

Hello

채팅 서비스 만들기

더모아 팀 세션

world

본 PPT는 Django-tutorial 채팅 만들기(https://channels.readthedocs.io/en/stable/tutorial/part_1.html)
를 바탕으로 제작되었습니다.

장고 튜토리얼의 내용을 채팅 기능 구현에 초점을 맞춰 요약하였기 때문에
자세한 내용을 학습하려면 튜토리얼 원문을 보시거나 구글링을 통해 학습하는 것을 추천합니다.

1. 프로젝트 파일 생성 및 인덱스 페이지 구현



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)



Owner *

Repository name *

 Jae12ho ▾ / django_chat 

Great repository names are short and memorable. Need inspiration? How about [curly-pancake?](#)

Description (optional)

- ☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.
- ☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

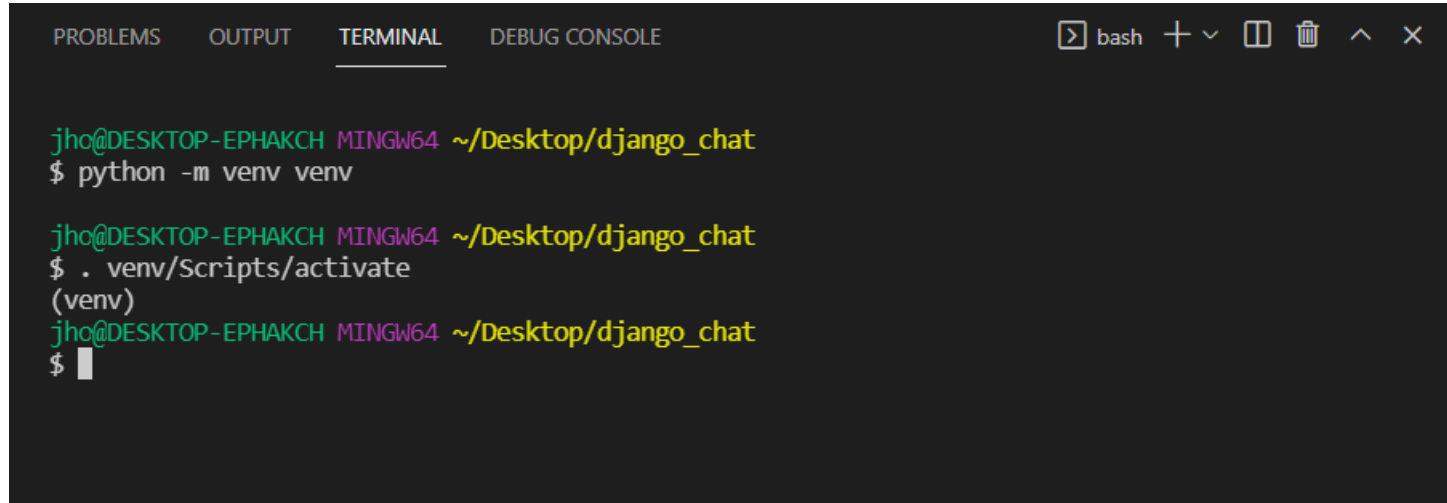
Skip this step if you're importing an existing repository.

- ☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)
- ☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)
- ☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

This will set  `main` as the default branch. Change the default name in your [settings](#).

Create repository

깃 허브에서 레퍼지토리를 생성해줍니다.



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
bash + - □ □ ^ ×

jho@DESKTOP-EPHAKCH MINGW64 ~/Desktop/django_chat
$ python -m venv venv

jho@DESKTOP-EPHAKCH MINGW64 ~/Desktop/django_chat
$ . venv/Scripts/activate
(venv)
jho@DESKTOP-EPHAKCH MINGW64 ~/Desktop/django_chat
$
```

생성한 레퍼지토리를 ‘git clone [레퍼지토리의 HTTPS주소]’ 명령어로 불러오고 불러와진 폴더에서 가상 환경을 생성하여 실행해줍니다.

```
(venv)
jho@DESKTOP-EPHAKCH MINGW64 ~/Desktop/django_chat
$ pip install django
Collecting django
  Using cached Django-3.2.6-py3-none-any.whl (7.9 MB)
Collecting pytz
```

```
(venv)
jho@DESKTOP-EPHAKCH MINGW64 ~/Desktop/django_chat
$ python -m pip install -U channels
```

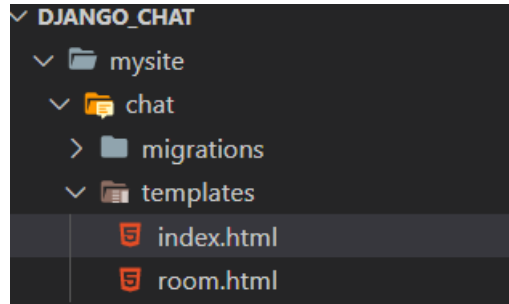
위 명령어를 입력해 장고와 채널을 설치해 줍니다.

