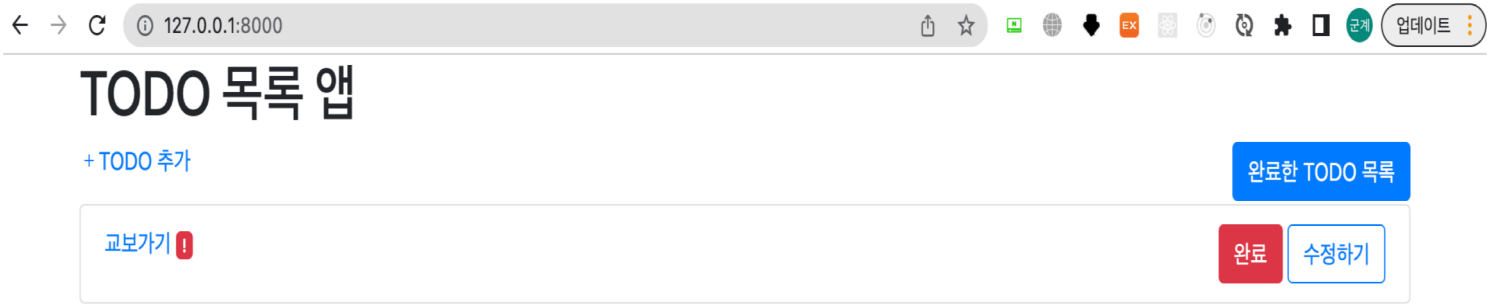
# ToDo_WebService

# 프로젝트 개요

- 프로젝트
  - ✓ 개요: ToDo 의 CRUD
  - ✓ 개발 환경
    - Programming Language
      - Back End: Python3
      - Front End: HTML, CSS, Javascript
    - Back End Framework: Django
    - Database: MySQL
    - IDE: VS-Code

