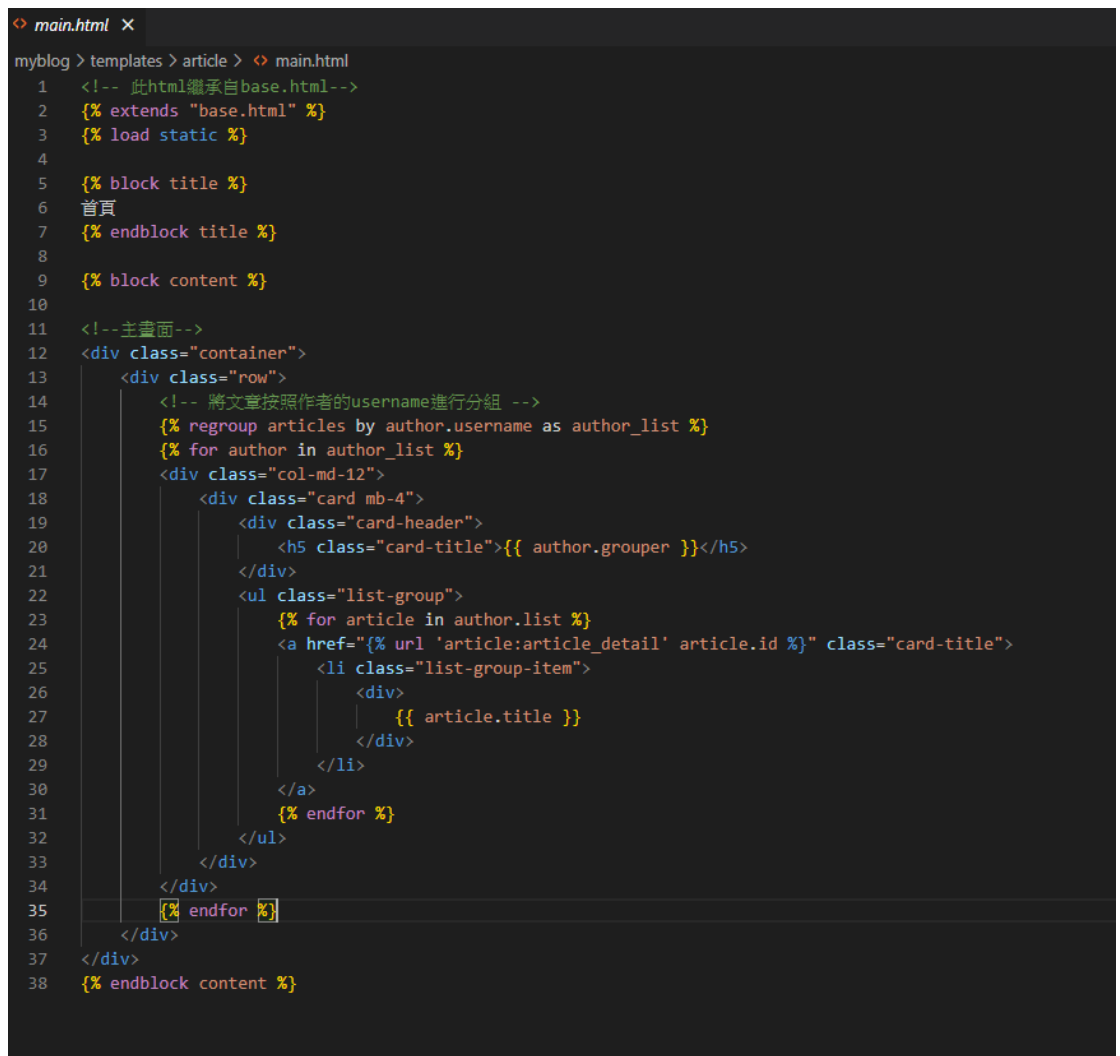
article\_main：將文章按照作者排序，回傳文章 context



```
myblog > article > views.py > article_list
1 from django.shortcuts import render, redirect
2 from django.contrib.auth.models import User
3 from django.http import HttpResponseRedirect
4 from .models import ArticlePost
5 from .forms import ArticlePostForm
6 import markdown
7 from django.contrib.auth.decorators import login_required
8
9 # 文章主頁
10 def article_main(request):
11     articles = ArticlePost.objects.all().order_by('author__username')
12     context = {'articles': articles}
13     return render(request, 'article/main.html', context)
14
15 # 文章詳細內容
16 def article_detail(request, id):
17     # 如果用戶已登入
18     article = ArticlePost.objects.get(id=id)
19     article.total_views += 1
20     article.save(update_fields=['total_views'])
21     # 讓html可支援markdown語法
22     article.body = markdown.markdown(article.body, extensions=['markdown.extensions.extra', 'markdown.extensions.codehilite'])
23     context = {'article': article}
24     return render(request, 'article/detail.html', context)
25
26 # 文章列表
```

