

# CM3035 Advanced Web Development

Lesson L12

Created by Ben Gay

# Introduction to Web Sockets

real-time applications e.g. Online Chat

Dual-channel full-duplex bidirectional communication simply means that the client and the server can talk in real-time without having to continuously make requests, and contrary to HTTP where request is always initiated by the client, and response is processed by the server, with web sockets the communication can go either way i.e. from server to client or client to server.

# ASGI (Asynchronous Server Gateway Interface)

In order to handle a WebSocket connection, channels has **routing.py** and **consumers.py**.

Let's begin with starting Virtual environment , **L12proj** and **L12app**

**pip3 install channels**

# L12proj/settings.py

```
33
34 INSTALLED_APPS = [
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     'L12app.apps.L12AppConfig',
42     'channels',
43     # 'L12app',
44 ]
45
```

# L12proj/asgi.py

```
4
5 import os
6 from channels.auth import AuthMiddlewareStack
7 from channels.routing import ProtocolTypeRouter, URLRouter
8 from django.core.asgi import get_asgi_application
9 import L12app.routing
10
11 os.environ.setdefault("DJANGO_SETTINGS_MODULE", "L12proj.settings")
12
13 application = ProtocolTypeRouter({
14     "http": get_asgi_application(),
15     "websocket": AuthMiddlewareStack(
16         URLRouter(
17             L12app.routing.websocket_urlpatterns
18         )
19     ),
20 })
```

# L12proj/settings.py

```
127  
128 ASGI_APPLICATION = 'L12proj.asgi.application'  
129
```

# templates/chat/index.html

```
index.html — ~/Desktop/djangoven/L12proj/L12app
settings.py  untitled  index.html  consumers.py  routing.py  asgi.py  urls.py — L12a...  urls.py — L12p

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8"/>
5   <title>Chat Room</title>
6 </head>
7 <body>
8   What chat room would you like to enter?<br>
9   <input id="room-name-input" type="text" size="100"><br>
0   <input id="room-name-submit" type="button" value="Enter">
1
2   <script>
3     document.querySelector('#room-name-input').focus();
4     document.querySelector('#room-name-input').onkeyup = function(e) {
5       if (e.keyCode === 13) { // enter, return
6         document.querySelector('#room-name-submit').click();
7       }
8     };
9
0     document.querySelector('#room-name-submit').onclick = function(e) {
1       var roomName = document.querySelector('#room-name-input').value;
2       window.location.pathname = 'L12app/' + roomName + '/';
3     };
4   </script>
5 </body>
6 </html>
7
```

# L12app/views.py

```
1  from django.shortcuts import render
2  |
3  def index(request):
4      return render(request, 'chat/index.html')
```



# L12app/urls.py

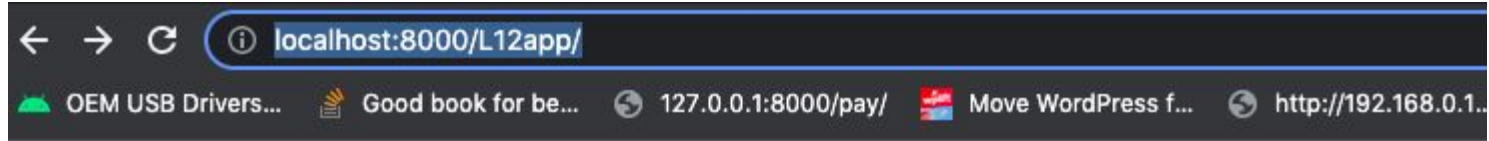
```
settings.py  untitled  index.html  consumers.py  routing.py  asgi.py
1  from django.urls import path
2  from . import views
3  from django.urls import include, path
4  #from L12app.views import (indexView , postFriend , checkNick)
5
6
7
8  urlpatterns = [
9      path('', views.index, name='index'),
10     #path('<str:room_name>/', views.room, name='room'),
11 ]
12
```

# L12proj/urls.py

```
5  """
6  from django.contrib import admin
7  from django.urls import path , include
8
9  urlpatterns = [
10     path('admin/', admin.site.urls),
11     path('L12app/', include('L12app.urls')),
12 ]
13
```

# python manage.py runserver

<http://localhost:8000/L12app/>



What chat room would you like to enter?

