

# Django Admin site



**Estimated time needed:** 20 minutes

In this lab, you will learn to use the admin site provided by Django to manage models, customize the admin site, and create Inline classes for adding related objects on the same page.

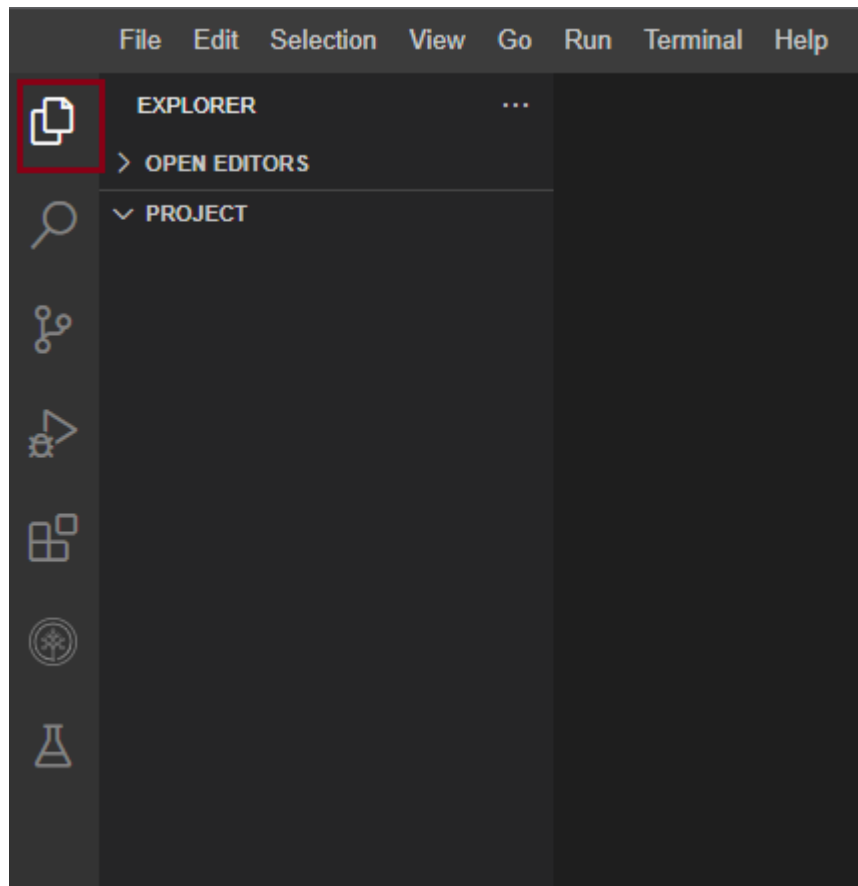
## Learning Objectives

- Understand the concepts of Django admin site
- Create and customize your Django admin site to manage the content of an onlinecourse app

