App structures

A Django project is a web application that may consist of one or more sub-modules called apps.

In this reading, you'll learn about the app structure, how to create an app and how to configure project settings to include the app.

What is a Django app?

As mentioned above, an app is responsible for performing one single task out of the many involved in the complete web application, represented by the Django project.

It implies that a project comprises many independent sub-applications, yet they may communicate among themselves.

For example, a trading organization website may have one app for managing customer data, another for suppliers, and another for stock management. However, the important feature of the Django app is that it is reusable.

Hence, a customer data management app in one project can be linked to the website project for another organization without modification.

For simplicity, you are going to uniformly use one app inside one project throughout the course, but it is good to know the utility of Django framework as a multi-app framework.

When a Django project is created with the **`startproject`** command, it creates a container folder. Django puts a **`manage.py`** script and the project package folder in the outer folder.

The **`startapp`** command option of the **`manage.py`** script creates a default folder structure for the app of that name.

Here's how to create a **`demoapp`** in the **`demoproject`** folder.

```
(djenv) C:\djenv\demoproject>python manage.py startapp demoapp
```

A folder with the app's name is created inside the parent folder. It has a few Python scripts. The folder structure looks like this:

```
C:\djenv\demoproject
|   db.sqlite3
|   manage.py
|
├──demoapp
|   |   admin.py
|   |   apps.py
|   |   models.py
|   |   tests.py
|   |   views.py
|   |   __init__.py
|   |
|   └──migrations
|           __init__.py
|
└──demoproject
    |   asgi.py
    |   settings.py
    |   urls.py
    |   wsgi.py
    |   __init__.py
```

Let's briefly explain the Python scriptscreated in the **`demoapp`** folder.

views.py

In Django, a view is a user-defined function that's called when Django's URL dispatcher identifies the client's request URL and matches it with a URL pattern defined in the urls.py file.

The auto-created views file is empty at the beginning.

Let's add a view function called index() in it by saving the following snippet.

```
from django.http import HttpResponse
def index(request):
    return HttpResponse("Hello, world. This is the index view of Demoapp.")
```

urls.py
The project package has a file of this name that defines the URL patterns for the project.

On similar lines, you need to provide the URL routing mechanism for the app.

One important thing to note here in contrast to the video you have seen earlier is that the `urls.py` file can be configured at both the project and app level. In the example below, the urls.py will be configured at both the project and app-level.
The app folder doesn't have a file of this name when created. Hence, you have to create one.

Save the following snippet as **`urls.py`** in the **demoapp** folder.

```
from django.urls import path
from . import views

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

Next, you need to update the **`urlpatterns`** list in the project folder's urls.py and include the app's URL configurations.
The updated **`demoproject/urls.py`** should look like this:

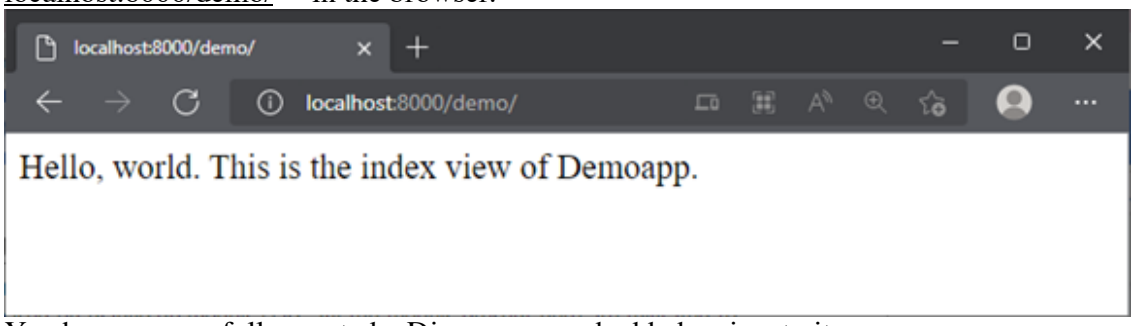
```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('demo/', include('demoapp.urls')),
    path('admin/', admin.site.urls),
]
```

models.py
The data models required for processing in this app are created in this file. It is empty by default. A data model is a Python class based on **`django.db.modelsclass`**.All the models present here are migrated to the database tables. For now, leave his file as it is without adding any models. You will learn how to work with models later.
tests.py
You'll write the tests to be run on the app in this file. Let's keep this file as it is without modifying it.

Update settings.py
Lastly, you need to update the list of INSTALLED_APPS in the project's setting file. This list already contains some pre-installed apps. Add the name of **`demoapp`** so that it looks like this:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'demoapp',
]
```

After Performing the steps explained above, run the Django development server and visit localhost:8000/demo/ in the browser.



You have successfully created a Django app and added a view to it.

Note here that the files present in the app structure are not limited to this reading. The app directory as you will see also consists of other files such as **forms.py**, **serializers.py** among others.
In this reading, you learned to create an app, update the project settings to include it and add a view function.

Mark as completed

Like Dislike Report an issue