myblog\templates\article\main.html

將文章按照作者的 username 分組，同組的放在同一個 card 裡，點選 list 的文章，會跳到該文章的內容畫面



```
myblog > templates > article > main.html
1 <!-- 此html繼承自base.html -->
2 {% extends "base.html" %}
3 {% load static %}
4
5 {% block title %}
6 首頁
7 {% endblock title %}
8
9 {% block content %}
10
11 <!-- 主畫面 -->
12 <div class="container">
13     <div class="row">
14         <!-- 將文章按照作者的username進行分組 -->
15         {% regroup articles by author.username as author_list %}
16         {% for author in author_list %}
17             <div class="col-md-12">
18                 <div class="card mb-4">
19                     <div class="card-header">
20                         <h5 class="card-title">{{ author.grouper }}</h5>
21                     </div>
22                     <ul class="list-group">
23                         {% for article in author.list %}
24                         <a href="{% url 'article:article_detail' article.id %}" class="card-title">
25                             <li class="list-group-item">
26                                 <div>
27                                     {{ article.title }}
28                                 </div>
29                             </li>
30                         </a>
31                         {% endfor %}
32                     </ul>
33                 </div>
34             </div>
35         {% endfor %}
36     </div>
37 </div>
38 {% endblock content %}
```

我的文章

myblog\article\views.py

article\_list：限制為登入者可使用，篩選目前使用者的文章，放入列表後回傳列表內的文章

```
26 #文章列表
27 @login_required(login_url='userprofile:login')
28 def article_list(request):
29     user = request.user
30     articles = ArticlePost.objects.filter(author=user)
31     context = {'articles': articles}
32     return render(request, 'article/list.html', context)
33
34
```

myblog\templates\article\list.html

for 迴圈挑出列表裡所有文章，以 card 顯示文章標題、100 字摘要，下方可以看到文章的瀏覽量和跳轉文章內容的按鈕，以及瀏覽量

```
<> list.html X
myblog > templates > article > <> list.html
1 <!-- 此html繼承自base.html-->
2 {% extends "base.html" %}
3 {% load %}
4
5 {% block title %}
6 首頁
7 {% endblock title %}
8
9 {% block content %}
10
11 <!-- 文章列表 -->
12 <div class="container">
13     <div class="row mt-2">
14
15         {% for article in articles %}
16         <div class="col-4 mb-4">
17             <!-- 卡片容器 -->
18             <div class="card h-100">
19                 <div class="card-body">
20                     <h4 class="card-title">{{ article.title }}</h4>
21                     <p class="card-text">作者： {{ article.author }}</p>
22                 </div>
23                 <!-- 文章內容摘要 -->
24                 <div class="card-body">
25                     <p class="card-text">{{ article.body|slice:'100' }}...</p>
26                 </div>
27                 <div class="card-footer">
28                     <a href="{% url 'article:article_detail' article.id %}" class="btn">閱讀文章</a>
29                     <!-- 顯示瀏覽量 -->
30                     <span>
31                         <small class="col align-self-end" style="color: gray;">
32                             瀏覽： {{ article.total_views }}
33                         </small>
34                     </span>
35                 </div>
36             </div>
37         </div>
38         {% endfor %}
39
40     </div>
41 </div>
42 {% endblock content %}
```

