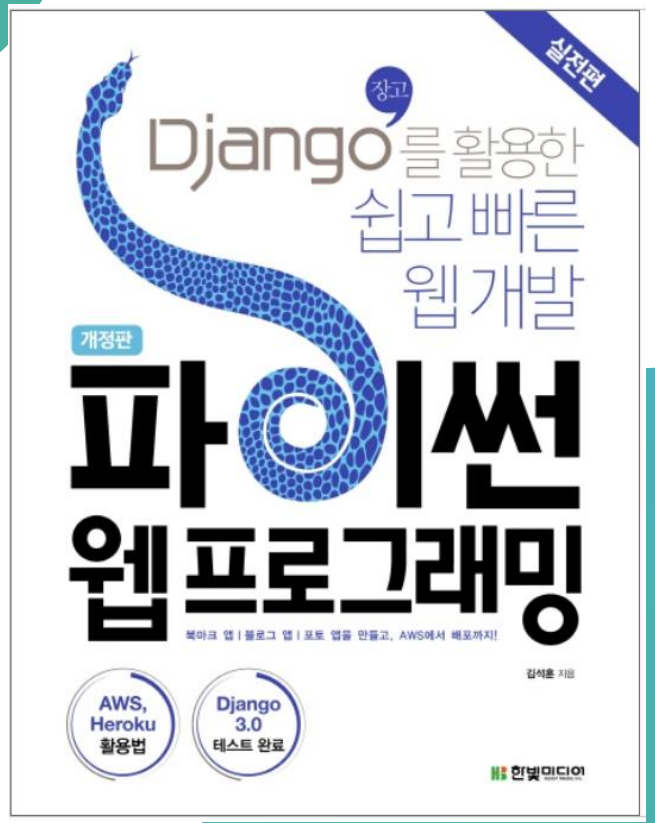
CHAPTER 08 Blog 앱 확장 - 검색기능

파이썬 웹프로그래밍

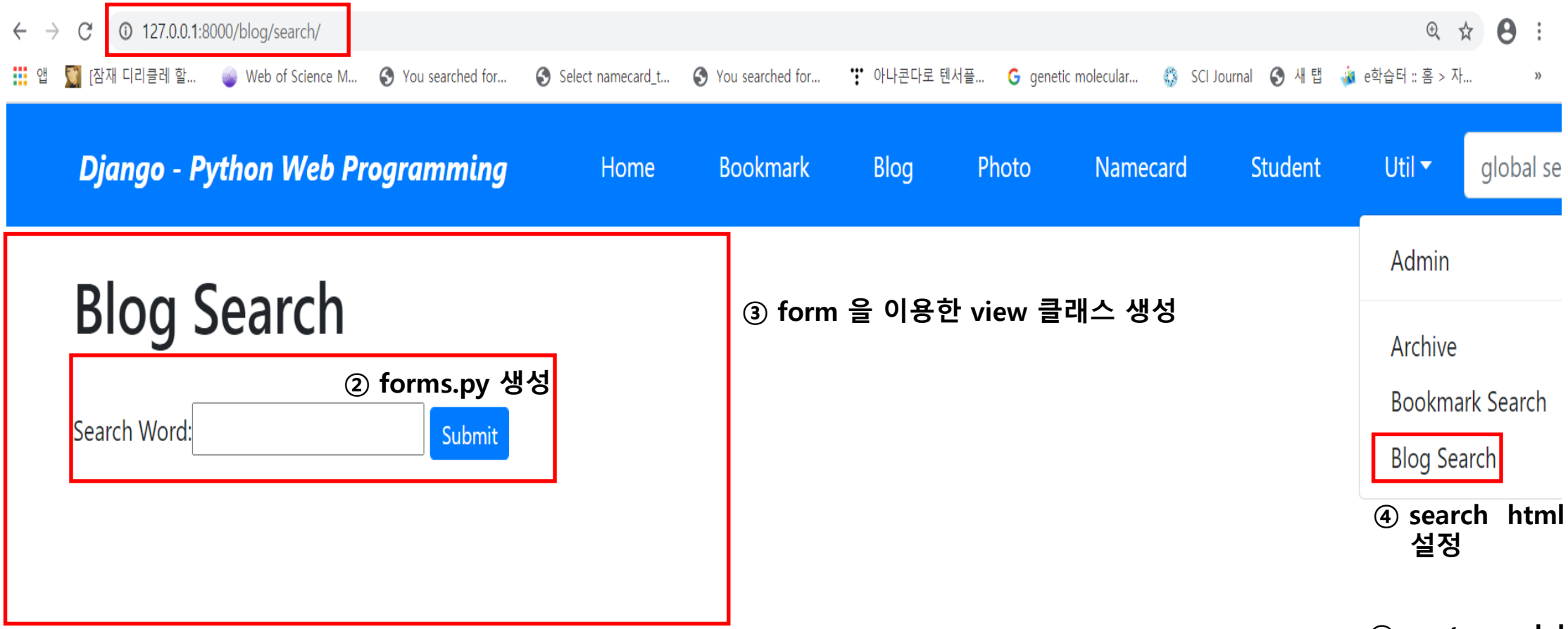


가천대학교 컴퓨터공학과
왕보현

CHAPTER 08 Blog 앱 확장 – 검색 기능

Search

① URL 추가



② forms.py 생성

③ form 을 이용한 view 클래스 생성

④ search html 링크 설정

⑤ post_search.html 생성

1. URLconf

ch99₩blog₩urls.py

```python
from django.urls import path, re_path
from blog import views

app_name = 'blog'
urlpatterns = [

 # Example: /blog/
 path('', views.PostLV.as_view(), name='index'),

 # Example: /blog/post/ (same as /blog/)
 path('post/', views.PostLV.as_view(), name='post_list'),

 # Example: /blog/post/django-example/
 re_path(r'^post/(?P<slug>[-\w]+)/$', views.PostDV.as_view(), name='post_detail'),

 # Example: /blog/archive/
 path('archive/', views.PostAV.as_view(), name='post_archive'),

 # Example: /blog/archive/2019/
 path('archive/<int:year>/', views.PostYAV.as_view(), name='post_year_archive'),

 # Example: /blog/archive/2019/nov/
 path('archive/<int:year>/<str:month>/', views.PostMAV.as_view(), name='post_month_archive'),