Enter

# Channel Layers

```
pip3 install channels_redis
```

# L12proj/settings.py

```
1 CHANNEL_LAYERS = {  
2     "default": {  
3         "BACKEND": "channels_redis.core.RedisChannelLayer",  
4         "CONFIG": {  
5             "hosts": [("127.0.0.1", 6379)],  
6         },  
7     },  
8 }  
9
```

Numbers

# L12app/urls.py

```
1 from django.urls import path
2 from . import views
3 from django.urls import include, path
4 #from L12app.views import (indexView , postFriend , checkNickName
5
6
7
8 urlpatterns = [
9     path('', views.index, name='index'),
10    path('<str:room_name>/', views.room, name='room'),
11 ]
12
```

# L12app/views.py

```
1  from django.shortcuts import render
2
3  def index(request):
4      return render(request, 'chat/index.html')
5
6  def room(request, room_name):
7      return render(request, 'chat/room.html', {
8          'room_name': room_name
9      })
10
```

# templates/chat/room.html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8"/>
5   <title>Chat Room</title>
6 </head>
7 <body>
8   <textarea id="chat-log" cols="100" rows="20"></textarea><br>
9   <input id="chat-message-input" type="text" size="100"><br>
10  <input id="chat-message-submit" type="button" value="Send">
11  {{ room_name|json_script:"room-name" }}
12 <script>
13   const roomName = JSON.parse(document.getElementById('room-name').textContent);
14
15   const chatSocket = new WebSocket(
16     'ws://'
17     + window.location.host
18     + '/ws/L12app/'
19     + roomName
20     + '/'
21   );
22
23   // onmessage - An event listener to be called when a message is received from the server.
24   chatSocket.onmessage = function(e) {
25     // JSON.parse() converts the JSON object back into the original object,
26     // then examine and act upon its contents.
27     const data = JSON.parse(e.data);
```



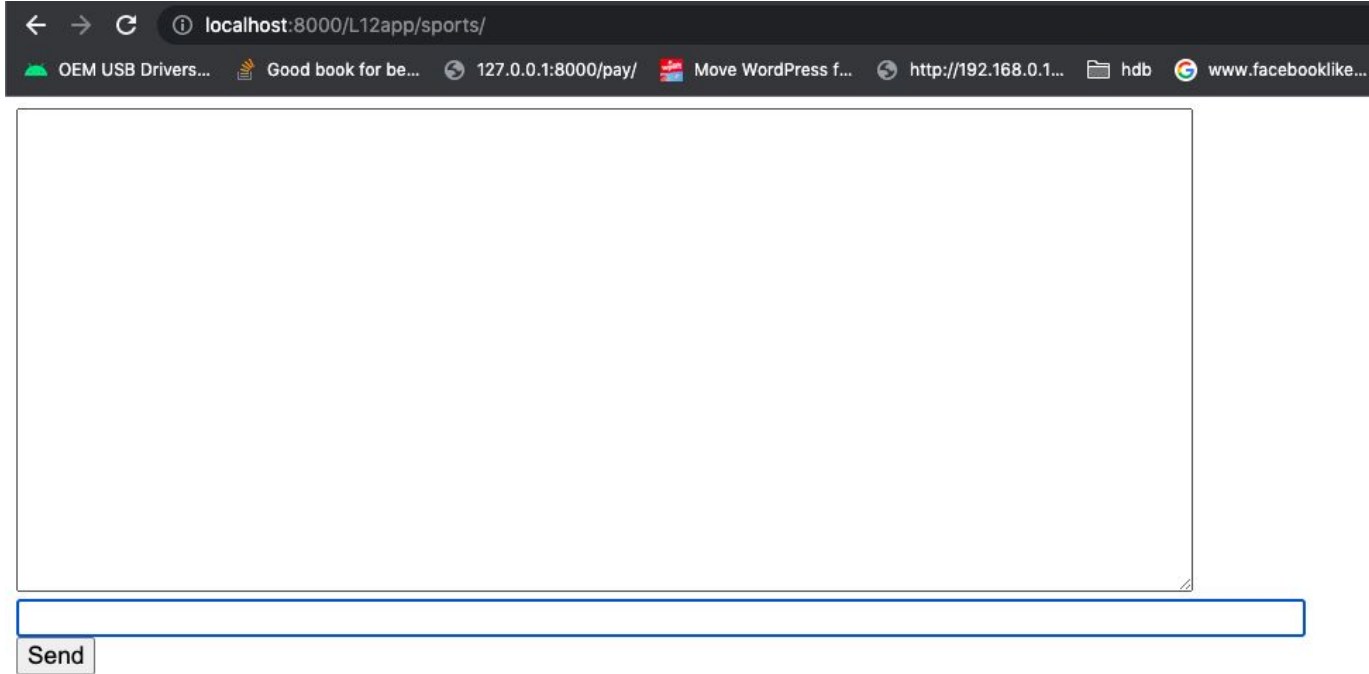
# templates/chat/room.html