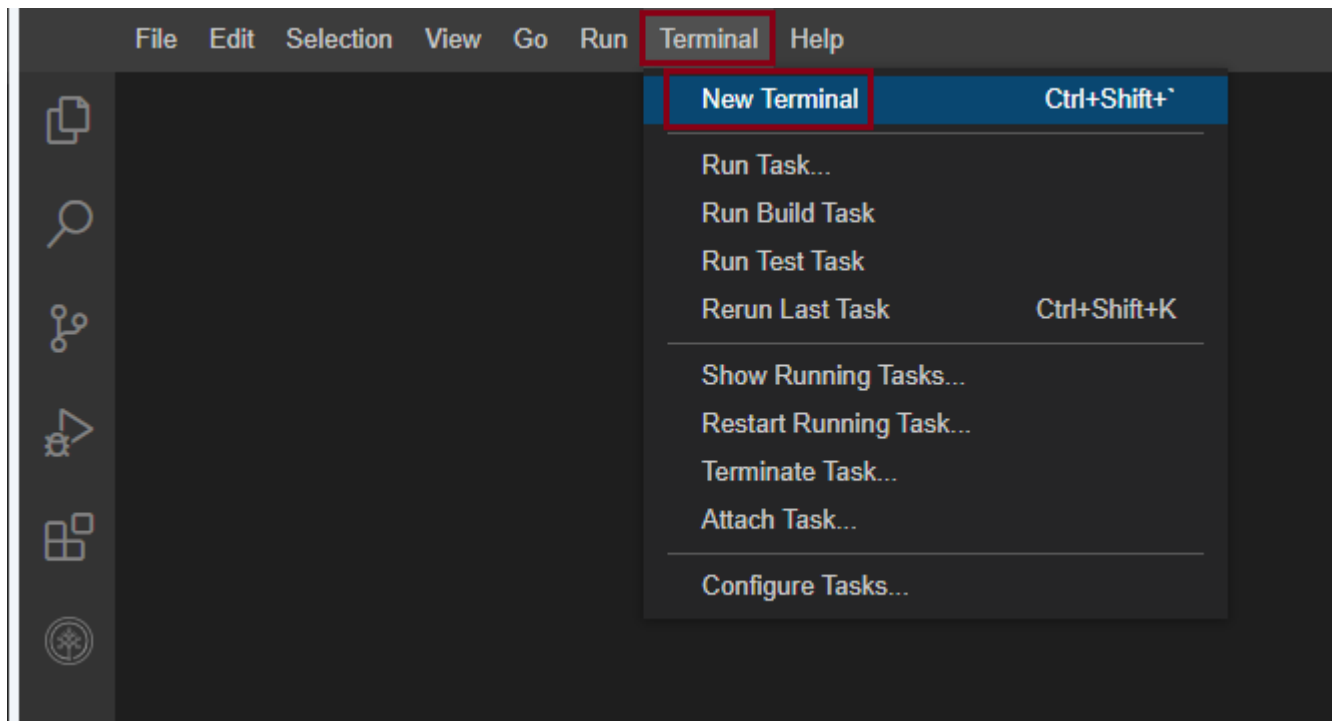
## Working with files in Cloud IDE

If you are new to Cloud IDE, this section will show you how to create and edit files, which are part of your project, in Cloud IDE.

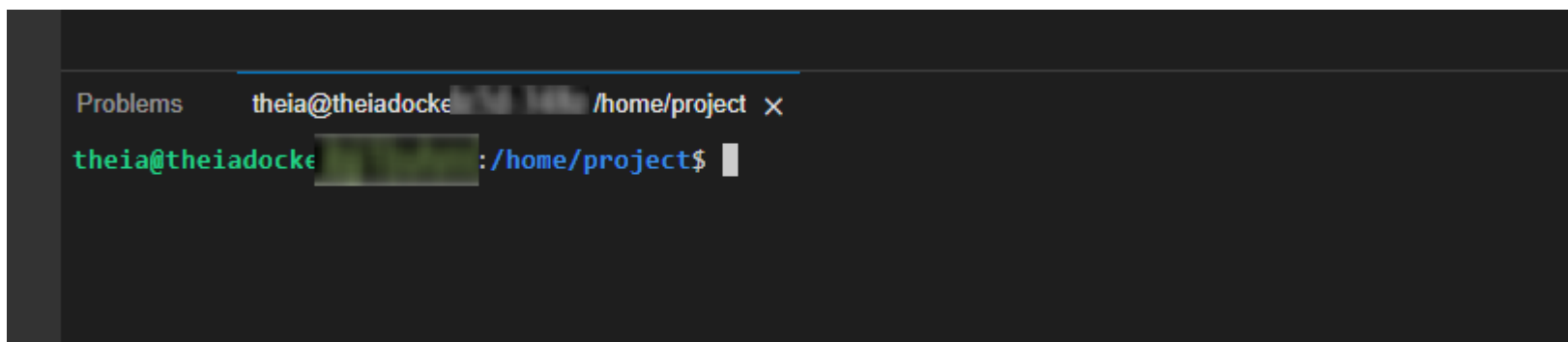
To view your files and directories inside Cloud IDE, click on this files icon to reveal it.



Click on New, and then New Terminal.



This will open a new terminal where you can run your commands.



