Django Admin site

**Estimated time needed:** 20 minutes

Learning Objectives

- Understand the concepts of Django admin site
- Create and customize your Django admin site to manage the content of an `onlinecourse` app
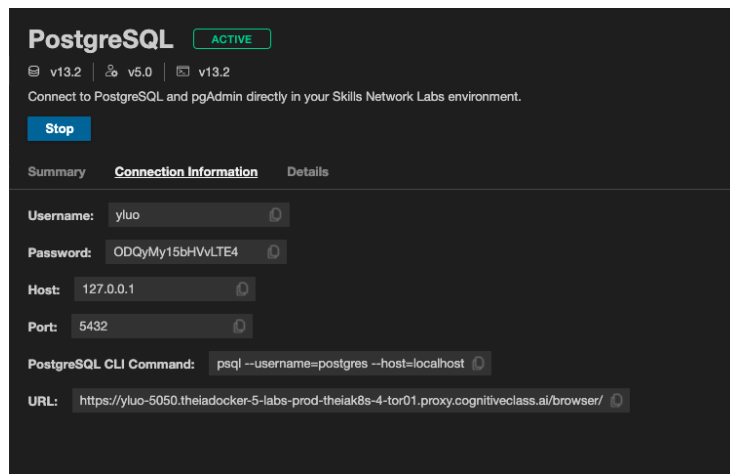
Start PostgreSQL in Theia

`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 <u>Skills Network Labs</u>, a pre-installed PostgreSQL instance is provided for you.

You can start PostgreSQL from UI by finding the SkillsNetwork icon on the left menu bar and selecting PostgreSQL from the DATABASES menu item:
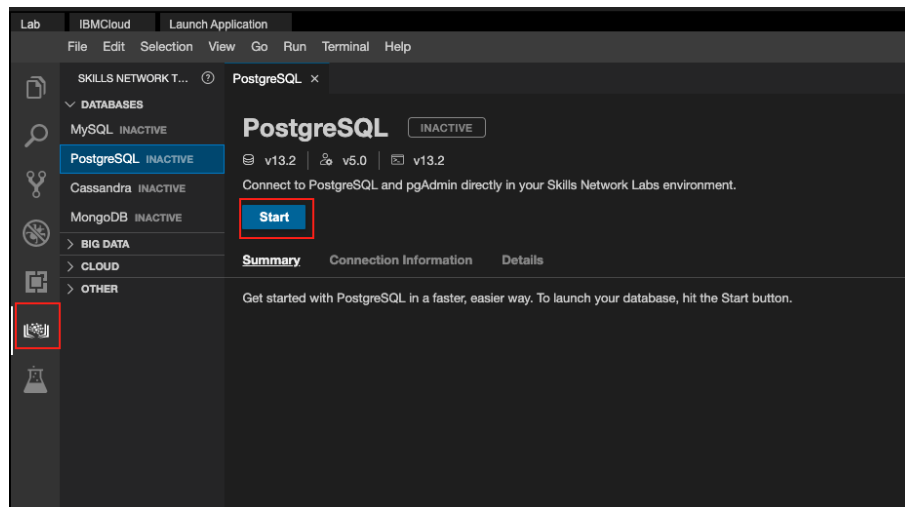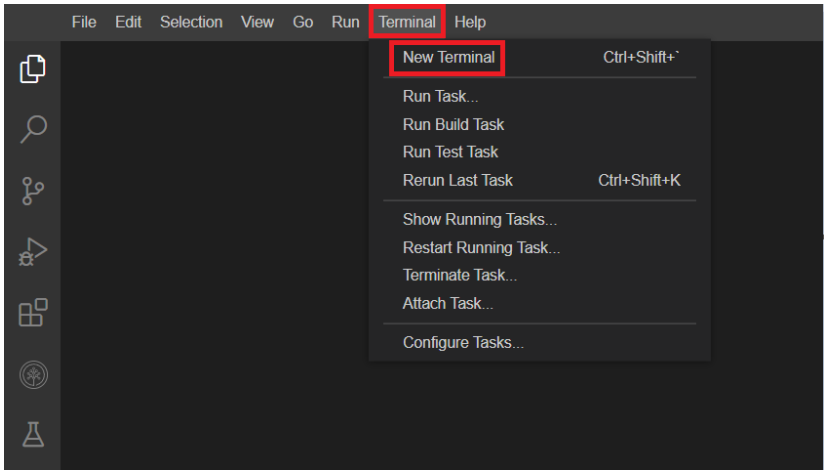

Once the PostgreSQL has been started, you can check the server connection information from the UI.
Please markdown the connection information such as generated `username`, `password`, and `host`, etc, which will be used to configure Django app to connect to this database.