```
25 // JSON.parse() converts the JSON object back into the original object,  
26 // then examine and act upon its contents.  
27 const data = JSON.parse(e.data);  
28 document.querySelector('#chat-log').value += (data.message + '\n');  
29 };  
30  
31 // onclose - An event listener to be called when the connection is closed.  
32 chatSocket.onclose = function(e) {  
33     console.error('Chat socket closed unexpectedly');  
34 };  
35  
36 document.querySelector('#chat-message-input').focus();  
37 document.querySelector('#chat-message-input').onkeyup = function(e) {  
38     if (e.keyCode === 13) { // enter, return  
39         document.querySelector('#chat-message-submit').click();  
40     }  
41 };  
42  
43 document.querySelector('#chat-message-submit').onclick = function(e) {  
44     const messageInputDom = document.querySelector('#chat-message-input');  
45     const message = messageInputDom.value;  
46  
47     // Send the msg object as a JSON-formatted string.  
48     chatSocket.send(JSON.stringify({  
49         'message': message  
50     }));
```

# templates/chat/room.html

```
61  
62      // Blank the text input element, ready to receive the next line of text from the user.  
63      messageInputDom.value = '';  
64      };  
65      </script>  
66 </body>  
67 </html>  
68
```

# python3 manage.py runserver



# SyncConsumer WebSocket

## L12app/consumer.py

```
1 import json
2 from asgiref.sync import async_to_sync
3 from channels.generic.websocket import WebsocketConsumer
4
5
6 class ChatConsumer(WebsocketConsumer):
7     def connect(self):
8         self.room_name = self.scope['url_route']['kwargs']['room_name']
9         self.room_group_name = 'chat_%s' % self.room_name
10
11         # Join room group
12         async_to_sync(self.channel_layer.group_add)(
13             self.room_group_name,
14             self.channel_name
15         )
16
17         self.accept()
18
19     def disconnect(self, close_code):
20         # Leave room group
21         async_to_sync(self.channel_layer.group_discard)(
22             self.room_group_name,
23             self.channel_name
24         )
25
26         # Receive message from WebSocket
27     def receive(self, text_data):
```

```
22         self.room_group_name,
23         self.channel_name
24     )
25
26     # Receive message from WebSocket
27     def receive(self, text_data):
28         text_data_json = json.loads(text_data)
29         message = text_data_json['message']
30
31         # Send message to room group
32         async_to_sync(self.channel_layer.group_send)(
33             self.room_group_name,
34             {
35                 'type': 'chat_message',
36                 'message': message
37             }
38         )
39
40         # Receive message from room group
41     def chat_message(self, event):
42         message = event['message']
43
44         # Send message to WebSocket
45         self.send(text_data=json.dumps({
46             'message': message
47         })))
48
```

# L12app/routing.py

```
1 from django.urls import re_path
2 from . import consumers
3
4 websocket_urlpatterns = [
5     re_path(r'ws/L12app/(?P<room_name>\w+)/$', consumers.ChatConsumer.as_asgi()),
6 ]
7
```

# L12proj/asgi.py