## Concepts covered in the lab

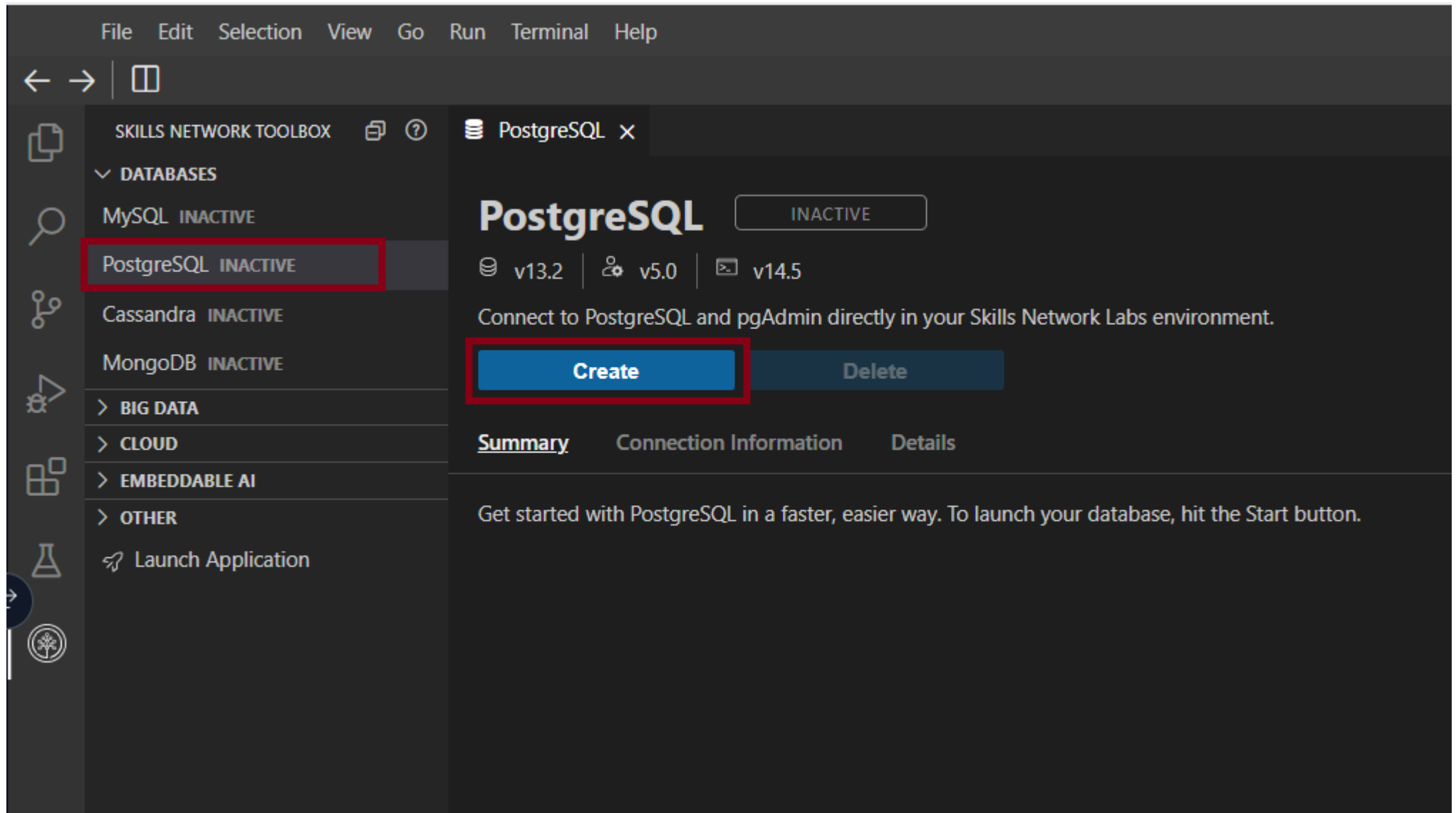
1. Django admin site: A built-in feature of the Django web framework that serves as an interface and allows authorized

users to perform various management operations on the data stored in the application's database.

2. **Superuser:** A user account with administrative privileges that allows superusers to perform administrative tasks and manage the application's data.
3. **Admin class:** A class that helps fine-tune the behavior, appearance, and functionality of models within the Django admin site.
4. **Inline class:** Allow you to include related model instances in the same page/form as the parent model, instead of switching between different forms or screens.

## Start PostgreSQL in Theia

If you are using the Theia environment hosted by [Skills Network Labs](#), you can start the pre-installed PostgreSQL instance 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.