```
(venv)
jho@DESKTOP-EPHAKCH MINGW64 ~/Desktop/django_chat
$ django-admin startproject mysite
(venv)
jho@DESKTOP-EPHAKCH MINGW64 ~/Desktop/django_chat
$ cd mysite/
(venv)
jho@DESKTOP-EPHAKCH MINGW64 ~/Desktop/django_chat/mysite
$ python manage.py startapp chat
(venv)
jho@DESKTOP-EPHAKCH MINGW64 ~/Desktop/django_chat/mysite
$
```

‘mysite’ 라는 이름의 프로젝트를 생성하고
‘chat’ 이라는 이름의 앱을 생성합니다.

```
settings.py X
mysite > mysite > settings.py
24
25 # SECURITY WARNING: don't run with debug turned on in production
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'chat',
35     'django.contrib.admin',
36     'django.contrib.auth',
37     'django.contrib.contenttypes',
38     'django.contrib.sessions',
39     'django.contrib.messages',
40     'django.contrib.staticfiles',
41 ]
42
```

앱을 생성했으니 settings.py에서 INSTALLED_APPS에 'chat' 앱을 추가해줍니다.



‘chat’앱 폴더 안에 ‘templates’폴더를 만들고,
그 안에 index.html, room.html 파일을 만들어 줍니다.

```
index.html X room.html
<!-- chat/templates/index.html -->
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>Chat Rooms</title>
</head>
<body>
  What chat room would you like to enter?<br>
  <input id="room-name-input" type="text" size="100"><br>
  <input id="room-name-submit" type="button" value="Enter">

  <script>
    document.querySelector('#room-name-input').focus();
    document.querySelector('#room-name-input').onkeyup = function(e) {
      if (e.keyCode === 13) { // enter, return
        document.querySelector('#room-name-submit').click();
      }
    };

    document.querySelector('#room-name-submit').onclick = function(e) {
      var roomName = document.querySelector('#room-name-input').value;
      window.location.pathname = '/' + roomName + '/';
    };
  </script>
</body>
</html>
```

Index.html의 코드를 작성해줍니다.
(위 코드 복사가능)

Vscode기준) 들여쓰기가 적용되지 않는다면 코드 전체 선택 후 Ctrl + K + F 를 하면 들여쓰기가 적용됩니다.

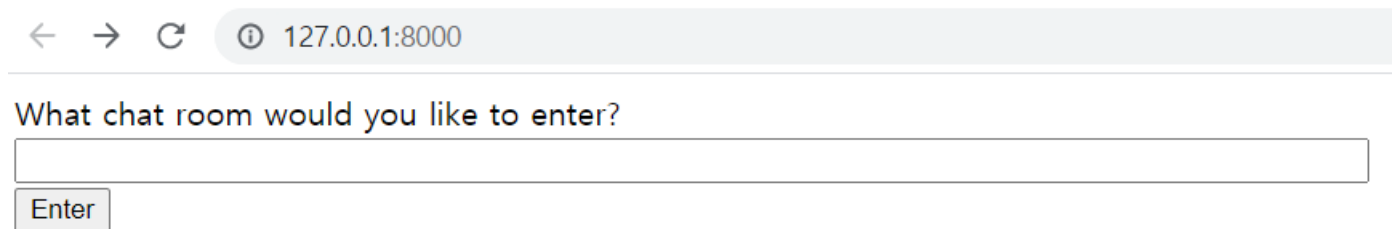
```
views.py ×
mysite > chat > views.py > index
1  from django.shortcuts import render
2
3  # Create your views here.
4  def index(request):
5      return render(request, 'index.html')
```

Index.html 파일을 만들었으니 이것을 렌더링 해줄 view코드를 작성합니다.

```
from django.contrib import admin
from django.urls import path
from chat.views import *

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

View에 접근할 수 있도록 urls.py에서 path를 추가해줍니다.



A screenshot of a web browser window. The address bar shows the URL '127.0.0.1:8000'. Below the address bar, the text 'What chat room would you like to enter?' is displayed. Underneath this text is a long, empty text input field. To the left of the input field is a small button labeled 'Enter'.

이제 'python manage.py runserver' 통해 웹을 실행시키고 확인해봅니다.

위와 같이 방을 선택하는 창이 뜨면 성공입니다 !

성공하셨으면 Ctrl + C를 통해 웹서버를 꺼줍니다.

2. 채팅 구현을 위한 asgi 설정

```
views.py  asgi.py 1 X
mysite > mysite > asgi.py > ...

# mysite/asgi.py
import os

from channels.routing import ProtocolTypeRouter, URLRouter
from Django.core.asgi import get_asgi_application

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "mysite.settings")

application = ProtocolTypeRouter({
    "http": get_asgi_application(),
})
```

이제 채팅방 구현에 필요한 Websocket을 사용하기 위해 우리가 만든 웹 서버를 asgi형태로 구동시켜야 합니다.