# 프로젝트 개요

❖ 프로젝트



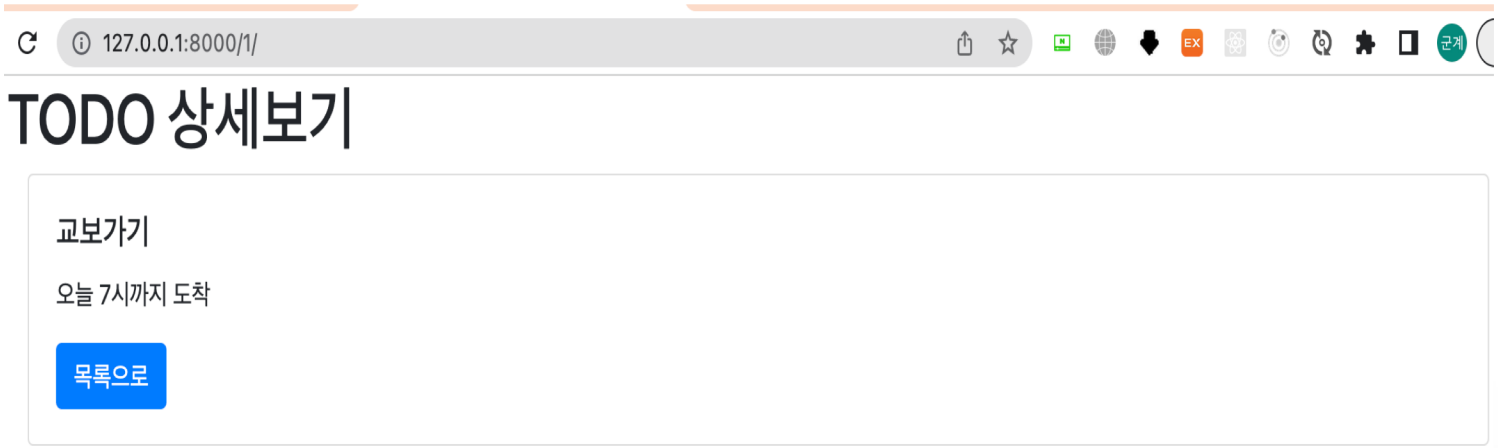
# 프로젝트 개요

❖ 프로젝트


127.0.0.1:8000/post/
TODO 추가하기
Title:
교보가기
오늘 7시까지 도착
Description:
Important:
등록

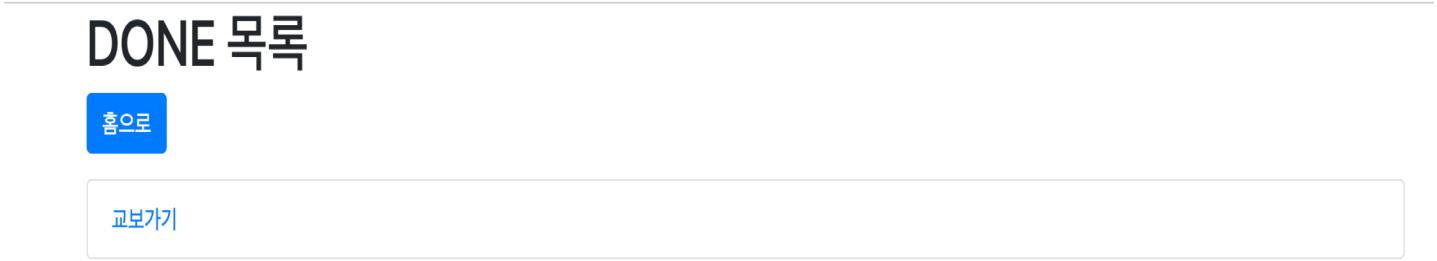
# 프로젝트 개요

❖ 프로젝트



# 프로젝트 개요

❖ 프로젝트



# 프로젝트 설정

- 프로젝트 생성
  - ✓ 프로젝트 디렉토리 생성: TODO
  - ✓ 가상 환경 생성

    python3 -m venv myvenv

    source myvenv/bin/activate
  - ✓ 필요한 패키지 설치
    - pip install django
    - pip install mysqlclient
  - ✓ 프로젝트 생성: django-admin startproject mytodo .
  - ✓ 애플리케이션 생성: python manage.py startapp todo

# 프로젝트 설정

- 프로젝트 생성
  - ✓ mytodo/settings.py 파일을 수정

```
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['127.0.0.1']


# Application definition

INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "todo",
]
```

# 프로젝트 설정

- 프로젝트 생성
  - ✓ mytodo/settings.py 파일을 수정

```
# Database
# https://docs.djangoproject.com/en/4.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'adam',
        'USER': 'root',
        'PASSWORD': 'wnddkd', # mariaDB 설치 시 입력한 root 비밀번호 입력
        'HOST': '127.0.0.1',
        'PORT': ''
    }
}
```

# 프로젝트 설정

- 프로젝트 생성
  - ✓ mytodo/settings.py 파일을 수정

```
# Internationalization
# https://docs.djangoproject.com/en/4.1/topics/i18n/

LANGUAGE_CODE = "en-us"

TIME_ZONE = "Asia/Seoul"

USE_I18N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.1/howto/static-files/

STATIC_URL = "static/"

# Default primary key field type
# https://docs.djangoproject.com/en/4.1/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = "django.db.models.BigAutoField"
```

# 프로젝트 설정

- 프로젝트 생성
  - ✓ MySQL에 접속해서 사용할 데이터베이스 생성 - adam
  - ✓ todo 디렉토리 안에 templates 디렉토리를 생성하고 그 안에 todo 디렉토리를 생성
  - ✓ todo 디렉토리 안에 forms.py, urls.py 파일을 생성

# 프로젝트 설정

- ToDo 모델 생성
  - ✓ 데이터 구조
    - title: 제목
    - description: 설명
    - created: 생성 날짜
    - complete: 완료 여부
    - important: 중요 여부

# 프로젝트 설정

- ToDo 모델 생성
  - ✓ todo 디렉토리의 models.py 수정

```
from django.db import models

class Todo(models.Model):
    title = models.CharField(max_length=100)
    description = models.TextField(blank=True)
    created = models.DateTimeField(auto_now_add=True)
    complete = models.BooleanField(default=False)
    important = models.BooleanField(default=False)

    def __str__(self):
        return self.title
```

# 프로젝트 설정

- ToDo 모델 생성
  - ✓ 모델 생성 명령 수행
    - python manage.py makemigrations
    - python manage.py migrate
  - ✓ 관리자 계정 생성
    - python manage.py createsuperuser