 # Example: /blog/archive/2019/nov/10/
 path('archive/<int:year>/<str:month>/<int:day>/', views.PostDAV.as_view(), name='post_day_archive'),

 # Example: /blog/archive/today/
 path('archive/today/', views.PostTAV.as_view(), name='post_today_archive'),

 # Example: /blog/search/
 path('search/', views.SearchFormView.as_view(), name='search'),

]
```

추가

① URL /blog/search/ 요청을 처리할 뷰 클래스를 SearchFormView로 지정
URL 패턴의 이름은 이름공간 포함해서 blog:search가 됨
SearchFormView 클래스형 뷰는 폼을 보여주고 폼에 들어있는 데이터를 처리하기 위한 뷰

2. blog₩forms.py 코딩하기

ch99₩blog₩forms.py

```
from django import forms                                    ①


class PostSearchForm(forms.Form):                           ②
    search_word = forms.CharField(label='Search Word')      ③
```

① 장고는 폼을 클래스로 표현할 수 있도록 하는 기능을 django.forms 모듈에서 제공함. 이 모듈을 import

② 폼을 정의하기 위해서는 django.forms 모듈의 Form 클래스를 상속받아 클래스를 정의해야 함

③ 폼을 정의하는 방법은 테이블의 모델 클래스를 정의하는 방법과 유사
CharField 필드: TextInput 위젯으로 표현
label 인자인 Search Word : 폼 위젯 앞에 출력되는 레이블 변수
변수 search_word : input 태그에 대한 name 속성이 되어 사용자가 입력한 값을 저장하는 데 사용됨.
결국 HTML 요소 <input type="text"> 하나를 만든 것

