Django

16 July 2021 22:55

Python Backend Web Development Course (with Django)
Following : Python Backend Web Development Course (with Django)
From : https://www.freecodecamp.org/news/backend-web-development-with-python-full-course/amp/

Python also support **try, except, else** and **try, except, finally** :-

```
Try :
        ……
except:
        ……
Else :
        ……
```

Simple Inheritance :-

```
class Student():
   name = 'RJ'
   age = 20
   gender = 'male'

class Person(Student):
   pass

p1 = Person()
print(p1.name)
```

Django is a Python Web Framework. We can use Python to build web applications with Django.

Steps :-

1) Install Python
2) Pip install django
3) Pip install virtualenvwrapper-win (installing virtual environment package)
4) Mkvirtualenv myapp (creating virtual environment named myapp)
5) Django-admin startproject myproject (creating new django project named myproject)
6) Deactivate (will exit from virtual environment)
7) Workon myapp (will activate virtual environment again)

Django App is different from Django Project. Different apps combination makes a project.

To create an App in Django Project :-

8) Python manage.py startapp myapp (should be in same directory where we add created django project i.e. Step 5. This will create a folder in myproject named myapp with bunched of files).
9) python manage.py runserver (to run the django server) will print only django template yet
10) Add html files in new folder named templates and add it in settings.py in TEMPLATES = […]
    - 'DIRS': [BASE_DIR/'templates'],
    - Add myapp in settings.py in INSTALLED_APPS = [….]
11) Now we need to render our html pages :-
    a. Create a new file name urls.py in myapp folder and add this :-

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

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

    b. Now add a new function named index in views.py in myapp as :-

```
from django.shortcuts import render

# Create your views here.
def index(request):
    return render(request, 'index.html')
```

c. Now we need to add the url that we created in myapp in (12.a) in our main project. Open urls.py of myproject and add it like this :-

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

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

12) python manage.py runserver (to run the django server) it will run and print the data of index.html

By Default, if we don't specify the method, it will be treated as GET method.

CSRF token require to counter the attacks.
Use this : {% csrf_token %}

Template files are the HTML file we use in Django.
Any External fine we use in our **Static Files.** Like we have an external CSS file that is linking to the HTML file, that is a static file. An image or external video all those are static files.

=> First we need to make migrations then migrate.

1) python manage.py makemigrations // any changes we have made in the modules file (like in models file, etc), will save that changes.
2) Python manage.py migrate // to send all this changes into our database

Need to run above 2 commands every time we make any changes in module files like models.py

Command for Creating a superuser to login on http://127.0.0.1:8000/admin

Python manage.py createsuperuser

How to Make a new page and link it :-

1) Add url in urls.py
```
urlpatterns = [
    path('', views.index, name='index'),
    path('register', views.register, name='register'),
    path('login', views.login, name='login')
]
```

2) Write function in views.py
```
def login(request):
    return render(request, 'login.html')
```

3) Write html file (in template folder) (e.g. : login.html)

Connecting Postgress Database (Postgresql) to Django Project :-

1) Download and Install postgreSQL
2) Download and install pgAdmin
3) Run pgAdmin and create new database in Servers -> PostgreSQL -> Databases and name it anything like myproject

4) Go back to settings.py in myproject folder and changes DATABASES = { …… } as followes

```
DATABASES = {
    'default': {
        # 'ENGINE': 'django.db.backends.sqlite3',
        'ENGINE': 'django.db.backends.postgresql',
        'NAME':  'myproject',
        'USER': 'postgres',
        'PASSWORD' : 'password',
        'HOST': 'localhost'
    }
}
```

5) pip install psycopg2 and pip install Pillow
6) Run python manage.py makemigrations in project folder (will return No changes detected)
7) Run python manage.py migrate (this will create all tables in pgAdmin -> Servers -> PostgreSQL -> Databases -> myproject -> Schemas -> Tables) Right click on a table and view it's data.
8) In this way we can connect any database to django project (It's uses by default SQLite)