# 프로젝트 설정

- ToDo 모델 생성
  - ✓ 데이터베이스에서 테이블 확인
    - show tables;

| | Tables_in_adam |
|---|---|
| 1 | auth_group |
| 2 | auth_group_permissions |
| 3 | auth_permission |
| 4 | auth_user |
| 5 | auth_user_groups |
| 6 | auth_user_user_permissions |
| 7 | django_admin_log |
| 8 | django_content_type |
| 9 | django_migrations |
| 10 | django_session |
| 11 | mydjango_item |
| 12 | todo_todo |
| 13 | usertbl |

# 프로젝트 설정

- 관리자 기능 설정
  - ✓ 관리자 페이지에서 모델을 사용할 수 있도록 하기 위해서 todo/admin.py 파일에 등록

```
from django.contrib import admin
from .models import Todo

# Register your models here.
admin.site.register(Todo)
```

# 프로젝트 설정

- 관리자 기능 설정
  - ✓ 관리자 페이지에 접속할 수 있도록 mytodo/urls.py 파일에 path 설정 확인

```
from django.contrib import admin
from django.urls import path

urlpatterns = [
    path("admin/", admin.site.urls),
]
```

# ToDo 목록 조회

- 뷰 함수 작성
  - ✓ 완료되지 않은 todo 목록을 출력할 뷰 함수를 todo/views.py에 작성

```
from django.shortcuts import render, redirect
from .models import Todo


def todo_list(request):
    todos = Todo.objects.filter(complete=False)
    return render(request, 'todo/todo_list.html', {'todos': todos})
```

# ToDo 목록 조회

- 템플릿 기능 작성
  - ✓ todo/templates/todo 디렉토리 안에 완료되지 않은 todo 목록을 출력할 todo_list.html 파일을 생성하고 작성

```
<html>
  <head>
    <title>TODO 목록 앱</title>
    <link
      rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
    />
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
    icons@1.7.1/font/bootstrap-icons.css">
  </head>
```

# ToDo 목록 조회

- 템플릿 기능 작성
  - ✓ todo/templates/todo 디렉토리 안에 완료되지 않은 todo 목록을 출력할 todo_list.html 파일을 생성하고 작성

```
<body>
<div class="container">
  <h1>TODO 목록 앱</h1>
  <p>
    <a href=""><i class="bi-plus"></i>TODO 추가</a>
    <a href="" class="btn btn-primary" style="float:right">완료한 TODO 목록</a>
  </p>
```

# ToDo 목록 조회

- 템플릿 기능 작성
  - ✓ todo/templates/todo 디렉토리 안에 완료되지 않은 todo 목록을 출력할 todo_list.html 파일을 생성하고 작성

```
<ul class="list-group">
  {% for todo in todos %}
  <li class="list-group-item">
    <a href="">{{ todo.title }}</a>
    {% if todo.important %}
      <span class="badge badge-danger">!</span>
    {% endif %}
    <div style="float:right">
      <a href="" class="btn btn-danger">완료</a>
      <a href="" class="btn btn-outline-primary">수정하기</a>
    </div>
  </li>
  {% endfor %}
 </ul>
</div>
 </body>
</html>
```

# ToDo 목록 조회

- URL 연결
  - ✓ 완료되지 않은 todo 목록을 출력하기 위한 url을 mytodo/urls.py에 작성

```
from django.contrib import admin
from django.urls import path
from todo import views


urlpatterns = [
    path("admin/", admin.site.urls),
    path('', views.todo_list, name='todo_list'),
]
```

# ToDo 목록 조회

- 테스트
  - ✓ 서버 실행: python manage.py runserver
  - ✓ 브라우저에서 접속: 127.0.0.1:8000

# ToDo 생성

- 데이터 삽입을 위한 form 생성
  - ✓ ToDo를 위한 폼을 todo/forms.py에 작성

```
from django import forms
from .models import Todo


class TodoForm(forms.ModelForm):
    class Meta:
        model = Todo
        fields = ('title', 'description', 'important')
```