그러기 위해서 mysite/mysite 경로 안에 asgi.py파일이 없다면 생성하고

위 코드를 복사해서 넣어줍니다.

Vscode기준) 들여쓰기가 적용되지 않는다면 코드 전체 선택 후 Ctrl + K + F 를 하면 들여쓰기가 적용됩니다.

WebSocket 이란?

Websocket은 서버와 유저 간의 socket connection을 유지하여 실시간으로 양방향 통신이 가능하도록 하는 기술이다.

<https://duckdevelope.tistory.com/19> (이와 관련된 자세한 블로그 글)

Asgi 란?

Asgi란 Asynchronous Server Gateway Interface의 약자로, 웹 서버와 프레임워크(Django)를 비동기적으로 연결해주는 python의 표준 인터페이스이다.

기존에 우리가 사용하던 것은 동기적인 wsgi인데 이는 웹 소켓 통신이 불가하기 때문에, Agsi 방식으로 웹 소켓 통신을 이용할 수 있게 한다.

<https://wookkl.tistory.com/45> (이와 관련된 자세한 블로그 글)

```
INSTALLED_APPS = [  
    'channels',  
    'chat',  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
]
```

websocket과 asgi를 사용할 수 있도록 도와주는 channels라는 라이브러리를 사용할 것입니다.

Settings.py 파일에서 INSTALLED_APPS에 'channels'를 추가해줍니다.

```
# Channels  
ASGI_APPLICATION = 'mysite.asgi.application'|
```

그런 다음, 위 코드를 settings.py 파일 맨 아래에 넣어줍니다.


```
(venv)
jho@DESKTOP-EPHAKCH MINGW64 ~/Desktop/django_chat/mysite
$ python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
August 24, 2021 - 22:40:22
Django version 3.2.6, using settings 'mysite.settings'
Starting ASGI/Channels version 3.0.4 development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

이제 웹 서버가 asgi로 동작하는지 확인하기 위해

python manage.py runserver를 통해 웹 서버를 실행시킵니다.

밑줄 친 부분처럼 뜨면 성공 !

3. 채팅방 구현

```

room.html X
mysite > chat > templates > room.html > html
<!-- chat/templates/room.html -->
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>Chat Room</title>
</head>
<body>
  <textarea id="chat-log" cols="100" rows="20"></textarea><br>
  <input id="chat-message-input" type="text" size="100"><br>
  <input id="chat-message-submit" type="button" value="Send">
  {{ room_name|json_script:"room-name" }}
  <script>
    const roomName = JSON.parse(document.getElementById('room-name').textContent);

    const chatSocket = new WebSocket(
      'ws://'
      + window.location.host
      + '/ws/chat/'
      + roomName
      + '/'
    );

    chatSocket.onmessage = function(e) {
      const data = JSON.parse(e.data);
      document.querySelector('#chat-log').value += (data.message + '\n');
    };

    chatSocket.onclose = function(e) {
      console.error('Chat socket closed unexpectedly');
    };

    document.querySelector('#chat-message-input').focus();
    document.querySelector('#chat-message-input').onkeyup = function(e) {
      if (e.keyCode === 13) { // enter, return
        document.querySelector('#chat-message-submit').click();
      }
    };