3. view 파일 코딩하기

ch99₩blog₩views.py 1 - 3장에 추가

```
from django.views.generic import ListView, DetailView, TemplateView
from django.views.generic import ArchiveIndexView, YearArchiveView, MonthArchiveView
from django.views.generic import DayArchiveView, TodayArchiveView
from blog.models import Post
from django.views.generic import FormView                    ①
from blog.forms import PostSearchForm                        ②
from django.db.models import Q                               ③
from django.shortcuts import render                          ④
```

① FormView 클래스형 제네릭 뷰를 임포트

② 검색 폼으로 사용할 PostSearchForm 폼 클래스를 임포트. PostSearchForm 클래스는 앞에서 forms.py 파일에 정의

③ 검색 기능에 필요한 Q 클래스를 임포트 함.

④ 단축 함수 render()를 임포트

3. view 파일 코딩하기

ch99₩blog₩views.py 2 - 3장과 같은 내용

```
#--- ListView
class PostLV(ListView):
    model = Post
    template_name = 'blog/post_all.html'
    context_object_name = 'posts'
    paginate_by = 2


#--- DetailView
class PostDV(DetailView):
    model = Post
```

3. view 파일 코딩하기

ch99₩blog₩views.py 3- 3장과 같은 내용

```
#--- ArchiveView
class PostAV(ArchiveIndexView):
    model = Post
    date_field = 'modify_dt'


class PostYAV(YearArchiveView):
    model = Post
    date_field = 'modify_dt'
    make_object_list = True


class PostMAV(MonthArchiveView):
    model = Post
    date_field = 'modify_dt'


class PostDAV(DayArchiveView):
    model = Post
    date_field = 'modify_dt'


class PostTAV(TodayArchiveView):
    model = Post
    date_field = 'modify_dt'
```

3. view 파일 코딩하기

ch99₩blog₩views.py 4 - 3장에 추가

```
#--- FormView
class SearchFormView(FormView):
    form_class = PostSearchForm
    template_name = 'blog/post_search.html'

    def form_valid(self, form):
        searchWord = form.cleaned_data['search_word']
        post_list = Post.objects.filter(Q(title__icontains=searchWord) |  Q(description__icontains=searchWord) | Q(content__icontains=searchWord)).distinct()

        context = {}
        context['form'] = form
        context['search_term'] = searchWord
        context['object_list'] = post_list

        return render(self.request, self.template_name, context)   # No Redirection
```

3. view 파일 코딩하기

ch99₩blog₩views.py 4

```
#--- FormView
class SearchFormView(FormView):                    ⑤
    form_class = PostSearchForm                    ⑥
    template_name = 'blog/post_search.html'        ⑦
```

⑤ FormView 클래스형 제네릭 뷰를 상속받아 SearchFormView 클래스형 뷰를 정의

FormView 제네릭 뷰 : GET 요청인 경우 폼을 화면에 보여주고 사용자의 입력을 기다림

사용자가 폼에 데이터를 입력한 후 제출하면 이는 POST 요청으로 접수되며 FormView 클래스는 데이터에 대한 유효성 검사를 함

데이터가 유효하면 form_valid() 함수를 실행한 후에 적절한 URL로 리다이렉트 시키는 기능을 갖고 있음

⑥ 폼으로 사용할 클래스를 PostSearchForm으로 지정함.

⑦ SearchFormView에서 호출할 html 문서 지정

3. view 파일 코딩하기

ch99₩blog₩views.py 4

cleaned_data={'search_word';' key value '}

```
def form_valid(self, form):                                    ⑧
    searchWord = form.cleaned_data['search_word']              ⑨
    post_list = Post.objects.filter(Q(title__icontains=searchWord) |  Q(description__icontains=searchWord) | Q(content__icontains=searchWord)).distinct() ⑩
```

