**Project Directory Structure in Detail**

## **1. Typical Django Project Directory Structure**

When you create a Django project using:

django**-**admin startproject myproject

You get a structure like this:

* myproject**/**
* manage**.**py
* myproject**/**
* \_\_init\_\_**.**py
* asgi**.**py
* settings**.**py
* urls**.**py
* wsgi**.**py
* The outer myproject/ folder is the **project root**.
* The inner myproject/ folder is the **project package** containing important config files.

## **2. File-by-File Explanation**

### **A.** manage.py

**Purpose:**

* Command-line utility for running and managing your project.
* It sets the environment and calls Django commands like runserver, migrate, createsuperuser, etc.

**Default Code:**

*#!/usr/bin/env python*

**import** os

**import** sys

**def** main()**:**

    os**.**environ**.**setdefault('DJANGO\_SETTINGS\_MODULE'**,** 'myproject.settings')

**try:**

**from** django**.**core**.**management **import** execute\_from\_command\_line

**except** ImportError **as** exc**:**

**raise** ImportError("Couldn't import Django") **from** exc

    execute\_from\_command\_line(sys**.**argv)

**if** \_\_name\_\_ **==** '\_\_main\_\_'**:**

    main()

**Key Points:**

* os.environ.setdefault sets the **settings.py** file for Django.
* execute\_from\_command\_line runs any command you pass (like python manage.py runserver).

### **B.** \_\_init\_\_.py

**Purpose:**

* This is an **empty file** (by default) that tells Python that the folder is a **package**.
* Without it, Python won’t recognize the folder as a package.

**Default Code:**

# Empty file (you can leave it empty)

### **C.** settings.py

**Purpose:**

* Stores all the **configuration settings** for the project: database, installed apps, templates, static files, etc.

**Default Code Highlights:**

**import** os

**from** pathlib **import** Path

*# BASE\_DIR = project root path*

BASE\_DIR **=** Path(\_\_file\_\_)**.**resolve()**.**parent**.**parent

*# SECURITY KEY (keep secret in production)*

SECRET\_KEY **=** 'your-secret-key'

*# DEBUG mode (True in development, False in production)*

DEBUG **=** True

*# Allowed hosts (domains/IPs that can serve the project)*

ALLOWED\_HOSTS **=** []

*# Installed apps (all apps & Django default apps)*

INSTALLED\_APPS **=** [

    'django.contrib.admin'**,**

    'django.contrib.auth'**,**

    'django.contrib.contenttypes'**,**

    'django.contrib.sessions'**,**

    'django.contrib.messages'**,**

    'django.contrib.staticfiles'**,**

]

*# Middleware (functions that process requests/responses)*

MIDDLEWARE **=** [

    'django.middleware.security.SecurityMiddleware'**,**

    'django.contrib.sessions.middleware.SessionMiddleware'**,**

    'django.middleware.common.CommonMiddleware'**,**

    'django.middleware.csrf.CsrfViewMiddleware'**,**

    'django.contrib.auth.middleware.AuthenticationMiddleware'**,**

    'django.contrib.messages.middleware.MessageMiddleware'**,**

    'django.middleware.clickjacking.XFrameOptionsMiddleware'**,**

]

*# Root URL configuration*

ROOT\_URLCONF **=** 'myproject.urls'

*# Template settings*

TEMPLATES **=** [

    {

        'BACKEND'**:** 'django.template.backends.django.DjangoTemplates'**,**

        'DIRS'**:** []**,**   *# your custom templates folders*

        'APP\_DIRS'**:** True**,**  *# looks for templates in app/templates/*

        'OPTIONS'**:** {

            'context\_processors'**:** [

                'django.template.context\_processors.debug'**,**

                'django.template.context\_processors.request'**,**

                'django.contrib.auth.context\_processors.auth'**,**

                'django.contrib.messages.context\_processors.messages'**,**

            ]**,**

        }**,**

    }**,**

]

*# WSGI application*

WSGI\_APPLICATION **=** 'myproject.wsgi.application'

*# Database (default: SQLite)*

DATABASES **=** {

    'default'**:** {

        'ENGINE'**:** 'django.db.backends.sqlite3'**,**

        'NAME'**:** BASE\_DIR **/** 'db.sqlite3'**,**

    }

}

*# Password validators*

AUTH\_PASSWORD\_VALIDATORS **=** [

    {'NAME'**:** 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator'}**,**

    {'NAME'**:** 'django.contrib.auth.password\_validation.MinimumLengthValidator'}**,**

    {'NAME'**:** 'django.contrib.auth.password\_validation.CommonPasswordValidator'}**,**

    {'NAME'**:** 'django.contrib.auth.password\_validation.NumericPasswordValidator'}**,**

]

*# Static files (CSS, JS, images)*

STATIC\_URL **=** 'static/'

*# Default primary key field type*

DEFAULT\_AUTO\_FIELD **=** 'django.db.models.BigAutoField'

**Key Points:**

* You configure **apps, database, templates, middleware, static/media files** here.
* BASE\_DIR is very important for building paths.