文章內容

myblog\article\views.py

article\_detail：抓取文章 id 增加該文章閱覽數，讓文章內部可適應 markdown (雖然我們在新增文章時都沒特別使用 markdown 語法進行測試)

```
from django.shortcuts import render, redirect
from django.contrib.auth.models import User
from django.http import HttpResponseRedirect
from .models import ArticlePost
from .forms import ArticlePostForm
import markdown
from django.contrib.auth.decorators import login_required

# 文章主頁
def article_main(request):
    articles = ArticlePost.objects.all().order_by('author__username')
    context = {'articles': articles}
    return render(request, 'article/main.html', context)

# 文章詳細內容
def article_detail(request, id):
    # 如果用戶已登入
    article = ArticlePost.objects.get(id=id)
    article.total_views += 1
    article.save(update_fields=['total_views'])
    # 讓html可適應markdown語法
    article.body = markdown.markdown(article.body, extensions=['markdown.extensions.extra', 'markdown.extensions.codehilite'])
    context = {'article': article}
    return render(request, 'article/detail.html', context)
```

myblog\templates\article\detail.html

列出文章標題、文章作者和文章內容，假設是作者本人，則顯示編輯文章和刪除文章連結

```
<> detail.html X
myblog > templates > article > <> detail.html
1 <!-- 此html繼承自base.html-->
2 {% extends "base.html" %}
3 {% load %}
4
5 {% block title %}
6 文章標題
7 {% endblock title %}
8
9 {% block content %}
10 <div class="container">
11     <div class="row">
12         <h1 class="col-12 mt-4 mb-4">{{ article.title }}</h1>
13         <div class="col-12 alert alert-success">
14             作者：{{ article.author }}
15             <br>
16             瀏覽：{{ article.total_views }}
17
18             <!--只開放文章作者自己編輯和刪除文章-->
19             {% if user.is_authenticated and user == article.author %}
20                 <a href="{% url 'article:article_delete' article.id %}">刪除文章</a>
21                 <a href="{% url 'article:article_update' article.id %}">編輯文章</a>
22             {% endif %}
23         </div>
24     </div>
25
26     <!--文章內容-->
27     <div class="col-12">
28         <p>{{ article.body|safe }}</p>
29     </div>
30 </div>
31
32 {% endblock content %}
```

## 編輯文章

myblog\article\views.py

article\_update：獲取文章 id，儲存新標題和新內容並回傳，如果標題和內文不合規定則提醒，沒有改變就將原本內容回傳，如果登入帳戶不是文章作者則無法編輯

```
# 更新文章
@login_required(login_url='userprofile:login')
def article_update(request, id):
    # 獲取文章id
    article = ArticlePost.objects.get(id=id)

    # 過濾非作者的用戶
    if request.user != article.author:
        return HttpResponse("抱歉，你無權修改這篇文章。")

    # 用戶提交數據(POST)
    if request.method == "POST":
        article_post_form = ArticlePostForm(data=request.POST)
        if article_post_form.is_valid():
            # 儲存新的title和body
            article.title = request.POST['title']
            article.body = request.POST['body']
            article.save()
            return redirect("article:article_detail", id=id)
        else:
            return HttpResponse("內容有誤，請重新填寫。")

    # 用戶請求數據(GET)
    else:
        article_post_form = ArticlePostForm()
        # 將原文章內容傳回去
        context = { 'article': article, 'article_post_form': article_post_form }
        return render(request, 'article/update.html', context)
```