# ToDo 생성

- 데이터 삽입을 위한 화면 생성
  - ✓ ToDo를 위한 화면을 todo/templates/todo 디렉토리에 todo_post.html 파일에 작성

```
<html>
  <head>
    <title>TODO 목록 앱</title>
    <link
      rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
    />
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
    icons@1.7.1/font/bootstrap-icons.css">
  </head>
```

# ToDo 생성

- 데이터 삽입을 위한 화면 생성
  - ✓ ToDo를 위한 화면을 todo/templates/todo 디렉토리에 todo_post.html 파일에 작성

```
<body>
<div class="container">
  <h1>TODO 추가하기</h1>
  <div class="container">
    <div class="row">
      <div class="col-md-12">
        <div class="card">
          <div class="card-body">
            <form method="POST">
              {% csrf_token %} {{ form.as_p }}
              <button type="submit" class="btn btn-primary">등록</button>
            </form>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
 </body>
</html>
```

# ToDo 생성

- CSRF
  - ✓ Cross-site request forgery 공격의 약자로 사용자가 자기 의지와는 상관없이 공격자가 의도한 행위를 특정 웹사이트에 요청하게 하는 방식으로 수행하는 공격
  - ✓ 결제 기능이 들어간 사이트나 개인정보를 수정하는 페이지를 상대로 CSRF공격을 수행한다면 많은 금전적 피해를 입을 수 있는데 실제로 옥션에서 예전에 CSRF 취약점으로 공격을 당해서 많은 피해을 입은 사례도 존재
  - ✓ CSRF 공격을 막기 위해 Django 에서는 CSRF 토큰 이라는 일종의 난수 비슷한 것을 도입했는데 이 토큰을 통해서 사이트 이외의 다른 사이트에서 HTTP 요청을 보내는 경우 유효하지 않은 요청이라 판단해서 403에러를 내면서 요청을 거부
  - ✓ 템플릿을 사용하는 경우

```
<form>
 {% csrf_token %}
 <input type="text" name="name"/>
 ....
</form>
```

# ToDo 생성

- 데이터 삽입을 위한 URL 처리 함수
  - ✓ 데이터 삽입 처리를 위한 함수를 todo/views.py 파일에 추가

```
from .forms import TodoForm

def todo_post(request):
    if request.method == "POST":
        form = TodoForm(request.POST)
        if form.is_valid():
            todo = form.save(commit=False)
            todo.save()
            return redirect('todo_list')
    else:
        form = TodoForm()
    return render(request, 'todo/todo_post.html', {'form': form})
```

# ToDo 생성

- 데이터 삽입을 위한 url 설정
  - ✓ 데이터 삽입 처리를 위한 함수를 mytodo/urls.py 파일에 추가

```
from django.contrib import admin
from django.urls import path
from todo import views


urlpatterns = [
    path("admin/", admin.site.urls),
    path('', views.todo_list, name='todo_list'),
    path('post/', views.todo_post, name='todo_post'),
]
```

# ToDo 생성

- 데이터 삽입을 위한 url 연결
  - ✓ 데이터 삽입 요청을 todo_list.html 파일에 연결

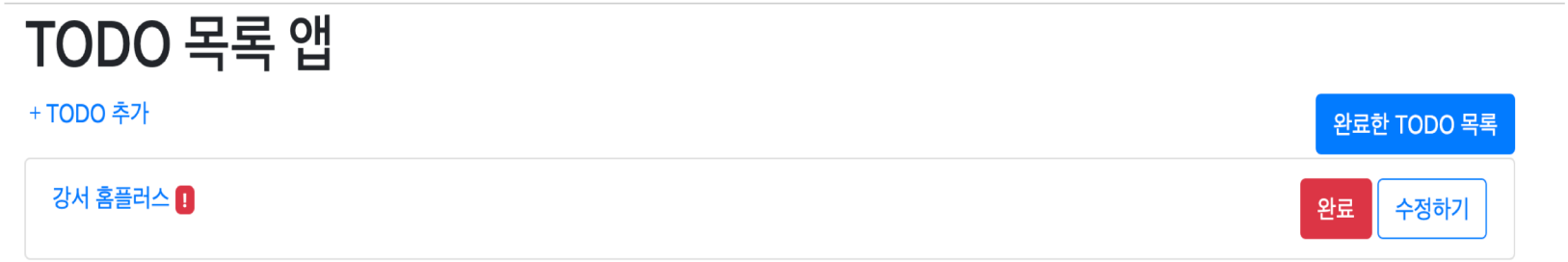
```
<a href="{% url 'todo_post' %}"><i class="bi-plus"></i>TODO 추가</a>
```