    document.querySelector('#chat-message-submit').onclick = function(e) {
      const messageInputDom = document.querySelector('#chat-message-input');
      const message = messageInputDom.value;
      chatSocket.send(JSON.stringify({
        'message': message
      }));
      messageInputDom.value = '';
    };
  </script>
</body>
</html>

```

이제 채팅방을 만들 차례입니다.

room.html 파일에 왼쪽의 코드를 복사해서 넣어주세요.

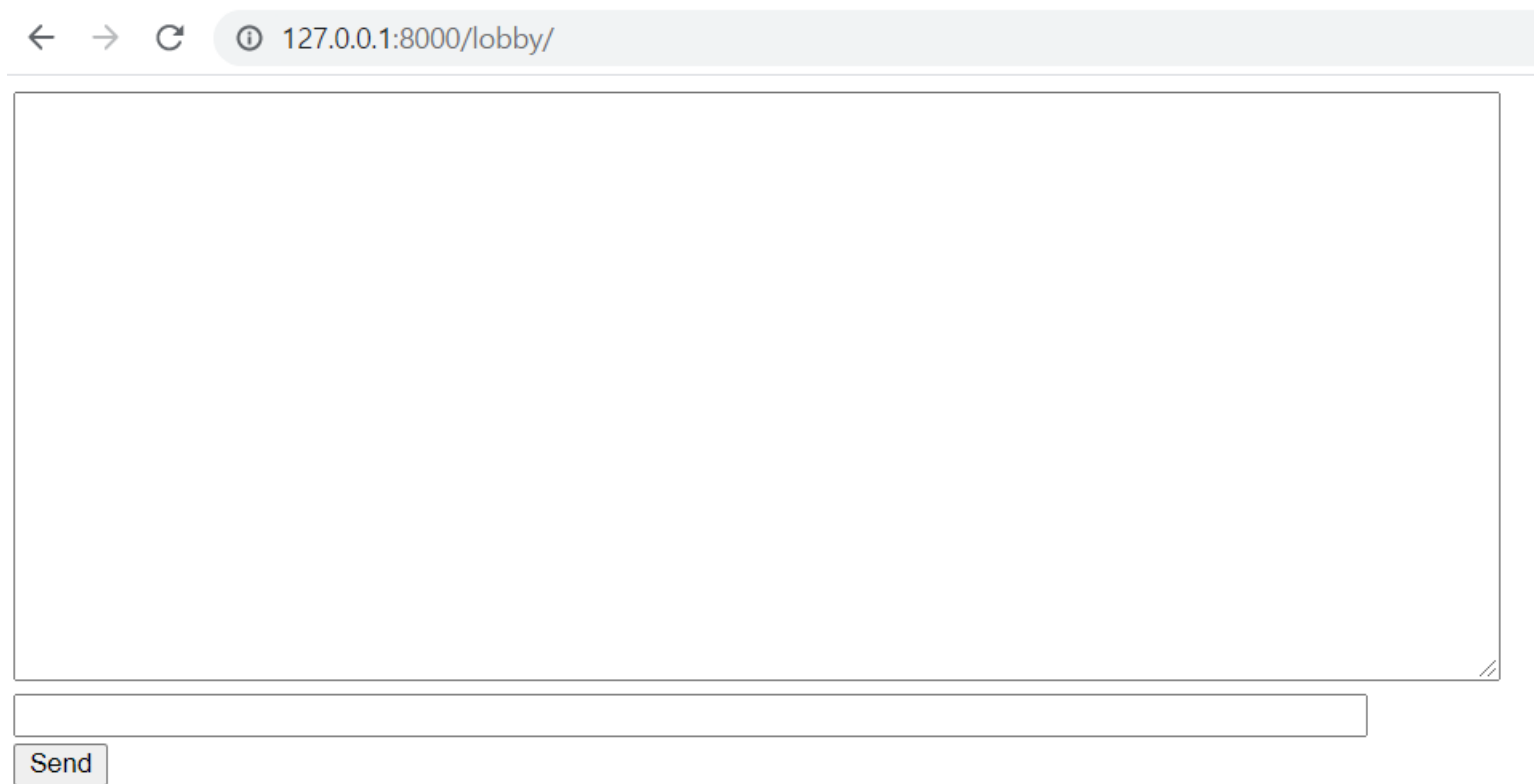
Vscode기준) 들여쓰기가 적용되지 않는다면 코드 전체 선택 후 Ctrl + K + F 를 하면 들여쓰기가 적용됩니다.

```
room.html  views.py  X
mysite > chat > views.py > room
1  from django.shortcuts import render
2
3  # Create your views here.
4  def index(request):
5      return render(request, 'index.html')
6
7  def room(request, room_name):
8      return render(request, 'room.html', {
9          'room_name': room_name
10     })
```

room.html 를 작성했으니 room.html 파일을 렌더링 해줄 view코드를 추가해줍니다.

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', index, name="index"),
    path('<str:room_name>/', room, name='room'),
]
```

이전과 마찬가지로 urls.py에 room으로 연결해줄 url path를 추가해줍니다.

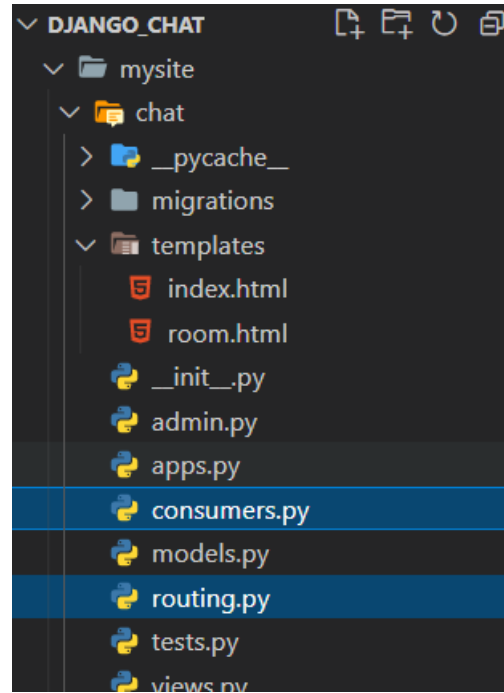


이제 웹 서버를 실행하고

Index페이지에서 'lobby'를 입력하면 채팅방에 들어와집니다.

이제 메시지를 보내면 채팅창에 메시지가 출력되는 것을 만들겁니다.