For creating a Database in django we first create models:-

1) Create Database in models.py file as :-

```
from django.db import models
from datetime import datetime

# Create your models here.
class Room(models.Model):
    name = models.CharField(max_length=1000)

class Message(models.Model):
    value = models.CharField(max_length=1000000)
    date = models.DateTimeField(default=datetime.now, blank=True)
    user = models.CharField(max_length=1000000)
    room = models.CharField(max_length=1000000)
```

2) Now run python manage.py makemigrations
3) Run python manage.py migrate (need to run these 2 commands every time we change anything in models.py
4) Create a super user using python manage.py createsuperuser. Its used to login into http://127.0.0.1:8000/admin/
5) Need to register the models in admin.py as followes :-

```
from django.contrib import admin
from .models import Room, Message

# Register your models here.s
admin.site.regiter(Room)
admin.site.register(Message)
```

Django Rest Framework :- is a library which allow us to build API's in our django project

1) First install django using : pip install django
2) Create new project named restframework using : django-admin startproject restframework
3) Cd restframework
4) Install django rest framework using : pip install djangorestframework
5) Open settings.py and add : 'rest_framework' in INSTALLED_APPS = […]
6) Open urls.py and add : path('api-auth/', include('rest_framework.urls')), in urlpatterns = […]

Serializers in Django Rest Framework :-
Serializers is a structure/representation that represents a data, we want to return in a JSON format, or are saved in JSON format.

Authentication in Django Rest Framework :-
It allow us to protect our API endpoint. The user may require an API key or a token or something before he can access that particular API.

How to do it in your project :-

1) from rest_framework.permissions import IsAuthenticated (write this in views.py of django project)
2) Add this in views.py of django project

```
class TesView(APIView):

    permission_classes = (IsAuthenticated, )
    ……
```

3) Open settings.py and add this in the end :

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES' : [
        'rest_framework.authentication.TokenAuthentication'
    ]
}
```

Add this also :-

```
INSTALLED_APPS = [
  …….
    'rest_framework.authtoken'
    …….
]
```

4) Run python manage.py makemigrations and python manage.py migrate
5) Flush up all the data that is in our database using :- (will delete all the database)
    a. python manage.py flush and enter yes
    b. Need to create superuser again using :-
        i. python manage.py createsuperuser
6) Now open http://127.0.0.1:8000/admin/ we will have a new section named AUTH TOKEN with Tokens sub category.
7) Now we try to get or post anything we will get this response :-

```
{
    "detail": "Authentication credentials were not provided."
}
```

The user will need a token to access that particular API.

8) To create a token using command line :

```
python manage.py drf_create_token username
```

Ex : -

```
python manage.py drf_create_token rj
```

Will return this : Generated token c36baf2fc2f21d33ed647e0297c075a8876b2506 for user rj

9) Now we can go to postman -> in Get -> Authorization and write :-
    Key : Authorization
    Value : Token c36baf2fc2f21d33ed647e0297c075a8876b2506
    Add to : header

Now when we call the get method it will return the data :-

```
{
    "name": "",
    "age": null
}
```

10) Now we want to automate, so that every user will get a token automatically.
11) Open urls.py of the project and add :-

```
from rest_framework.authtoken.views import obtain_auth_token
```

We don't need to create a new view in views. This is default, built in view, which if from the rest framework. It allows us to send user's credentials, and then obtains the authentications token for us.

Now add this in urlpatterns = [….

```
    path('api/token', obtain_auth_token, name='obtain')
….
]
```

12) Now got to postman -> POST -> write this http://127.0.0.1:8000/api/token
    Select Body -> form-data

Write this :-

Username : rj
Password : password

Click on Send and it will return the token as :-


```
{
   "token": "c36baf2fc2f21d33ed647e0297c075a8876b2506"
}
```