# ToDo 생성

- 확인
  - ✓ 브라우저에서 확인

# ToDo 생성

- 확인
  - ✓ 브라우저에서 확인



# ToDo 상세보기

- 데이터 상세 보기를 위한 URL 연결
  - ✓ 데이터 상세보기 요청을 todo_list.html 파일에 연결

```
<a href="{% url 'todo_detail' pk=todo.pk %}">{{ todo.title }}</a>
```

# ToDo 상세보기

- 데이터 상세 보기를 위한 화면 생성
  - ✓ 데이터 상세보기 요청 화면을 todo/templates/todo 디렉토리에 todo_detail.html 파일을 생성하고 작성

```
<html>
  <head>
    <title>TODO 목록 앱</title>
    <link
      rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
    />
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
     icons@1.7.1/font/bootstrap-icons.css">
  </head>
```

# ToDo 상세보기

- 데이터 상세 보기를 위한 화면 생성
  - ✓ 데이터 상세보기 요청 화면을 todo/templates/todo 디렉토리에 todo_detail.html 파일을 생성하고 작성

```
<body>
<div class="container">
  <h1>TODO 상세보기</h1>
  <div class="container">
    <div class="row">
      <div class="col-md-12">
        <div class="card">
          <div class="card-body">
            <h5 class="card-title">{{ todo.title }}</h5>
            </h5>
            <p class="card-text">{{ todo.description }}</p>
            <a href="{% url 'todo_list' %}" class="btn btn-primary">목록으로</a>
          </div>
        </div>
      </div>
    </div>
  </div>
 </body>
</html>
```

# ToDo 상세보기

- 데이터 상세 보기를 위한 뷰 처리 함수
  - ✓ 데이터 상세보기 요청을 처리하기 위한 함수를 todo/views.py 파일에 추가

```
def todo_detail(request, pk):
    todo = Todo.objects.get(id=pk)
    return render(request, 'todo/todo_detail.html', {'todo': todo})
```

# ToDo 상세보기

- 데이터 상세 보기를 위한 url 설정
  - ✓ 데이터 상세보기 요청 url을 mytodo/urls.py 파일에 추가

```
path('<int:pk>/', views.todo_detail, name='todo_detail')
```

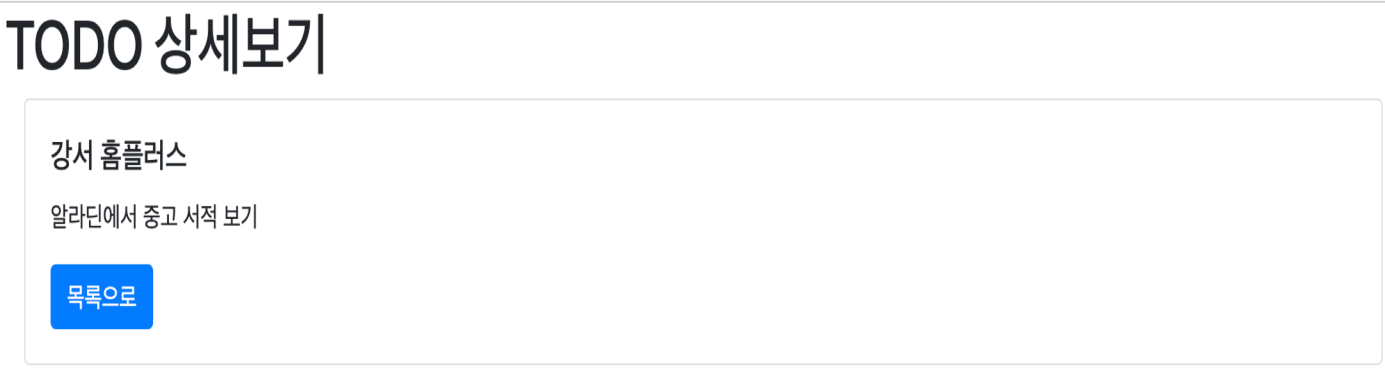
# ToDo 상세보기

- 데이터 상세 보기를 위한 url 설정
  - ✓ 데이터 상세보기 요청을 todo_list.html 파일의 title 출력 영역을 수정해서 링크 설정

```
<a href="{% url 'todo_detail' pk=todo.pk %}">{{ todo.title }}</a>
```