4. 채팅 구현을 위한 웹소켓 연결



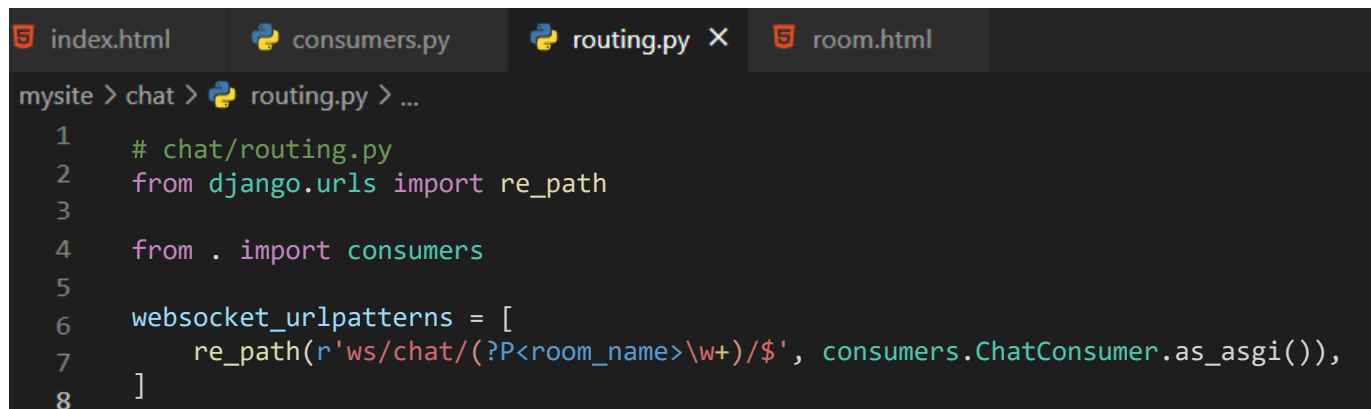
‘chat’앱 폴더 안에 ‘consumers.py’, ‘routing.py’파일을 생성해줍니다.

```
index.html consumers.py X room.html
mysite > chat > consumers.py > ChatConsumer > receive
1  # chat/consumers.py
2  import json
3  from channels.generic.websocket import WebsocketConsumer
4
5  class ChatConsumer(WebsocketConsumer):
6      def connect(self):
7          self.accept()
8
9      def disconnect(self, close_code):
10         pass
11
12     def receive(self, text_data):
13         text_data_json = json.loads(text_data)
14         message = text_data_json['message']
15
16         self.send(text_data=json.dumps({
17             'message': message
```

consumers.py 파일에 위 코드를 복사해서 넣어줍니다.

consumers.py은 웹 소켓의 연결, 연결해제, 수신의 작업을 수행합니다.

Vscode기준) 들여쓰기가 적용되지 않는다면 코드 전체 선택 후 Ctrl + K + F 를 하면 들여쓰기가 적용됩니다.



```
mysite > chat > routing.py > ...
1  # chat/routing.py
2  from django.urls import re_path
3
4  from . import consumers
5
6  websocket_urlpatterns = [
7      re_path(r'ws/chat/(?P<room_name>\w+)/$', consumers.ChatConsumer.as_asgi()),
8  ]
```

routing.py 파일에 위 코드를 복사해서 넣어줍니다.

routing.py는 HTTP 요청을 consumers.py으로 연결해주는 중간다리 역할을 수행합니다.

Vscode기준) 들여쓰기가 적용되지 않는다면 코드 전체 선택 후 Ctrl + K + F 를 하면 들여쓰기가 적용됩니다.