Optional: Start PostgreSQL from terminal

You may also start PostgreSQL from terminal:

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 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
```
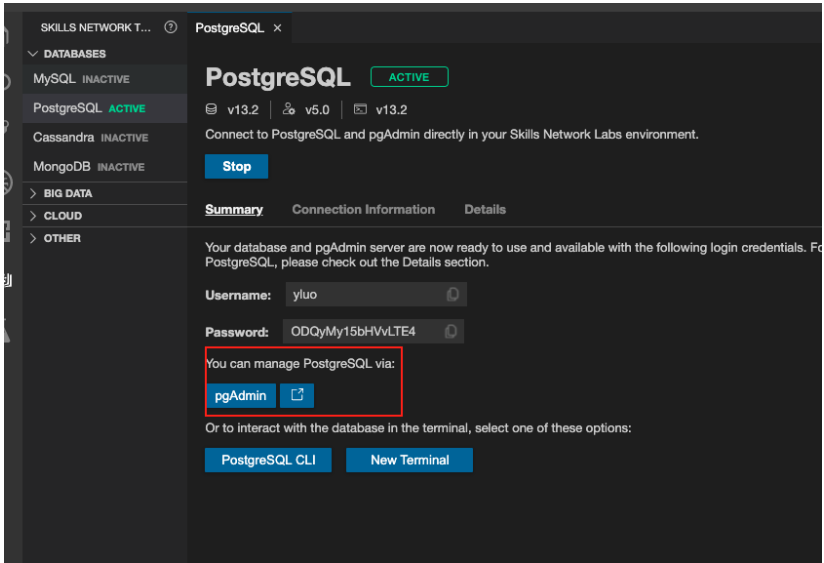
Copied!

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

1. 1

1. python3 -m pip install psycopg2-binary

Copied!

Also, a `pgAdmin` instance is installed and started for you. It is a popular PostgreSQL admin and development tool for you to manage your PostgreSQL server interactively.

Import an `onlinecourse` App Template

If the terminal was not open, go to `Terminal > New Terminal` and make sure your current Theia directory is `/home/project`.
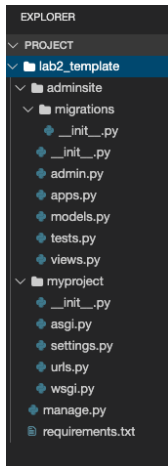
- Run the following command-lines to download a code template for this lab

1. 1
2. 2

3. 3

```
1. wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-CD0251EN-SkillsNetwork/labs/m4_django_app/lab2_template.zip"
2. unzip lab2_template.zip
3. rm lab2_template.zip
```

Copied!

Your Django project should look like the following:

Next, we need to set up a proper runtime environment for Django app.

- If the terminal was not open, go to `Terminal > New Terminal` and `cd` to the project folder

```
1. 1
```

```
1. cd lab2_template
```

Copied!