⑧ POST 요청으로 들어온 데이터의 유효성 검사를 실시해 에러가 없으면 form_valid() 메소드를 실행

⑨ 유효성 검사를 통과하면, 사용자가 입력한 데이터들은 cleaned_data 사전에 존재함.
이 사전에서 search_word 값을 추출해 searchWord 변수에 지정함. search_word 키는 PostSearchForm 클래스에서 정의한 필드명

⑩ Q 객체는 filter()메소드의 매칭 조건을 다양하게 줄 수 있도록 함. 예제에서는 3 개의 조건을 OR 문장으로 연결하고 있음.
각 조건의 icontains 연산자 : 대소문자를 구분하지 않고 단어가 포함되어 있는지 검사함.
distinct() 메소드 : 중복된 객체를 제외
이 줄의 의미는 Post 테이블의 모든 레코드에 대해 title, description, content 컬럼에 searchWord가 포함된 레코드를 대소문자 구분 없이 검색하고 서로 다른 레코드들만 리스트로 만들어서 post_list 변수에 저장하라는 의미

3. view 파일 코딩하기

ch99₩blog₩views.py 4

```
context = {}                                                    ⑪
context['form'] = form                                          ⑫
context['search_term'] = searchWord                             ⑬
context['object_list'] = post_list                              ⑭

return render(self.request, self.template_name, context)  # No Redirection    ⑮
```

⑪ 템플릿에 넘겨줄 컨텍스트 변수 context를 사전 형식으로 정의함.

⑫ form 객체, 즉 PostSearchForm 객체를 컨텍스트 변수 form에 지정함.

⑬ 검색용 단어 searchWord를 컨텍스트 변수 search_term에 지정함.

⑭ 검색 결과 리스트인 post_list를 컨텍스트 변수 object_list에 지정함.

⑮ 단축 함수 render() : 템플릿 파일과 컨텍스트 변수를 처리해, 최종적으로 HttpResponse 객체를 반환.

4. Template 파일 코딩하기

ch99₩templates₩base.html - 5장의 base.html을 수정

```
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav mr-auto">
            <li class="nav-item mx-1 btn btn-primary">
                <a class="nav-link text-white" href="{% url 'home' %}">Home</a></li>
            <li class="nav-item mx-1 btn btn-primary">
                <a class="nav-link text-white" href="{% url 'bookmark:index' %}">Bookmark</a></li>
            <li class="nav-item mx-1 btn btn-primary">
                <a class="nav-link text-white" href="{% url 'blog:index' %}">Blog</a></li>
            <li class="nav-item mx-1 btn btn-primary">
                <a class="nav-link text-white" href="">Photo</a></li>

            <li class="nav-item dropdown mx-1 btn btn-primary">
                <a class="nav-link dropdown-toggle text-white" href="#" data-toggle="dropdown">Util</a>
                <div class="dropdown-menu">
                    <a class="dropdown-item" href="{% url 'admin:index' %}">Admin</a>
                    <div class="dropdown-divider"></div>
                    <a class="dropdown-item" href="{% url 'blog:post_archive' %}">Archive</a>
                    <a class="dropdown-item" href="{% url 'blog:search' %}">Search</a>   #수정
                </div>
            </li>
        </ul>

        <form class="form-inline my-2" action="" method="post"> {% csrf_token %}
            <input class="form-control mr-sm-2" type="search" placeholder="global search" name="search_word">
        </form>
    </div>
</nav>
```

4. Template 파일 코딩하기

ch99₩blog₩templates₩post_search.html 1

➔ 8장에서 새로 작성

```
{% extends "base.html" %}
{% block title %}post_search.html{% endblock %}
{% block content %}
    <h1>Blog Search</h1>
    <br>
    <form action="." method="post"> {% csrf_token %}
        {{ form.as_table }}
        <input type="submit" value="Submit" class="btn btn-primary btn-sm">
    </form>
    <br/><br/>
    {% if object_list %}
        {% for post in object_list %}
        <h2><a href='{{ post.get_absolute_url }}'>{{ post.title }}</a></h2>
        {{ post.modify_dt|date:"N d, Y" }}
        <p>{{ post.description }}</p>
        {% endfor %}
    {% elif search_term %}
    <b><i>Search Word({{ search_term }}) Not Found !</i></b>
    {% endif %}
{% endblock %}
```