myblog/templates/article/update.html

顯示舊文章標題和內文，完成按鈕確認送出

```
update.html x
myblog > templates > article > update.html
1 <!-- 此html繼承自base.html-->
2 {% extends "base.html" %} {% load %}
3 {% block title %} 更新文章 {% endblock title %}
4 {% block content %}
5 <div class="container">
6   <div class="row">
7     <div class="col-12">
8       <br>
9       <form method="post" action=".">
10        <!-- Django中要用POST數據都需要csrf_token-->
11        {% csrf_token %}
12        <div class="form-group">
13          <label for="title">文章標題</label>
14          <!-- value將數據初始設定為舊的文章內容-->
15          <input type="text" class="form-control" id="title" name="title" value="{{ article.title }}">
16        </div>
17        <div class="form-group">
18          <label for="body">文章正文</label>
19          <textarea type="text" class="form-control" id="body" name="body"
20            rows="12">{{ article.body }}</textarea>
21        </div>
22        <button type="submit" class="btn btn-primary">完成</button>
23      </form>
24    </div>
25  </div>
26 </div>
27 {% endblock content %}
```

刪除文章

myblog/article/views.py

article\_delete：獲取文章 id，刪除該 id 的文章，如果登入帳戶不是文章作者則無法刪除

```
61
62 # 刪除文章
63 @login_required(login_url='userprofile:login')
64 def article_delete(request, id):
65     # 根據id刪除對應文章
66     article = ArticlePost.objects.get(id=id)
67
68     # 過濾非作者的用戶
69     if request.user != article.author:
70         return HttpResponse("抱歉，你無權修改這篇文章。")
71
72     article.delete()
73     return redirect("article:article_list")
74
```

新增文章

myblog\article\views.py

article\_create：紀錄填寫內容，標記於目前使用者之下，如果填寫內容不合規則提醒

```
34
35 # 創建文章
36 @login_required(login_url='userprofile:login')
37 def article_create(request):
38     # 用戶提交數據(POST)
39     if request.method == "POST":
40         # 將用戶提交的數據給article_post_form
41         article_post_form = ArticlePostForm(data=request.POST)
42         # 判斷article_post_form內容是否有標題和內文且符合限制
43         if article_post_form.is_valid():
44             # 將數據內容保存
45             new_article = article_post_form.save(commit=False)
46             # 將提交數據的用戶定義為作者
47             new_article.author = request.user
48             # 將數據內容保存到資料庫
49             new_article.save()
50             return redirect("article:article_list")
51         else:
52             return HttpResponse("內容有誤，請重新填寫。")
53     # 用戶請求數據(GET)
54     else:
55         # 創建表單
56         article_post_form = ArticlePostForm()
57         # 給予數據內容
58         context = { 'article_post_form': article_post_form }
59         return render(request, 'article/create.html', context)
```

myblog\templates\article\create.html

card 格式記錄標題和內文，完成按鈕確認送出

```
create.html X
myblog > templates > article > create.html
1 {% extends "base.html" %}
2 {% load %}
3 {% block title %}新增文章{% endblock title %}
4 {% block content %}
5 <div class="container">
6     <div class="row justify-content-center">
7         <div class="col-md-8">
8             <div class="card">
9                 <div class="card-header">新增文章</div>
10                <div class="card-body">
11                    <form method="post" action=".">
12                        {% csrf_token %}
13                        <div class="form-group">
14                            <label for="title">文章標題</label>
15                            <input type="text" class="form-control" id="title" name="title">
16                        </div>
17                        <div class="form-group">
18                            <label for="body">文章內容</label>
19                            <textarea type="text" class="form-control" id="body" name="body" rows="12"></textarea>
20                        </div>
21                        <button type="submit" class="btn">完成</button>
22                    </form>
23                </div>
24            </div>
25        </div>
26    </div>
27 </div>
28 {% endblock content %}
```

