

Tutorial Part 1: Basic Setup

Note


If you encounter any issue during your coding session, please see the [Troubleshooting](#) section.

In this tutorial we will build a simple chat server. It will have two pages:

- An index view that lets you type the name of a chat room to join.
- A room view that lets you see messages posted in a particular chat room.

The room view will use a WebSocket to communicate with the Django server and listen for any messages that are posted.

We assume that you are familiar with basic concepts for building a Django site. If not we recommend you complete [the Django tutorial](#) first and then come back to this tutorial.

We assume that you have [Django installed](#) already. You can tell Django is installed and which version by running the following command in a shell prompt (indicated by the  prefix):

```
$ python3 -m django --version
```

We also assume that you have [Channels and Daphne installed](#) already. You can check by running the following command:

```
$ python3 -c 'import channels; import daphne; print(channels.__version__, daphne.__version__)'
```

This tutorial is written for Channels 4.0, which supports Python 3.7+ and Django 3.2+. If the Channels version does not match, you can refer to the tutorial for your version of Channels by using the version switcher at the bottom left corner of this page, or update Channels to the newest version.

This tutorial also **uses Docker** to install and run Redis. We use Redis as the backing store for the channel layer, which is an optional component of the Channels library that we use in the tutorial. [Install Docker](#) from its official website - there are official runtimes for Mac OS and Windows that make it easy to use, and packages for many Linux distributions where it can run natively.

Note

While you can run the standard Django `runserver` without the need for Docker, the channels features we'll be using in later parts of the tutorial will need Redis to run, and we recommend Docker as the easiest way to do this.

Creating a project

If you don't already have a Django project, you will need to create one.

From the command line, `cd` into a directory where you'd like to store your code, then run the following command:

```
$ django-admin startproject mysite
```

This will create a `mysite` directory in your current directory with the following contents:

```
mysite/  
  manage.py  
  mysite/  
    __init__.py  
    asgi.py  
    settings.py  
    urls.py  
    wsgi.py
```

Creating the Chat app

We will put the code for the chat server in its own app.

Make sure you're in the same directory as `manage.py` and type this command:

```
$ python3 manage.py startapp chat
```

That'll create a directory `chat`, which is laid out like this:

```
chat/
  __init__.py
  admin.py
  apps.py
  migrations/
    __init__.py
  models.py
  tests.py
  views.py
```

For the purposes of this tutorial, we will only be working with `chat/views.py` and `chat/__init__.py`. So remove all other files from the `chat` directory.

After removing unnecessary files, the `chat` directory should look like:

```
chat/
  __init__.py
  views.py
```

We need to tell our project that the `chat` app is installed. Edit the `mysite/settings.py` file and add `'chat'` to the `INSTALLED_APPS` setting. It'll look like this:

```
# mysite/settings.py
INSTALLED_APPS = [
    'chat',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

Add the index view

We will now create the first view, an index view that lets you type the name of a chat room to join.

Create a `templates` directory in your `chat` directory. Within the `templates` directory you have just created, create another directory called `chat`, and within that create a file called `index.html` to hold the template for the index view.

Your chat directory should now look like:

```
chat/
  __init__.py
  templates/
    chat/
      index.html
  views.py
```

Put the following code in `chat/templates/chat/index.html` :

```
<!-- chat/templates/chat/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 = '/chat/' + roomName + '/';
    };
  </script>
</body>
</html>
```

Create the view function for the room view. Put the following code in `chat/views.py` :

```
# chat/views.py
from django.shortcuts import render

def index(request):
    return render(request, "chat/index.html")
```

To call the view, we need to map it to a URL - and for this we need a URLconf.

To create a URLconf in the chat directory, create a file called `urls.py`. Your app directory should now look like:

```
chat/
  __init__.py
  templates/
    chat/
      index.html
  urls.py
  views.py
```

In the `chat/urls.py` file include the following code:

```
# chat/urls.py
from django.urls import path

from . import views

urlpatterns = [
    path("", views.index, name="index"),
]
```

The next step is to point the root URLconf at the `chat.urls` module. In `mysite/urls.py`, add an import for `django.urls.include` and insert an `include()` in the `urlpatterns` list, so you have:

```
# mysite/urls.py
from django.contrib import admin
from django.urls import include, path

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

Let's verify that the index view works. Run the following command:

```
$ python3 manage.py runserver
```

You'll see the following output on the command line:

```
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 19, 2022 - 10:05:13
```

```
Django version 4.1, using settings 'mysite.settings'
```

```
Starting development server at http://127.0.0.1:8000/
```

```
Quit the server with CONTROL-C.
```

Note

Ignore the warning about unapplied database migrations. We won't be using a database in this tutorial.

Go to <http://127.0.0.1:8000/chat/> in your browser and you should see the text “What chat room would you like to enter?” along with a text input to provide a room name.

Type in “lobby” as the room name and press enter. You should be redirected to the room view at <http://127.0.0.1:8000/chat/lobby/> but we haven't written the room view yet, so you'll get a “Page not found” error page.

Go to the terminal where you ran the `runserver` command and press Control-C to stop the server.

Integrate the Channels library

So far we've just created a regular Django app; we haven't used the Channels library at all. Now it's time to integrate Channels.

Let's start by creating a routing configuration for Channels. A Channels [routing configuration](#) is an ASGI application that is similar to a Django URLconf, in that it tells Channels what code to run when an HTTP request is received by the Channels server.

Start by adjusting the `mysite/asgi.py` file to include the following code:

```
# mysite/asgi.py
import os

from channels.routing import ProtocolTypeRouter
from django.core.asgi import get_asgi_application

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

application = ProtocolTypeRouter(
    {
        "http": get_asgi_application(),
        # Just HTTP for now. (We can add other protocols later.)
    }
)
```

Now add the Daphne library to the list of installed apps, in order to enable an ASGI versions of the `runserver` command.

Edit the `mysite/settings.py` file and add `'daphne'` to the top of the `INSTALLED_APPS` setting. It'll look like this:

```
# mysite/settings.py
INSTALLED_APPS = [
    'daphne',
    'chat',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

You'll also need to point Daphne at the root routing configuration. Edit the `mysite/settings.py` file again and add the following to the bottom of it:

```
# mysite/settings.py
# Daphne
ASGI_APPLICATION = "mysite.asgi.application"
```

With Daphne now in the installed apps, it will take control of the `runserver` command, replacing the standard Django development server with the ASGI compatible version.

 **Note**

The Daphne development server will conflict with any other third-party apps that require an overloaded or replacement runserver command. In order to solve such issues, make sure `daphne` is at the top of your `INSTALLED_APPS`, or remove the offending app altogether.

Let's ensure that the Channels development server is working correctly. Run the following command:

```
$ python3 manage.py runserver
```

You'll see the following output on the command line:

```
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 19, 2022 - 10:20:28
Django version 4.1, using settings 'mysite.settings'
Starting ASGI/Daphne version 3.0.2 development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Notice the line beginning with `Starting ASGI/Daphne ...`. This indicates that the Daphne development server has taken over from the Django development server.

Go to <http://127.0.0.1:8000/chat/> in your browser and you should still see the index page that we created before.

Go to the terminal where you ran the `runserver` command and press Control-C to stop the server.

This tutorial continues in [Tutorial 2](#).