# ToDo 상세보기

- 확인
  - ✓ 브라우저에서 확인



# ToDo 수정

- 데이터 수정 보기 URL을 설정
  - ✓ todo_list.html에서 수정하기 요청 링크를 설정

```
<a href="{% url 'todo_edit' pk=todo.pk %}" class="btn btn-outline-primary">수정하기
    </a>
```

# ToDo 수정

- 데이터 수정 보기 처리 함수
  - ✓ 데이터 수정 요청을 위한 함수를 todo/views.py 파일에 추가

```
def todo_edit(request, pk):
    todo = Todo.objects.get(id=pk)
    if request.method == "POST":
        form = TodoForm(request.POST, instance=todo)
        if form.is_valid():
            todo = form.save(commit=False)
            todo.save()
            return redirect('todo_list')
    else:
        form = TodoForm(instance=todo)
    return render(request, 'todo/todo_edit.html', {'form': form})
```

# ToDo 상세보기

- 데이터 수정을 위한 화면 생성
  - ✓ 데이터 상세보기 요청 화면을 todo/templates/todo 디렉토리에 todo_edit.html 파일을 생성하고 작성

```
<html>
  <head>
    <title>TODO 목록 앱</title>
    <link
      rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
    />
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
     icons@1.7.1/font/bootstrap-icons.css">
  </head>
```

# ToDo 상세보기

- 데이터 수정을 위한 화면 생성
  - ✓ 데이터 상세보기 요청 화면을 todo/templates/todo 디렉토리에 todo_edit.html 파일을 생성하고 작성

```
<body>
  <div class="container">
    <h1>TODO 수정하기</h1>
    <div class="container">
      <div class="row">
        <div class="col-md-12">
          <div class="card">
            <div class="card-body">
              <form method="POST">
                {% csrf_token %} {{ form.as_p }}
                <button type="submit" class="btn btn-primary">수정</button>
              </form>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
  </body>
</html>
```

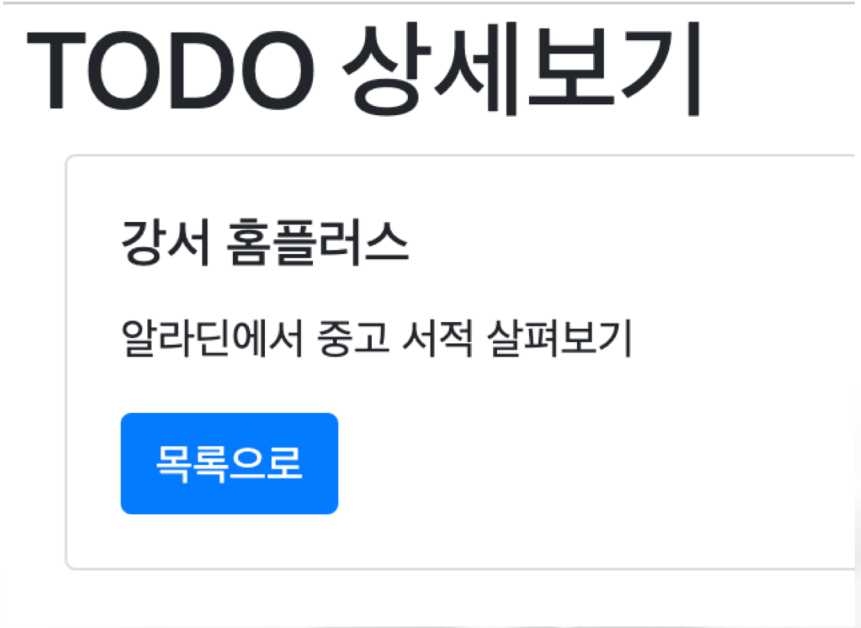
# ToDo 수정

- 데이터 수정 보기 URL 설정
  - ✓ 데이터 수정 요청을 위한 URL mytodo/urls.py 파일에 추가

    path('<int:pk>/edit/', views.todo_edit, name='todo_edit')