#### 4. 用戶功能

##### Urls

```
urls.py  X
myblog > userprofile > urls.py > ...

4  app_name = 'userprofile'
5
6  urlpatterns = [
7      # 登入
8      path('login/', views.user_login, name='login'),
9      # 登出
10     path('logout/', views.user_logout, name='logout'),
11     # 用戶註冊
12     path('register/', views.user_register, name='register'),
13     # 用戶刪除
14     path('delete/', views.user_delete, name='delete'),
15 ]
```

##### Forms

將傳進來的資料進行檢查後再將資料傳給 user\_register

```
from django import forms
from django.contrib.auth.models import User

class UserLoginForm(forms.Form):
    username = forms.CharField()
    password = forms.CharField()
# 註冊用戶表單
class UserRegisterForm(forms.ModelForm):
    # 覆寫 User 的密碼
    password = forms.CharField()
    password2 = forms.CharField()

    class Meta:
        model = User
        fields = ('username', 'email')

# 對兩次輸入的密碼是否一致進行檢查
def clean_password2(self):
    data = self.cleaned_data
    if data.get('password') == data.get('password2'):
        return data.get('password')
    else:
        raise forms.ValidationError("密碼輸入不一致,請重試。")
```

## 登入

myblog\userprofile\views.py

user\_login：當請求 POST 時，使用傳入的數據初始化，檢查輸入的帳號密碼是否符合要求。如果驗證失敗，密碼不匹配，給予錯誤訊息，如果驗證成功則進一步比對帳戶資料有沒有這筆帳號密碼，有的話則用 auth 的 login 將這個用戶保存在該 session 中保持登入狀態，沒有的話則回傳相對應的錯誤訊息。

```
views.py M X login.html urls.py forms.py ...\userprofile forms.py ...\article
myblog > userprofile > views.py > user_login
5 # 引入驗證登入
6 from django.contrib.auth.decorators import login_required
7 from .forms import UserLoginForm, UserRegisterForm
8
9
10 # 登入
11 def user_login(request):
12     if request.method == 'POST':
13         user_login_form = UserLoginForm(data=request.POST)
14         if user_login_form.is_valid():
15             # .cleaned_data 清洗出合法數據
16             data = user_login_form.cleaned_data
17             user = authenticate(username=data['username'], password=data['password'])
18             # 檢查帳號密碼是否在數據庫中有符合項目
19             if user:
20                 # 有對應資料則登入
21                 login(request, user)
22                 return redirect("article:article_list")
23             else:
24                 return HttpResponse("帳號或密碼有錯，請重新輸入")
25         else:
26             return HttpResponse("帳號或密碼不合規")
27     elif request.method == 'GET':
28         user_login_form = UserLoginForm()
29         context = { 'form': user_login_form }
30         return render(request, 'userprofile/login.html', context)
31     else:
32         return HttpResponse("請使用GET或POST")
33
```

myblog\templates\userprofile\login.html

帳號密碼欄必填，點選連結跳到註冊頁面

```
login.html x
myblog > templates > userprofile > login.html
1  {% extends "base.html" %}
2  {% load static %}
3  {% block title %}登入{% endblock title %}
4  {% block content %}
5
6  <div class="container">
7      <div class="row justify-content-center">
8          <div class="col-md-6">
9              <div class="card">
10                 <div class="card-header">登入</div>
11                 <div class="card-body">
12                     <form method="post" action=".">
13                         {% csrf_token %}
14                         <div class="form-group">
15                             <label for="username">帳號</label>
16                             <input type="text" class="form-control" id="username" name="username" required>
17                             <div class="invalid-feedback">請輸入帳號</div>
18                         </div>
19                         <div class="form-group">
20                             <label for="password">密碼</label>
21                             <input type="password" class="form-control" id="password" name="password" required>
22                             <div class="invalid-feedback">請輸入密碼</div>
23                         </div>
24                         <button type="submit" class="btn">確定</button>
25                         <h5>還沒有帳號? 點擊<a href='{% url "userprofile:register" %}'> 註冊帳號 </a></h5>
26                     </form>
27                 </div>
28             </div>
29         </div>
30     </div>
31 </div>
32 {% endblock content %}
```