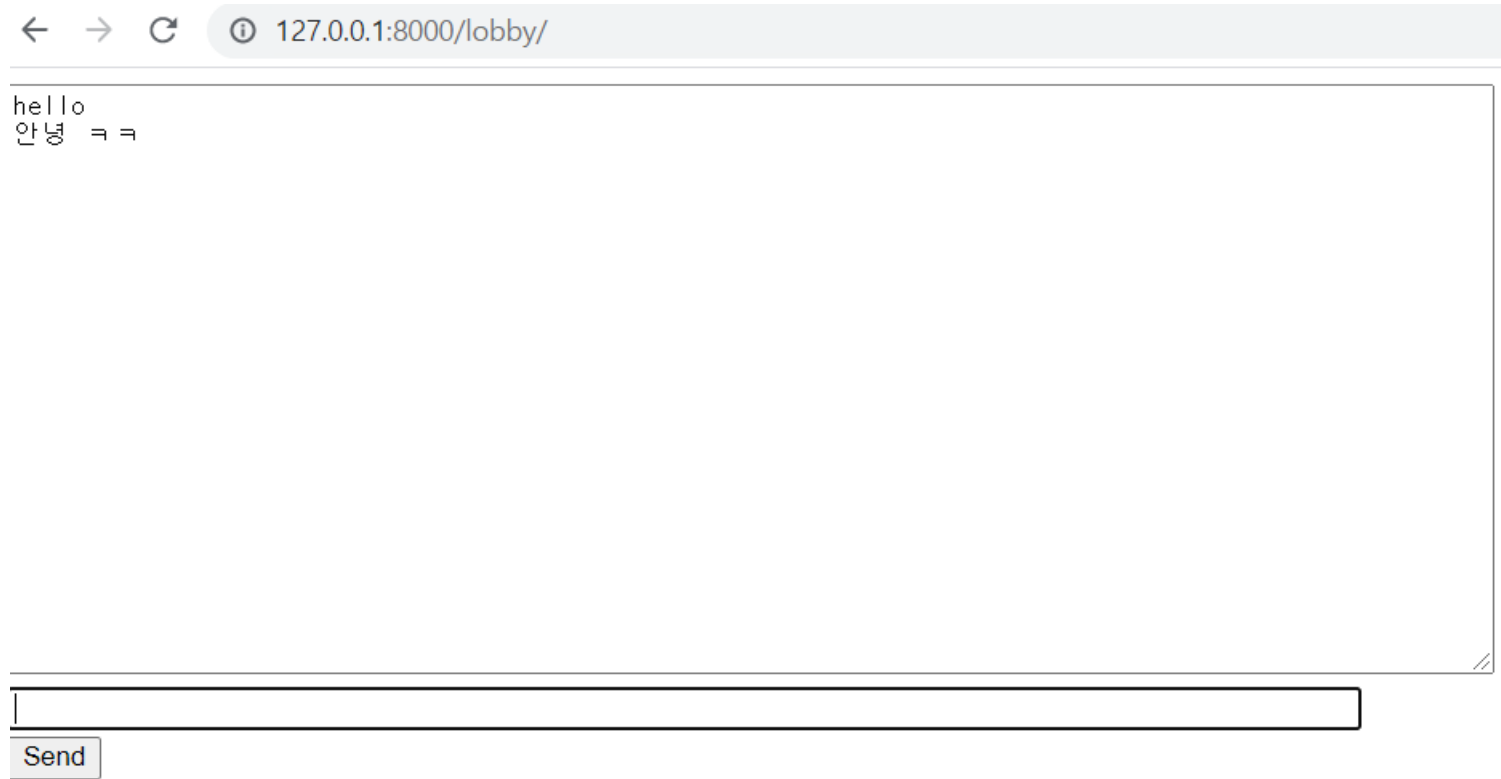
```
index.html consumers.py routing.py asgi.py × room.html
mysite > mysite > asgi.py > ...
1  # mysite/asgi.py
2  import os
3
4  from channels.auth import AuthMiddlewareStack
5  from channels.routing import ProtocolTypeRouter, URLRouter
6  from django.core.asgi import get_asgi_application
7  import chat.routing
8
9  os.environ.setdefault("DJANGO_SETTINGS_MODULE", "mysite.settings")
10
11 application = ProtocolTypeRouter({
12     "http": get_asgi_application(),
13     "websocket": AuthMiddlewareStack(
14         URLRouter(
15             chat.routing.websocket_urlpatterns
16         )
17     ),
18 })
```

아까 작성했던 asgi.py코드를 위 코드로 바꿔줍니다.

이 코드는 웹 서버가 web socket연결도 받을 수 있게 해줍니다.

복사 가능.

Vscode기준) 들여쓰기가 적용되지 않는다면 코드 전체 선택 후 Ctrl + K + F 를 하면 들여쓰기가 적용됩니다.



웹 서버를 실행시키고 자신이 입력한 채팅이 채팅창에 출력되는 모습을 확인할 수 있습니다.

하지만, 이는 자신의 페이지에서만 출력되기 때문에,
다른 사람의 페이지에서도 출력되도록 하는 작업을 해줘야 합니다.

5. 완벽한 채팅기능 구현

```
# mysite/settings.py
# Channels
ASGI_APPLICATION = 'mysite.asgi.application'
CHANNEL_LAYERS = {
    'default': {
        'BACKEND': 'channels_redis.core.RedisChannelLayer',
        'CONFIG': {
            "hosts": [('127.0.0.1', 6379)],
        },
    },
}
```

settings.py 파일의 끝부분에 CHANNEL_LAYERS코드를 복사해서 추가해줍니다.

CHANNEL_LAYER는 채널과 그룹의 개념을 가지고 있는데,

‘그룹: 채팅방, 채널: 유저’ 의 느낌으로 생각하면 됩니다.

