

Optional: Create a Standalone Django ORM Project Template



Estimated time needed: 20 minutes

Learning Objectives

- Create a standalone Django ORM project and app
- Save this project as a template to learn and build more complex Django ORM apps

Start PostgreSQL in Theia

PostgreSQL, also known as Postgres, is an open-source relational database management system and it is one of the main databases Django uses.

If you are using the Theia environment hosted by [Skills Network Labs](#), a pre-installed PostgreSQL instance is provided for you which can be started with one simple command line.

You can skip this step if you have already started it in previous labs.

- If the terminal was not open, go to Terminal > New Terminal and run:

```
1. 1
1. start_postgres
```

Copied!

- Once PostgreSQL is started, you can check the server connection information in the terminal. You need to save the connection information such as generated username, password, and host, etc, which will be used to configure the Django app to connect to this database.

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10

1. Starting your Postgres database....
2. This process can take up to a minute.
3.
4. Postgres database started, waiting for all services to be ready....
5.
6. Your Postgres database is now ready to use and available with username: postgres password: Nzg3Mi15bHVvLTIZ
7.
8. You can access your Postgres database via:
9.   • The browser at: https://yluo-5050.theiadocker-1.proxy.cognitiveclass.ai
10.  • CommandLine: psql --username=postgres --host=localhost
```

Copied!

- Django needs an adapter called Psycopg as an interface to work with PostgreSQL, you can install it using the following command:

```
1. 1
1. python3 -m pip install psycopg2-binary
```

Copied!

Create a Simplified Django ORM Project

Before starting the lab, make sure your current Theia directory is `/home/project`.

First, we need to install Django related packages if not installed previously.

- Go to terminal and run:

1. 1

1. python3 -m pip install Django

Copied!

In this lab, instead of creating a complete Django web project and app using command-line utilities, you will be creating a simplified app with only ORM component from scratch.

Within the ORM-only app, you can define your models and easily perform CRUD operations on your model objects.

More importantly, you can create and run Python script files to do that instead of typing Python code into shell line by line.

Let's start by creating an empty project folder

- In the Theia menu, click **File > New Folder** and type `ormtemplate` which acts as the container folder for an empty Django project.
- Right-click the `ormtemplate` folder, and create a subfolder `standalone` which acts as the container folder for an empty Django app called `standalone`
- Right-click the `ormtemplate` folder again, and create an empty `settings.py` file and a `manage.py` file

Next, we will add the content to `manage.py` file acting as a command-line interface managing our Django project

- Open the empty `manage.py` file, and copy and paste the following code snippet

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18

1. import os
2. import sys
3.
4. if __name__ == "__main__":
5.     os.environ.setdefault("DJANGO_SETTINGS_MODULE", "settings")
6.     try:
7.         from django.core.management import execute_from_command_line
8.     except ImportError:
9.         try:
10.             import django
11.         except ImportError:
12.             raise ImportError(
13.                 "Couldn't import Django. Are you sure it's installed and "
14.                 "available on your PYTHONPATH environment variable? Did you "
15.                 "forget to activate a virtual environment?"
16.             )
17.         raise
18.     execute_from_command_line(sys.argv)
```

Copied!

The code snippet be added to the `manage.py` file as the setting module of our `ormtemplate` project and be able to execute Django built-in commands such as migrations.

Next, let's add a simple database settings to `settings.py`

- Open the empty `settings.py`, copy and paste the following code snippet

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
```

```

9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17

1. # PostgreSQL
2. DATABASES = {
3.     'default': {
4.         'ENGINE': 'django.db.backends.postgresql_psycopg2',
5.         'NAME': 'postgres',
6.         'USER': 'postgres',
7.         'PASSWORD': 'Place with your password generated in Step 1',
8.         'HOST': 'localhost',
9.         'PORT': '5432',
10.     }
11. }
12.
13. INSTALLED_APPS = (
14.     'standalone',
15. )
16.
17. SECRET_KEY = 'SECRET KEY for this Django Project'

```

Copied!

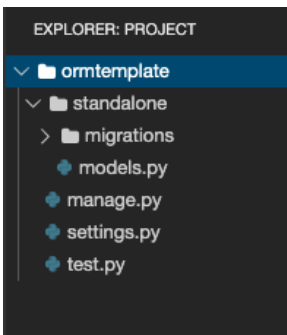
The above code snippet adds standalone app as an installed app and adds the default database to be the pre-installed PostgreSQL we created in Step 1.

You just need to update the PASSWORD field to be the password generated in Step 1.

So far we have created a very simple Django project. Next let's add content to our standalone app.

- Click ormtemplate/standalone folder and create an empty models.py containing model definitions and a folder named migrations containing migration scripts in standalone app.

After that, your app structure should look like the following:



- Now your Django standalone app is ready and you can start test if the standalone app is working.

Test the standalone App

- Open orm/models.py file and copy and paste the following code snippet to define a simple Test model