# ToDo 수정

- 확인
  - ✓ 브라우저에서 확인

# ToDo 완료

- ToDo 완료 버튼 과 완료 목록을 보여주는 URL 연결
  - ✓ todo_list.html 파일 수정

```
<html>
  <head>
    <title>TODO 목록 앱</title>
    <link
      rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
    />
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
    icons@1.7.1/font/bootstrap-icons.css">
  </head>
  <body>
  <div class="container">
    <h1>TODO 목록 앱</h1>
    <p>
      <a href="{% url 'todo_post' %}"><i class="bi-plus"></i>TODO 추가</a>
      <a href="{% url 'done_list' %}" class="btn btn-primary" style="float:right">완료한
    TODO 목록</a>
    </p>
```

# ToDo 완료

- ToDo 완료 버튼 과 완료 목록을 보여주는 URL 연결
  - ✓ todo_list.html 파일 수정

```
<ul class="list-group">
  {% for todo in todos %}
  <li class="list-group-item">
    <a href="{% url 'todo_detail' pk=todo.pk %}">{{ todo.title }}</a>
    {% if todo.important %}
      <span class="badge badge-danger">!</span>
    {% endif %}
    <div style="float:right">
      <a href="{% url 'todo_done' pk=todo.pk %}" class="btn btn-danger">완료</a>
      <a href="{% url 'todo_edit' pk=todo.pk %}" class="btn btn-outline-primary">수
정하기</a>
    </div>
  </li>
  {% endfor %}
</ul>
</body>
</div>
</html>
```

# ToDo 완료

- ToDo 완료 버튼 과 완료 목록 처리
  - ✓ ToDo 완료 와 완료 목록 보기를 처리할 함수를 todo/views.py 파일에 추가

```
def done_list(request):
    dones = Todo.objects.filter(complete=True)
    return render(request, 'todo/done_list.html', {'dones': dones})


def todo_done(request, pk):
    todo = Todo.objects.get(id=pk)
    todo.complete = True
    todo.save()
    return redirect('todo_list')
```

# ToDo 완료

- ToDo 완료 버튼 과 완료 목록 처리
  - ✓ ToDo 완료 목록 보기를 위한 화면을 todo/templates/todo 디렉토리에 done_list.html 파일로 생성하고 작성

```
<html>
  <head>
    <title>TODO 목록 앱</title>
    <link
      rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
    />
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
    icons@1.7.1/font/bootstrap-icons.css">
  </head>
```

# ToDo 완료

- ToDo 완료 버튼 과 완료 목록 처리
  - ✓ ToDo 완료 목록 보기를 위한 화면을 todo/templates/todo 디렉토리에 done_list.html 파일로 생성하고 작성

```
<body>
<div class="container">
  <h1>DONE 목록</h1>
  <p>
    <a href="{% url 'todo_list' %}" class="btn btn-primary">홈으로</a>
  </p>
  <ul class="list-group">
    {% for done in dones %}
    <li class="list-group-item">
      <a href="{% url 'todo_detail' pk=done.pk %}">{{ done.title }}</a>
      {% if done.important %}
        <span class="badge badge-danger">!</span>
      {% endif %}
    </li>
    {% endfor %}
  </ul>
</body>
</div>
</html>
```

# ToDo 완료

- ToDo 완료 버튼 과 완료 목록 URL 설정
  - ✓ mytodo/urls.py 파일에 URL 설정 코드 추가

```
from django.contrib import admin
from django.urls import path
from todo import views

urlpatterns = [
    path("admin/", admin.site.urls),
    path('', views.todo_list, name='todo_list'),
    path('post/', views.todo_post, name='todo_post'),
    path('<int:pk>/', views.todo_detail, name='todo_detail'),
    path('<int:pk>/edit/', views.todo_edit, name='todo_edit'),
    path('done/', views.done_list, name='done_list'),
    path('done/<int:pk>', views.todo_done, name='todo_done')
]
```