채널 레이어를 통해 다른 사람과 채팅을 주고받을 수 있도록 구현할 수 있습니다.

```
index.html consumers.py X
mysite > chat > consumers.py > ChatConsumer > chat_message
1 # chat/consumers.py
2 import json
3 from channels.generic.websocket import AsyncWebsocketConsumer
4
5 class ChatConsumer(AsyncWebsocketConsumer):
6     async def connect(self):
7         self.room_name = self.scope['url_route']['kwargs']['room_name']
8         self.room_group_name = 'chat_%s' % self.room_name
9
10        # Join room group
11        await self.channel_layer.group_add(
12            self.room_group_name,
13            self.channel_name
14        )
15
16        await self.accept()
17
18    async def disconnect(self, close_code):
19        # Leave room group
20        await self.channel_layer.group_discard(
21            self.room_group_name,
22            self.channel_name
23        )
24
25    # Receive message from WebSocket
26    async def receive(self, text_data):
27        text_data_json = json.loads(text_data)
28        message = text_data_json['message']
29
30        # Send message to room group
31        await self.channel_layer.group_send(
32            self.room_group_name,
33            {
34                'type': 'chat_message',
35                'message': message
36            }
37        )
38
39    # Receive message from room group
40    async def chat_message(self, event):
41        message = event['message']
42
43        # Send message to WebSocket
44        await self.send(text_data=json.dumps({
45            'message': message
46        })))
```

consumers.py 파일의 코드를 왼쪽의 코드로 바꿔줍니다.
복사 가능.

바뀐 코드는 앞 슬라이드에서 말한
채널 레이어를 사용할 수 있도록 합니다.

각 코드에 관한 자세한 설명은 장고 튜토리얼에 나와있습니다.

https://channels.readthedocs.io/en/stable/tutorial/part_2.html
(페이지의 아래쪽에 있습니다.)

Vscode기준) 들여쓰기가 적용되지 않는다면 코드 전체 선택 후 Ctrl + K + F 를 하면 들여쓰기가 적용됩니다.

```
requirements.txt X
django_chat > requirements.txt
1
2 channels==3.0.4
3 channels-redis==3.3.0
4 Django==3.2.5
```

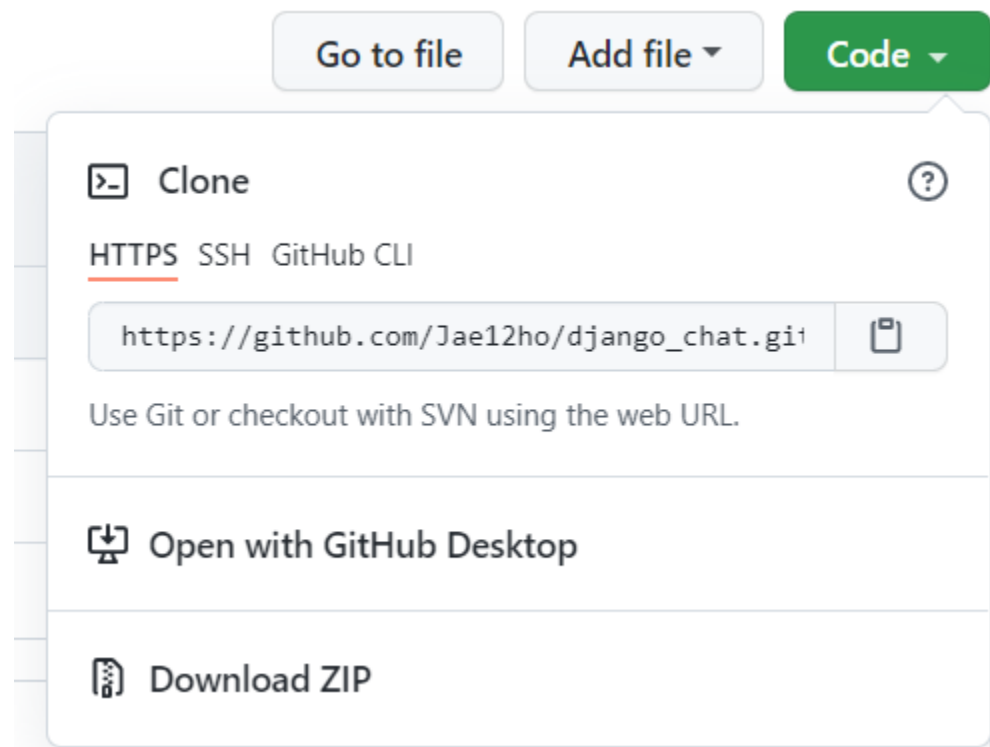
```
requirements.txt .gitignore X
django_chat > .gitignore
1 |venv/
```

이제 Django_chat 폴더 안에 requirements.txt, .gitignore 파일을 위와 같이 만들어 주고,

```
jho@DESKTOP-EPHAKCH MINGW64 ~/Desktop/django_chat/mysite (main)
$ git add .
(venv)
jho@DESKTOP-EPHAKCH MINGW64 ~/Desktop/django_chat/mysite (main)
$ git commit -m "django tutorial chat"
(venv)
jho@DESKTOP-EPHAKCH MINGW64 ~/Desktop/django_chat/mysite (main)
$ git push
```

GitHub에 푸쉬해줍니다.

6. AWS 서버에서 실행하기



자신이 올린 깃허브 레퍼지토리의 HTTPS주소를 복사합니다.

이제 마지막으로 Docker와 Redis를 설치해야 합니다.

Docker와 Redis에 대한 설명은 아래 블로그를 참고해주세요.