```
1. 1
```

```
1. python3 -m pip install -U -r requirements.txt
```

Copied!

The `requirements.txt` contains all necessary Python packages for you to run this lab.

- Open `settings.py` and find `DATABASES` section, and replace the value of `PASSWORD` with the one generated by `PostgreSQL` password.

Next, activate the models for an `onlinecourse` app which will be managed by admin site.

- You need to perform migrations to create necessary tables

```
1. 1
```

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

Copied!

- and run migration

```
1. 1
```

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

Copied!

Create a Superuser for Admin Site

Before you can access admin site, you will need to create a super user for the admin site.

- run

```
1. 1
```

```
1. python3 manage.py createsuperuser
```

Copied!

With `Username`, `Email`, `Password` entered, you should see a message indicates the super user is created:

```
1. 1
```

```
1. Superuser created successfully.
```
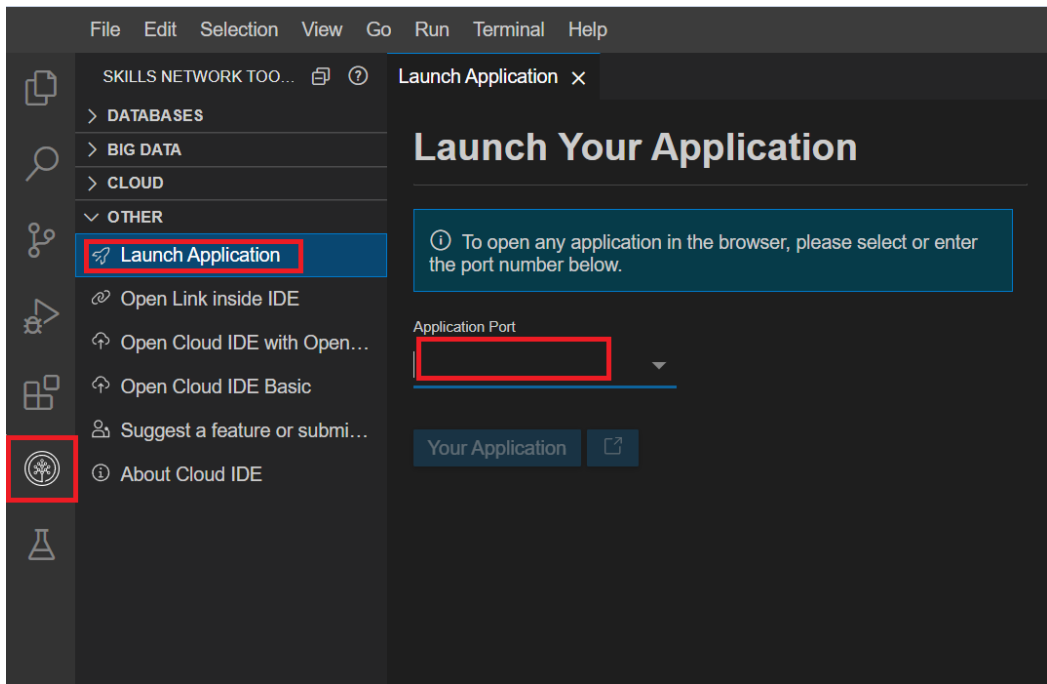
Copied!

Let's start our app and login with the super user.

- Then start the development server

```
1. 1
```

```
1. python3 manage.py runserver
```

Copied!

- Click on the Skills Network button on the left, it will open the **"Skills Network Toolbox"**. Then click the **Other** then **Launch Application**. From there you should be able to enter the port `8000` and launch.

When the browser tab opens, add the `/admin` path and your full URL should look like the following

```
https://userid-8000.theiadocker-1.proxy.cognitiveclass.ai/admin
```

- Login with the user name and password for the super user, then you should see admin site with only `Groups` and `Users` entries created for us. These two tables are created by Django by default for authentication and authorization purposes.

Register Models with Admin site

Once you register models defined in `adminsite/models.py` with admin site, you can then create managing pages for those models.

- Open `adminsite/admin.py`, and register `Course` and `Instructor` models

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