```

1. 1
2. 2
3. 3
4. 4
5. 5

1. from django.db import models
2.
3. # Test model
4. class Test(models.Model):
5.     name = models.CharField(max_length=30)

```

Copied!

Next, we can ask ormtemplate app to generate the Test table by running migration command-lines.

- cd into ormtemplate folder

```
1. 1
1. cd ormtemplate
```

Copied!

and now your current Theia folder should be /home/project/ormtemplate shown in the terminal

- Then, you need to generate migration scripts for standalone app

```
1. 1
1. python3 manage.py makemigrations standalone
```

Copied!

and you should see migration scripts generated for your Test model

```
1. 1
2. 2
3. 3

1. Migrations for 'standalone':
2.   standalone/migrations/0001_initial.py
3.   - Create model Test
```

Copied!

Note, if you see errors like
django.db.utils.OperationalError: FATAL: password authentication failed for user "postgres"
please re-run start_postgres to reset the PostgreSQL server and use the new password in settings.py.

- and run migration

```
1. 1
1. python3 manage.py migrate
```

Copied!

Next, you can write some Python testing code in a Python script file (*.py) to test your model.

- Click the ormtemplate folder and create a new file called test.py
- Open the empty test.py and add following code snippet to test your test model:

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
23. 23
24. 24
25. 25
26. 26
27. 27
28. 28
29. 29
30. 30
31. 31

1. # Django specific settings
2. import inspect
3. import os
4. os.environ.setdefault("DJANGO_SETTINGS_MODULE", "settings")
5. from django.db import connection
6. # Ensure settings are read
7. from django.core.wsgi import get_wsgi_application
8. application = get_wsgi_application()
```

```

9. # Your application specific imports
10. from standalone.models import Test
11.
12. # Delete all data
13. def clean_data():
14.     Test.objects.all().delete()
15.
16. # Test Django Model Setup
17. def test_setup():
18.     try:
19.         clean_data()
20.         test = Test(name="name")
21.         test.save()
22.         # Check test table is not empty
23.         assert Test.objects.count() > 0
24.         print("Django Model setup completed.")
25.     except AssertionError as exception:
26.         print("Django Model setup failed with error: ")
27.         raise(exception)
28.     except:
29.         print("Unexpected error")
30.
31. test_setup()

```

Copied!

The above code snippet first cleans the database and then inserts a test object. Then it checks if the test object was inserted correctly.

- At last, in the terminal, run the test.py:

```

1. 1
1. python3 test.py

```

Copied!

and you should see

```

1. 1
1. Django Model setup completed.

```

Copied!

Now you have successfully created a standalone Django ORM app and tested it with a simple Test model. You also get yourself familiar with Django app structure by creating them from scratch.

Next, you could download and save this project locally as a template for your future learning and Django ORM development activities.

- Right-click the root folder ormentemplate, and click Download to save your workspace.

For your practice, you could import it to a new Theia environment or your local Python environment as a starting point to develop more complex Django ORM models.

Summary

In this lab, you have created a standalone Django ORM app without creating a full Django web project. You could use this simple ORM app as a template for you to learn Django ORM as well as build more complex Django ORM apps in the future.

Author(s)

[Yan Luo]([linkedin.com/in/yan-luo-96288783](https://www.linkedin.com/in/yan-luo-96288783))

Changelog

Date	Version	Changed by	Change Description
30-Nov-2020	1.0	Yan Luo	Initial version created

© IBM Corporation 2020. All rights reserved.