4. Template 파일 코딩하기

ch99₩blog₩templates₩post_search.html 2 ➔ 8장에서 새로 작성

```
① <form action="." method="post"> {% csrf_token %} ②
    {{ form.as_table }}  ③
    <input type="submit" value="Submit" class="btn btn-primary btn-sm"> ④
</form>
```

① 검색 폼을 출력함. Submit 버튼을 누르면 POST 방식으로 현재와 동일한 URL로 요청이 전송됨

② CSRF 공격을 방지하기 위해 {% csfr_token%} 태그를 사용함

③ form 을 테이블 형식으로 표시. form은 뷰에서 넘겨준 PostSearchForm 객체임.

```
<tr>
  <th><label for="id_search_word">Search Word:</label></th>
  <td><input type="text" name="search_word" required id="id_search_word"></td>
</tr>
```

④ Submit 이라는 단어로 제출 버튼을 만듦

※ CSRF(Cross Site Request Forgery) 공격이란?
웹 어플리케이션 취약점 중 하나로 인터넷 사용자(희생자)가 자신의 의지와는 무관하게 공격자가 의도한 행위(수정, 삭제, 등록 등)를 특정 웹사이트에 요청하게 만드는 공격

4. Template 파일 코딩하기

ch99₩blog₩templates₩post_search.html 3 ➔ 8장에서 새로 작성

```
{% if object_list %}          ⑤
    {% for post in object_list %}   ⑥
    <h2><a href='{{ post.get_absolute_url }}'>{{ post.title }}</a></h2>  ⑦
    {{ post.modify_dt|date:"N d, Y" }}  ⑧
    <p>{{ post.description }}</p>       ⑨
    {% endfor %}
{% elif search_term %}   ⑩
<b><i>Search Word({{ search_term }}) Not Found !</i></b>   ⑪
{% endif %}
```

⑤ object_list 에 내용이 있는지 확인. 즉 검색 결과가 1개 이상 있는지 확인.

⑥ 검색 결과가 있다면 검색 결과를 순회하며 Post 객체의 title, modify_date, description 속성을 출력

⑦ 객체의 title속성을 <h2> 크기로 출력. title 텍스트에 URL 링크를연결. URL 링크는 객체의 get_absolute_url() 메소드를 호출하여 구함. /blog/post/slug단어/와 같은 형식이 될 것임

⑧ 다음줄에 modify_dt 속성을 출력하는데 July 05, 2015와 같은 형식으로 출력

⑨ 객체(레코드)의 description 속성(컬럼)을 출력

⑩ 검색 결과가 없는 경우 search_tem에 값이 있는지 확인. 사용자가 검색 단어를 입력했는지 여부를 판단하기 위함. 즉 사용자가 검색 단어를 입력하고 검색 결과가 없을 경우 다음줄의 Not Found 문장을 표시

5. 정리

ch99₩blog₩forms.py

```
class PostSearchForm(forms.Form):
    search_word = forms.CharField(label='Search Word')
```

ch99₩blog₩views.py

```
#--- FormView
class SearchFormView(FormView):
    form_class = PostSearchForm
    template_name = 'blog/post_search.html'

    def form_valid(self, form):
        searchWord = form.cleaned_data['search_word']
        post_list = Post.objects.filter(Q(title__icontains=searchWord) |
Q(description__icontains=searchWord) |
Q(content__icontains=searchWord)).distinct()

        context = {}
        context['form'] = form
        context['search_term'] = searchWord
        context['object_list'] = post_list

        return render(self.request, self.template_name, context)  # No
Redirection
```

ch99₩blog₩templates₩blog₩post_sea

```
<form action="." method="post"> {% csrf_token %}
    {{ form.as_table }}
    <input type="submit" value="Submit" class="btn btn-primary btn-sm">
</form>
```