註冊

myblog\userprofile\views.py

user\_register：資料通過驗證的話就註冊一個新用戶並登入

```
38
39 # 用戶註冊
40 def user_register(request):
41     if request.method == 'POST':
42         user_register_form = UserRegisterForm(data=request.POST)
43         if user_register_form.is_valid():
44             new_user = user_register_form.save(commit=False)
45             # 設置密碼
46             new_user.set_password(user_register_form.cleaned_data['password'])
47             new_user.save()
48             # 保存好數據後立即登入並返回博客列表頁面
49             login(request, new_user)
50             return redirect("article:article_list")
51         else:
52             return HttpResponse("註冊表單輸入有誤。請重新輸入~")
53     elif request.method == 'GET':
54         user_register_form = UserRegisterForm()
55         context = { 'form': user_register_form }
56         return render(request, 'userprofile/register.html', context)
57     else:
58         return HttpResponse("請使用GET或POST請求數據")
59
```

myblog\templates\userprofile\register.html

透過 input 將帳號、密碼等資料傳遞給 UserRegisterForm

```
register.html x
myblog > templates > userprofile > register.html
1  {% extends "base.html" %} {% load %}
2  {% block title %} 登入 {% endblock title %}
3  {% block content %}
4  <div class="container">
5      <div class="row">
6          <div class="register col-md-6">
7              <div class="form-container">
8                  <form method="post" action=".">
9                      {% csrf_token %}
10                     <!-- 帳號 -->
11                     <div class="form-group col-md-4">
12                         <label for="username">帳號</label>
13                         <input type="text" class="form-control" id="username" name="username" required>
14                     </div>
15                     <!-- 信箱 -->
16                     <div class="form-group col-md-4">
17                         <label for="email">Email</label>
18                         <input type="text" class="form-control" id="email" name="email">
19                     </div>
20                     <!-- 密碼 -->
21                     <div class="form-group col-md-4">
22                         <label for="password">設置密碼</label>
23                         <input type="password" class="form-control" id="password" name="password" required>
24                     </div>
25                     <!-- 確認密碼 -->
26                     <div class="form-group col-md-4">
27                         <label for="password2">確認密碼</label>
28                         <input type="password" class="form-control" id="password2" name="password2" required>
29                     </div>
30                     <br>
31                     <!-- 提交按鈕 -->
32                     <button type="submit" class="btn btn-primary">提交</button>
33                 </form>
34             </div>
35         </div>
36         <div class="col-md-6">
37             
39         </div>
40     </div>
41     {% endblock content %}
```

登出

myblog\userprofile\views.py

user\_logout：用戶登出

```
# 登出
def user_logout(request):
    logout(request)
    return redirect("article:article_list")
```

刪除用戶

myblog\userprofile\views.py

在測試的時後，

user\_delete：確認目前帳戶是要確認的帳戶，登出後，刪除該用戶一切資料

```
def user_delete(request):  
    user = request.user  
    # 驗證登入用戶、待刪除用戶是否相同  
    if request.user == user:  
        #退出登入，刪除數據並返回blog列表  
        logout(request)  
        user.delete()  
        return redirect("article:article_list")  
    else:  
        return HttpResponse("你沒有刪除操作的權限。")
```

#### 5. 互動功能

myblog\static\style.css

當鼠標懸停在卡片上時卡片放大

輸入時背景顏色改變

鼠標停在 list 上會改變背景

鼠標停在按鈕上時按鈕顏色變深

註冊時，輸入時輸入寬會拉長

編輯或刪除文章時，鼠標指向的連結會變粉紅色

### 四、 操作手冊

環境

py -m pip install Django

py -m pip install markdown

運行

python manage.py runserver

管理員帳密

User

testuser123

bluewhale

bw00manager

普通帳密

test12

test12

管理員介面

<http://127.0.0.1:8000/admin/>

首頁

<http://127.0.0.1:8000/article/article-main/>