Setting Up Django

 This concept page provides a step-by-step guide on setting up a new Django project and understanding the project structure. It covers the installation process, creating a new project, and exploring the essential files and directories within a Django project.

Concept Overview

Topics

* Installing Django
* Creating a New Project
* Project Structure
* Django Apps
* Running a Django App

Learning Objectives

* Install Django on your development environment
* Create and run a new Django project using the command-line utility
* Understand the purpose and structure of the essential files and directories within a Django project

Installing Django

Setting up a Django project is a crucial first step in building a web application. Django provides a command-line utility to create a new project with a predefined structure. Understanding the project structure and the role of each component is essential for efficient development.

*As Django is a Python web framework, it requires Python. Hence you must first install Python if you have not yet done so. In addition checkout*[*here*](https://savanna.alxafrica.com/rltoken/miaYrsd2pqzfZ6hFIvqh_w)*to see what Python versions can be used with Django.*

Before creating a new Django project, you need to install Django on your development environment. You can install Django using pip, the Python package installer. Open your command prompt or terminal and run the following command to install the latest version of Django:

pip install django

Once the installation is complete, you can proceed to create a new Django project.

Creating a New Project

Django provides a command-line utility called django-admin to create a new project.  
To create a new project navigate to the directory where you want to create your project and execute the below command by replacing project\_name with the desired name for your project.

django-admin startproject project\_name

This command will create a new directory with the project name and several files and directories inside it.

Project Structure

After creating a new Django project, you’ll find the following essential files and directories:

project\_name/

manage.py

mysite/

\_\_init\_\_.py

settings.py

urls.py

asgi.py

wsgi.py

* project\_name/: The root directory of your project.
* manage.py: A command-line utility for managing your Django project.
* project\_name/: A Python package directory with project-specific settings and configurations.
* \_\_init\_\_.py: An empty file that tells Python that this directory should be considered a Python package.
* settings.py: This file contains the project’s settings and configurations, such as database settings, installed apps, and static file settings.
* urls.py: This file defines the URL patterns for the project and maps them to the appropriate views.
* asgi.py: An entry point for ASGI-compatible web servers to serve the project.
* wsgi.py: An entry point for WSGI-compatible web servers to serve the project.

Running a Django Project

After setting up a new Django project, you’ll need to run the development server to see your application in action. Django provides a built-in development server that allows you to test and debug your application locally before deploying it to a production environment.

To run the Django development server, first navigate to the root directory of your Django project (the directory containing the manage.py file) and run the following command (This command will start the Django development server.):

python manage.py runserver

Once the server is running, you should see output similar to the following:

Performing system checks...

System check identified no issues (0 silenced).

March 25, 2024 - 15:50:53

Django version 4.1, using settings 'myproject.settings'

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

Quit the server with CTRL-BREAK.

Then open your web browser and visit http://127.0.0.1:8000/ to see the default Django welcome page.

**Note that the development server is designed for local development and testing purposes only. It should not be used in a production environment.**

When you’re ready to deploy your Django application, you’ll need to set up a production-ready web server like Apache or Nginx, along with a WSGI server like Gunicorn or uWSGI.

Django Apps

In the Django web framework, a project and an app are related but distinct concepts. A Django project is a collection of settings and configurations for a particular Django web application. It acts as a container for one or more Django apps. It defines the database settings, installed apps, middleware, templates, and other project-level configurations.

On the hand, an **app** in Django is a self-contained, reusable module that represents a specific functionality or feature of your web application. A Django project can have multiple apps, each responsible for a specific aspect of the application. Apps contain models (database schemas), views (handling HTTP requests and responses), templates (HTML files), and other app-specific files. Examples of apps might include a blog app, a user authentication app, an e-commerce app, etc.

Creating Django Apps

You create a new app within a project using the following command

python manage.py startapp book\_store

When you create a new Django app using the above command, Django generates several files within the app directory.

book\_store/

\_\_init\_\_.py

admin.py

apps.py

migrations/

\_\_init\_\_.py

models.py

tests.py

views.py

1. admin.py : This file is used to register your models with the Django admin interface, which provides a user-friendly way to manage your application’s data through a web interface.
2. apps.py: This file is used to define the configuration and metadata for your app. It contains a Config class that inherits from django.apps.AppConfig and includes metadata such as the app name and label.
3. migrations/: This directory is created the first time you run migrations for your app. It stores migration files that keep track of changes to your models, allowing you to evolve your database schema over time.
4. models.py: This file is where you define your data models, which represent the database tables for your application. Models are defined as Python classes that inherit from django.db.models.Model
5. tests.py: This file is used to write unit tests for your app’s models, views, and other components. Django provides a built-in testing framework to help you write and run tests.
6. views.py: This file contains the view functions that handle HTTP requests and return HTTP responses. Views are responsible for processing user input, interacting with models, and rendering templates.

Setting Up Your Django App

Adding A simple view : Open the file book\_store/views.py and add the following lines “` from django.http import HttpResponse

def index(request): return HttpResponse("Welcome to my book store.”) “`

This is the simplest view possible in Django. To call the view, we need to map it to a URL - and for this we need a URLconf. Hence, to create a URLconf in the book\_store directory, create a file called urls.py and update it to contain the following code.

from django.urls import path

from . import views

urlpatterns = [

path("", views.index, name="index"),

]

Next, lets update the root URLconf to include our updated route. Open the project\_name/urls.py and update it according to the below code

from django.contrib import admin

from django.urls import include, path

urlpatterns = [

path("books/", include("book\_store.urls")),

path("admin/", admin.site.urls),

]

The next step is to register it to our project so that it will be included when any tools are run. We do this by adding them to the INSTALLED\_APPS list found in settings.py file

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'book\_store.apps.BookStoreConfig',

]

Migrating and Running Django App

Now we need to migrate our changes. To do this run the following two commands one after the other

python manage.py makemigrations

python manage.py migrate

Finally run the project using python manage.py runserver and go to http://localhost:8000/books. You should be able to see the the welcome message.

Running Django project on a specific port

By default, the development server runs on http://127.0.0.1:8000/, but you can specify a different host and port using the following command:

python manage.py runserver [port]

For example, to run the server on http://localhost:8080/, use:

python manage.py runserver 8080

Practice Exercises

* Install Django on your development environment (if you haven’t already).
* Create a new Django project named "myproject” and run it
* Explore the project structure and identify the purpose of each file and directory.

Additional Resources

* [Django Installation Guide](https://savanna.alxafrica.com/rltoken/JFTzyOUk_FmXGO25WBaCog)
* [Writing your first Django app](https://savanna.alxafrica.com/rltoken/8U91FnfGR-TFfV5RuK7TnA)