5. 정리

ch99₩blog₩views.py

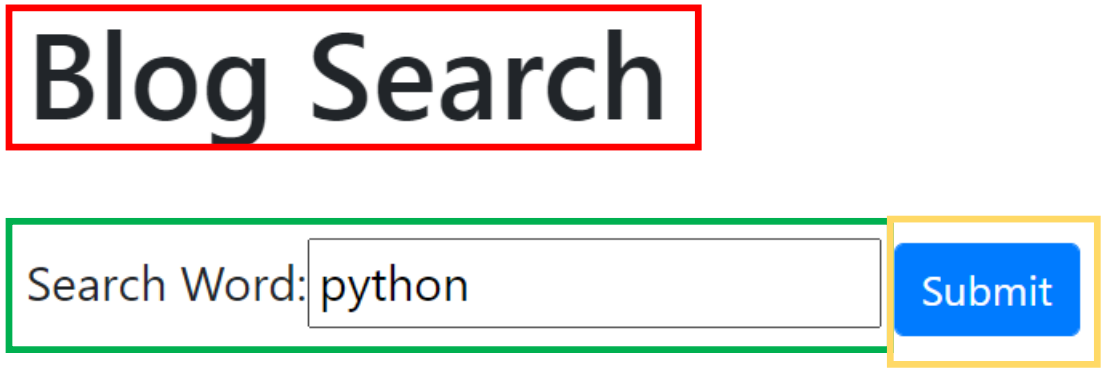
```
#--- FormView
class SearchFormView(FormView):
    form_class = PostSearchForm
    template_name = 'blog/post_search.html'

    def form_valid(self, form):
        searchWord = form.cleaned_data['search_word']
        post_list = Post.objects.filter(Q(title__icontains=searchWord) |
Q(description__icontains=searchWord) |
Q(content__icontains=searchWord)).distinct()

        context = {}
        context['form'] = form
        context['search_term'] = searchWord
        context['object_list'] = post_list

        return render(self.request, self.template_name, context)  # No
Redirection
```

ch99₩blog₩templates₩blog₩post_searc

```
{% if object_list %}
    {% for post in object_list %}
    <h2><a href='{{ post.get_absolute_url }}'>{{ post.title }}</a></h2>
    {{ post.modify_dt|date:"N d, Y" }}
    <p>{{ post.description }}</p>
    {% endfor %}
{% elif search_term %}
<b><i>Search Word({{ search_term }}) Not Found !</i></b>
{% endif %}
```

5. 정리

```
{% extends "base.html" %}
{% block title %}post_search.html{% endblock %}
{% block content %}
  <h1>Blog Search</h1>
  <br>
  <form action="." method="post"> {% csrf_token %}
    {{ form.as_table }}
    <input type="submit" value="Submit" class="btn btn-primary btn-sm">
  </form>
  <br/><br/>
  {% if object_list %}
    {% for post in object_list %}
    <h2><a href='{{ post.get_absolute_url }}'>{{ post.title }}</a></h2>
    {{ post.modify_dt|date:"N d, Y" }}
    <p>{{ post.description }}</p>
    {% endfor %}
  {% elif search_term %}
  <b><i>Search Word({{ search_term }}) Not Found !</i></b>
  {% endif %}
{% endblock %}
```



```
{% extends "base.html" %}
{% block title %}post_search.html{% endblock %}
{% block content %}
    <h1>Blog Search</h1>
    <br>
    <form action="." method="post"> {% csrf_token %}
        {{ form.as_table }}
        <input type="submit" value="Submit" class="btn btn-primary btn-sm">
    </form>
    <br/><br/>
    {% if object_list %}
        {% for post in object_list %}
        <h2><a href='{{ post.get_absolute_url }}'>{{ post.title }}</a></h2>
        {{ post.modify_dt|date:"N d, Y" }}
        <p>{{ post.description }}</p>
        {% endfor %}
    {% elif search_term %}
    <b><i>Search Word({{ search_term }}) Not Found !</i></b>
    {% endif %}
{% endblock %}
```