```
35 import os
36 from channels.auth import AuthMiddlewareStack
37 from channels.routing import ProtocolTypeRouter, URLRouter
38 from django.core.asgi import get_asgi_application
39 import L12app.routing
40
41 os.environ.setdefault("DJANGO_SETTINGS_MODULE", "L12proj.settings")
42
43 application = ProtocolTypeRouter({
44     "http": get_asgi_application(),
45     "websocket": AuthMiddlewareStack(
46         URLRouter(
47             L12app.routing.websocket_urlpatterns
48         )
49     ),
50 })
51
```

# Run Django server and Redis server

Install Redis server:

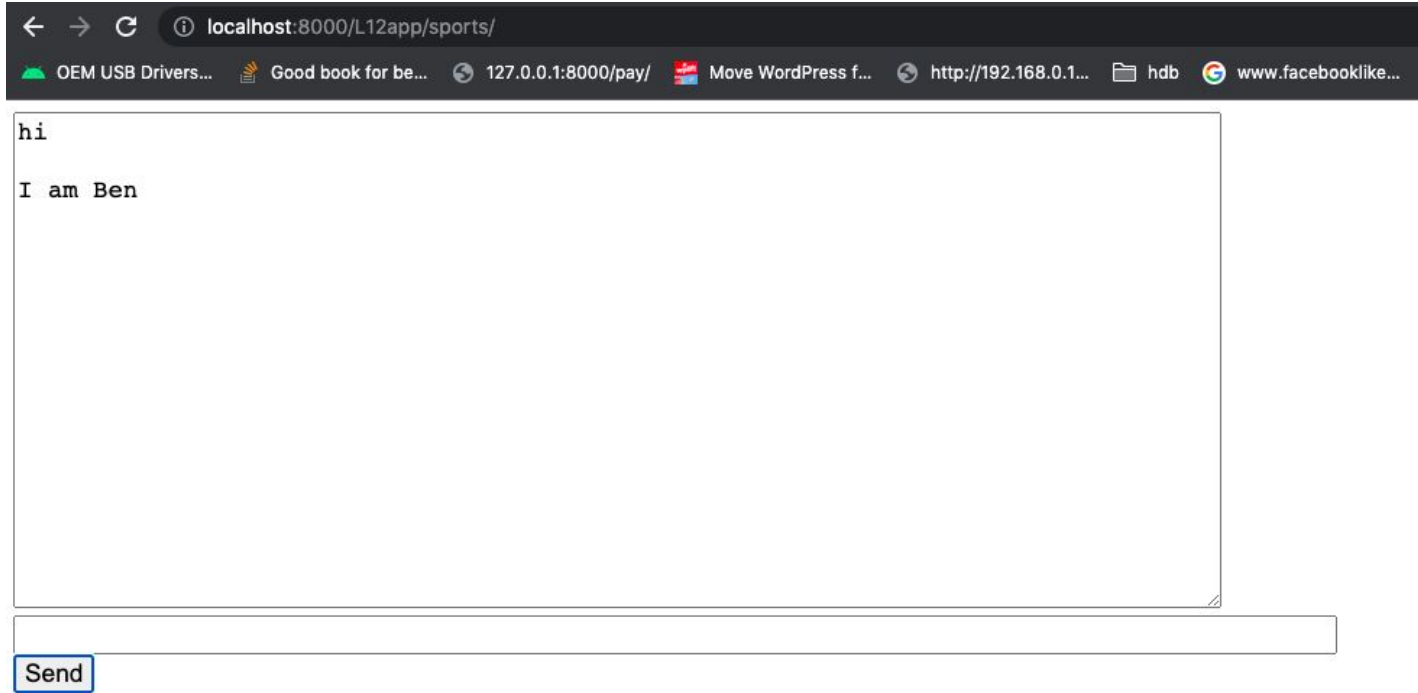
**brew install redis**

Run Redis server:

**brew services start redis**

**Python3 manage.py runserver**

# Output



A screenshot of a web browser window. The address bar shows 'localhost:8000/L12app/sports/'. The browser's tab bar contains several tabs: 'OEM USB Drivers...', 'Good book for be...', '127.0.0.1:8000/pay/', 'Move WordPress f...', 'http://192.168.0.1...', 'hdb', and 'www.facebooklike...'. The main content area is a large white rectangle containing the text 'hi' and 'I am Ben' on separate lines. Below this area is a long, empty white input field. At the bottom left, below the input field, is a small blue button with the text 'Send'.

hi

I am Ben

Send



End of Lesson 12