```
1.
2. admin.site.register(Course)
3. admin.site.register(Instructor)
```

Copied!

Now refresh the admin site, and you should be able to see `Courses` and `Instructors` under `ADMINSITE` section.

Let's create an instructor first. For simplification, we can assign the super user to be an instructor

- Click the green `Add` button in the `Instructors` row

- For the `User` field, choose the super user you just created:

Add instructor

User:
root   ✏️ ➕

☑ Full time

Total learners: 0

For fields in `Instructor` model, Django automatically creates corresponding web widgets for us to receive their values. For example, a checkbox is created for `Full Time` field and numeric input field for `Total learners`.

- Then choose if the instructor to be `Full Time` or not and enter 0 for `Total learners`.

- Similarly, you can create a course by entering its `Name`, `Description`, `Publication date`, etc. For the `Image` field, we defined an `ImageField` in `Course` model, and then Django admin site app automatically adds image uploading functionality for us.

Add course

Name:

online course

Image:

[Choose File] django.png

Description:

online course description

Pub date:

2020-11-29   Today | 📅

Note: You are 5 hours behind server time.

Instructors:

root

+

Hold down "Control", or "Command" on a Mac, to select more than one.

Total enrollment:

0

Now you should have an `Instructor` and `Course` created.

We leave objects `delete` and `edit` for you to explore as practice. You could also play with the admin site yourself to create more instructors and courses.

Next, let's customize our admin site.

Customize Admin Site

You may only want to include some of the model fields in the admin site.
To select the fields to be included, we create a `ModelAdmin` class and add a fields list with the field names to be included.

- Open `adminsite/admin.py`, add a `CourseAdmin` class:

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

```
1. class CourseAdmin(admin.ModelAdmin):
2.     fields = ['pub_date', 'name', 'description']
```
[Copied!]

- Update previous course registration `admin.site.register(Course)` with the `CourseAdmin` class:

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

```
1.
2. admin.site.register(Course, CourseAdmin)
```
[Copied!]

Now your `adminsite/admin.py` should look like the following:

```
1    from django.contrib import admin
2    from .models import Course, Instructor
3
4    # Register your models here.
5    class CourseAdmin(admin.ModelAdmin):
6        fields = ['pub_date', 'name', 'description']
7    admin.site.register(Course, CourseAdmin)
8    admin.site.register(Instructor)
```

- Refresh the admin page and try adding a course again, and you should see only
  `Pub date`, `Name`, `Description` fields are included.

Add course

Pub date:	[ ] Today | 🗓 Note: You are 5 hours behind server time.
Name:	[online course]
Description:	[ ]

Coding Practice: Customize Fields for Instructor Model

Include only `user` and `Full Time` fields for `Instructor` model.

```
1. 1
2. 2
3. 3
4. 4
```

```
1. class InstructorAdmin(admin.ModelAdmin):
2.     fields = #<HINT> add user, full_time field names here#
3.
4. admin.site.register(Instructor, InstructorAdmin)
```
Copied!

▼ Click here to see solution

```
1. 1
2. 2
3. 3
4. 4
```

```
1. class InstructorAdmin(admin.ModelAdmin):
2.     fields = ['user', 'full_time']
3.
4. admin.site.register(Instructor, InstructorAdmin)
```
Copied!

Associate Related Models

In previous coding practice, you were asked to include `user` model, which acts as a parent model for
`instructor`, and you can add such related object while creating an `Instructor` object.

Django admin provides a convenient way to associate related objects on a single model managing page.
This can be done by defining `Inline` classes.

Let's try to manage `Lesson` model together with `Course` model on `Course` admin page.

- Open `adminsite/admin.py`, add a `LessonInline` class before `CourseAdmin`:

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

```
1. class LessonInline(admin.StackedInline):
2.     model = Lesson
3.     extra = 5
```
Copied!

and update `CourseAdmin` class by adding a `inlines` list

1. 1
2. 2
3. 3

```
1. class CourseAdmin(admin.ModelAdmin):
2.     fields = ['pub_date', 'name', 'description']
3.     inlines = [LessonInline]
```

Copied!

Now you can see by adding the `inlines` list, you can adding lessons while creating a course

---

Add course

Pub date:	[ ] Today | 📅 Note: You are 5 hours behind server time.
Name:	online course
Description:	[ ]

LESSONS

Lesson: #1

Title:	title
Order:	0
Content:	[ ]

Lesson: #2

Summary

In this lab, you have learned how to use the admin site provided by Django to manage the models defined in the `models.py` of your Django app. You have also learned how to customize the admin site to include the subset of model fields as well as creating `Inline` classes to add related objects on the same page.

Author(s)

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

Changelog

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