Docker - <https://pkh11.medium.com/docker-%EB%8F%84%EC%BB%A4%EB%9E%80-%EB%AC%B4%EC%97%87%EC%9D%B8%EA%B0%80-8b93d1a46aa8>

Redis - <https://velog.io/@hyeondey/Redis-%EB%9E%80-%EB%AC%B4%EC%97%87%EC%9D%BC%EA%B9%8C>

해당 튜토리얼에서 채팅을 구현할 때 채널 레이어를 사용하는데,
채널 레이어는 Redis를 채팅 로그를 저장하는 저장소로 사용하게 됩니다.

```
ubuntu@ip-172-31-7-7:~$ git clone https://github.com/Jae12ho/django_chat.git
```

저번 배포시간에 만들었던 AWS가상환경을 켜고 위 명령어로 본인의 레퍼지토리를 불러옵니다.

```
(venv) ubuntu@ip-172-31-7-7:~$ cd django_chat/  
(venv) ubuntu@ip-172-31-7-7:~/django_chat$ sudo apt-get install virtualenv  
(venv) ubuntu@ip-172-31-7-7:~/django_chat$ virtualenv -p python3 venv  
(venv) ubuntu@ip-172-31-7-7:~/django_chat$ source venv/bin/activate
```

불러온 레퍼지토리로 들어가서 아래의 명령어를 통해 가상환경을 생성해줍니다.

```
sudo apt-get install virtualenv (가상환경 설치)  
virtualenv -p python3 venv (가상환경 생성)  
source venv/bin/activate (가상환경 실행)
```

```
(venv) ubuntu@ip-172-31-7-7:~/django_chat$ pip install -r requirements.txt
```

가상환경을 실행하였으면 pip install -r requirements.txt 명령어를 통해 필요한 라이브러리들을 설치해줍니다.

도커 설치하기

```
$ sudo apt update -y
```

```
$ sudo apt install -y apt-transport-https ca-certificates curl software-properties-common
```

도커 설치를 위한 몇 가지 패키지 설치

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

```
$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
```

```
$ sudo apt update -y
```

```
$ sudo apt install -y docker-ce
```

```
$ sudo systemctl start docker
```

도커 설치 및 실행

채팅 기능을 구현하기 위해 도커를 설치해야 합니다.

위 명령어를 입력하여 도커를 설치해줍니다.

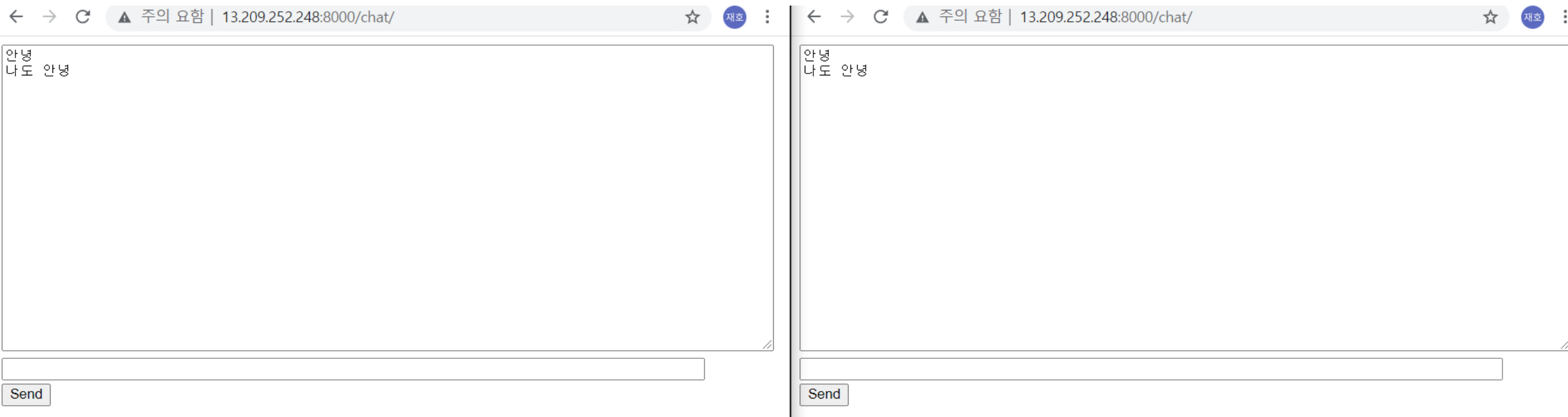
복사 가능.

Redis 설치하기

```
$ sudo docker pull redis:5
```

```
$ sudo docker run --name myredis -d -p 6379:6379 redis
```

Redis 설치까지 완료하면 끝 !



이제 `python manage.py runserver 0.0.0.0:8000` 명령어로 웹 서버를 실행시켜서 확인해보면

위 사진처럼 다른 사람에게도 채팅이 보여집니다.

다른 사람에게 서버 주소를 공유하고 채팅이 잘 작동하는지 확인해보세요 !