











Project Folder Structure

When you create a Django project, it generates several important files:

-  `manage.py` → Runs and manages the whole Django server. This file contains `def main()`, which initializes the server. **(Most important file for managing your Django project!)**
-  `__init__.py` → Marks the directory as a Python package. Required for Django apps to function properly.
-  `asgi.py` → Asynchronous Server Gateway Interface. Used for handling async requests (e.g., WebSockets).
-  `wsgi.py` → Web Server Gateway Interface. Used for deploying Django applications in production.
-  `settings.py` → **Project configuration file**. Contains:
 - Secret key 
 - Installed apps 
 - Middleware settings 
 - Database configurations 
 - Allowed hosts 
-  `urls.py` → Manages URL routing for the project. It maps URLs to their corresponding views.

Step 1: Install Django

Before starting, ensure you have Django installed. If not, install it using:

```
pip install django
```

Step 2: Create a New Django Project

To create a new project, use:

```
django-admin startproject projectname
```

📌 This will generate your project folder with all necessary files.

Step 3: Navigate to Your Project Directory

Change into your project directory to access `manage.py`:

```
cd projectname
```

Step 4: Run the Django Development Server

Start the server to check if everything is working properly:

```
python manage.py runserver
```

📌 After running this command, you will see a local server running at `http://127.0.0.1:8000/`. Open it in your browser to see the Django welcome page! 🎉

Step 5: Create a Django App

Django projects are divided into **apps** to keep things modular. To create an app, use:

```
python manage.py startapp appname
```

📌 This generates an app folder with essential files like `views.py`, `models.py`, and `urls.py`.

Step 6: Register the App in `settings.py`

1. Open `settings.py`
2. Locate the `INSTALLED_APPS` section
3. Add your new app:

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'appname', # Add this line  
]
```

📌 This ensures Django recognizes your app! ✅

Step 7: Set Up URL Routing

Every Django project has a `urls.py` file that manages routing.

1. Open `projectname/urls.py`
2. Modify it to include your app's URLs:

```
from django.contrib import admin  
from django.urls import path, include # Include is needed for app  
URLs  
  
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('', include('appname.urls')), # Include app URLs  
]
```

3. Inside your **app folder**, create a new file called `urls.py` and define routes:

```
from django.urls import path  
from . import views # Import views from the app  
  
urlpatterns = [  
    path('', views.index, name='index'), # Default homepage  
]
```



Step 8: Create a Basic View

A **view** is what Django returns when a user visits a URL.

1. Open `views.py` in your app folder.
2. Define a simple function to return "Hello, World!":

```
from django.http import HttpResponse

def index(request):
    return HttpResponse("Hello, World! 🌍")
```



Now, visiting `http://127.0.0.1:8000/` will show "Hello, World!" 🎉



Step 9: Render an HTML Page

Instead of returning plain text, let's load an HTML file.

1. Inside your **app folder**, create a folder called `templates`
2. Inside `templates`, create an `index.html` file
3. Modify your `views.py` to render this HTML file:

```
from django.shortcuts import render

def index(request):
    return render(request, 'index.html') # Render index.html
```



This ensures Django loads HTML templates properly! ✅



Step 10: Apply Migrations and Set Up Database

Django uses SQLite by default, but before using it, you need to apply migrations:

```
python manage.py migrate
```

🔑 This creates necessary tables for Django's built-in features like authentication and admin.

To create your own database models, modify `models.py` inside your app folder, then run:

```
python manage.py makemigrations  
python manage.py migrate
```

🔑 This updates the database schema with your model changes! ✅

🔑 Step 11: Create a Superuser for Admin Panel

Django comes with a built-in `admin panel`. To access it, create a superuser:

```
python manage.py createsuperuser
```

🔑 Follow the prompts to set up a username, email, and password.

🔑 Now, log in at `http://127.0.0.1:8000/admin/` using the credentials you created!

✅ Step 12: Run the Server Again and Test Everything!

After setting everything up, run the server again:

```
python manage.py runserver
```

Visit `http://127.0.0.1:8000/` and test your app! 🚀



Congratulations! You've Set Up Django Successfully!

quick checklist:

- ✓ Project is created
- ✓ App is registered in `settings.py`
- ✓ URLs are set up
- ✓ Views are created
- ✓ HTML templates are rendered
- ✓ Migrations are applied
- ✓ Admin panel is accessible