### **D.** urls.py

**Purpose:**

* This is the **URL dispatcher (router)** of your project.
* It maps URL paths to **views**.

**Default Code:**

**from** django**.**contrib **import** admin

**from** django**.**urls **import** path

urlpatterns **=** [

    path('admin/'**,** admin**.**site**.**urls)**,**

]

**Key Points:**

* You define urlpatterns as a list of all URL patterns.
* Each path() maps a URL to a view (or to other URL configs from apps).

### **E.** wsgi.py

**Purpose:**

* Entry point for **WSGI-compatible web servers** (like Gunicorn) to serve your Django app in production.
* Used for **synchronous requests**.

**Default Code:**

**import** os

**from** django**.**core**.**wsgi **import** get\_wsgi\_application

os**.**environ**.**setdefault('DJANGO\_SETTINGS\_MODULE'**,** 'myproject.settings')

application **=** get\_wsgi\_application()

**Key Points:**

* get\_wsgi\_application() loads Django and makes it ready for the web server.
* Used when deploying on **Apache / Gunicorn / Nginx**.

### **F.** asgi.py

**Purpose:**

* Entry point for **ASGI-compatible servers** (like Daphne, Uvicorn).
* Supports **asynchronous requests** (WebSockets, background tasks, etc.).

**Default Code:**

**import** os

**from** django**.**core**.**asgi **import** get\_asgi\_application

os**.**environ**.**setdefault('DJANGO\_SETTINGS\_MODULE'**,** 'myproject.settings')

application **=** get\_asgi\_application()

**Key Points:**

* Similar to wsgi.py but supports async features.
* Used for real-time apps (chat, notifications, etc.).

## **3. Flow Summary**

1. User sends a request → urls.py (routes it).
2. urls.py → calls a **view** from apps.
3. Views use **models** & return a **template**.
4. wsgi.py / asgi.py makes the project compatible with the web server.

## **1. Full Forms**

* **WSGI**: **Web Server Gateway Interface**
* **ASGI**: **Asynchronous Server Gateway Interface**

## **2. Purpose**

Both **WSGI** and **ASGI** are interfaces (protocols) that connect **web servers** (like Gunicorn, Uvicorn, Apache) with your **Django application**.

* They act as a **bridge** between the web server and Django.
* Without them, the server wouldn’t know how to handle Django requests.

## **3. Difference Between WSGI and ASGI**

| **Feature** | **WSGI (Web Server Gateway Interface)** | **ASGI (Asynchronous Server Gateway Interface)** |
| --- | --- | --- |
| **Nature** | Synchronous (one request at a time per worker) | Asynchronous + Synchronous (can handle multiple tasks) |
| **Requests** | Blocks until request is complete | Non-blocking, can handle multiple requests concurrently |
| **WebSockets** | ❌ Does not support WebSockets or long-lived connections | ✅ Supports WebSockets, HTTP2, and other real-time protocols |
| **Use Case** | Standard web apps (normal websites, APIs) | Real-time apps (chat, notifications, async APIs) |
| **Performance** | Good for normal apps | Better for high concurrency and real-time apps |
| **Example Servers** | Gunicorn, mod\_wsgi (Apache) | Daphne, Uvicorn, Hypercorn |

## **4. Why Django Uses Both?**

* **WSGI**: Traditional Django apps were synchronous. Django still supports it for backward compatibility.
* **ASGI**: Django (from version 3.0+) added support for **asynchronous features** like WebSockets and background tasks.

## **5. How They Work in Django**

* **wsgi.py** → Used in production with a **WSGI server**:

python

CopyEdit

from django.core.wsgi import get\_wsgi\_application

application = get\_wsgi\_application()

* + Handles normal HTTP requests.
* **asgi.py** → Used with an **ASGI server** (like Uvicorn):

python

CopyEdit

from django.core.asgi import get\_asgi\_application

application = get\_asgi\_application()

* + Handles both HTTP requests and async protocols (WebSockets).

## **6. When to Use Which?**

* If your app is a **regular website or REST API**, WSGI is fine.
* If you need **real-time features** (chat, live notifications, async tasks), you need ASGI.