File Edit Selection View Go Run Terminal Help

← →

☐

SKILLS NETWORK TOOLBOX

▼ DATABASES

MySQL INACTIVE

PostgreSQL ACTIVE

Cassandra INACTIVE

MongoDB INACTIVE

> BIG DATA

> CLOUD

> EMBEDDABLE AI

> OTHER

Launch Application

PostgreSQL X

ACTIVE

v13.2 | v5.0 | v14.5

Connect to PostgreSQL and pgAdmin directly in your Skills Network Labs environment.

Create Delete

Summary Connection Information Details

POSTGRES\_USERNAME: postgres

POSTGRES\_HOST: 172.21.244.157

POSTGRES\_PORT: 5432

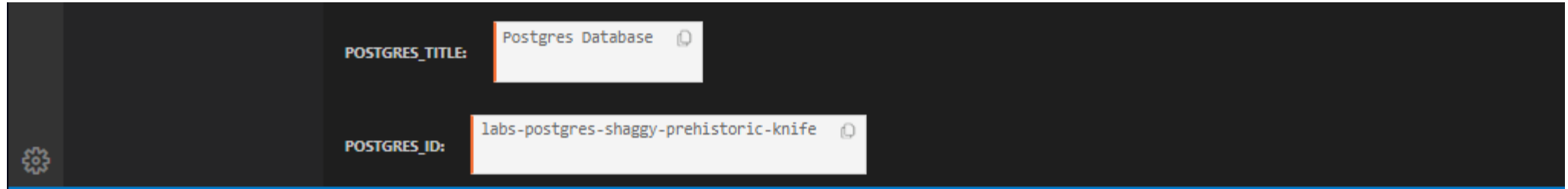
URL: https://labs-postgres-shaggy-prehistoric-knife.postgres.databases.labs.skills.network

POSTGRES\_URL: https://labs-postgres-shaggy-prehistoric-knife.postgres.databases.labs.skills.network

PostgreSQL CLI Command: export PGPASSWORD=pJDH4iXaVvFhH105j5EEAUP8; psql --host 172.21.244.157 -p 5432 -U postgres

POSTGRES\_COMMAND: export PGPASSWORD=pJDH4iXaVvFhH105j5EEAUP8; psql --host 172.21.244.157 -p 5432 -U postgres

POSTGRES\_PASSWORD: pJDH4iXaVvFhH105j5EEAUP8



A screenshot of a dark-themed form interface. On the left, there is a gear icon. The form contains two input fields. The first field is labeled 'POSTGRES\_TITLE:' and contains the text 'Postgres Database'. The second field is labeled 'POSTGRES\_ID:' and contains the text 'labs-postgres-shaggy-prehistoric-knife'. Both fields have a small circular icon to their right.

- Install these must-have packages before you setup the environment to access postgres.

```
pip install --upgrade distro-info  
pip3 install --upgrade pip==23.2.1
```

- Install the Psycopg adapter:

```
pip install psycopg2-binary==2.9.7
```

- A pgAdmin instance is also installed and started for you.

The screenshot shows the Skills Network IDE interface. On the left is a sidebar with a menu under 'SKILLS NETWORK TOOLBOX' containing 'DATABASES', 'BIG DATA', 'CLOUD', 'EMBEDDABLE AI', and 'OTHER'. Under 'DATABASES', 'MySQL' is 'INACTIVE' and 'PostgreSQL' is 'ACTIVE'. Below 'OTHER' is a 'Launch Application' option. The main panel is titled 'PostgreSQL' with an 'ACTIVE' status indicator. It shows versions 'v13.2', 'v5.0', and 'v14.5'. Below this is a message: 'Connect to PostgreSQL and pgAdmin directly in your Skills Network Labs environment.' with 'Create' and 'Delete' buttons. A tabbed interface has 'Summary', 'Connection Information', and 'Details'. The 'Summary' tab is active, showing: 'Your database and pgAdmin server are now ready to use and available with the following login credentials. For more details on to navigate PostgreSQL, please check out the Details section.' Below this, a red box highlights the text 'You can manage PostgreSQL via:' followed by 'pgAdmin' and a copy icon button. At the bottom, it says 'Or to interact with the database in the terminal, select one of these options:' with 'PostgreSQL CLI' and 'New Terminal' buttons.

## Import an onlinecourse App Template

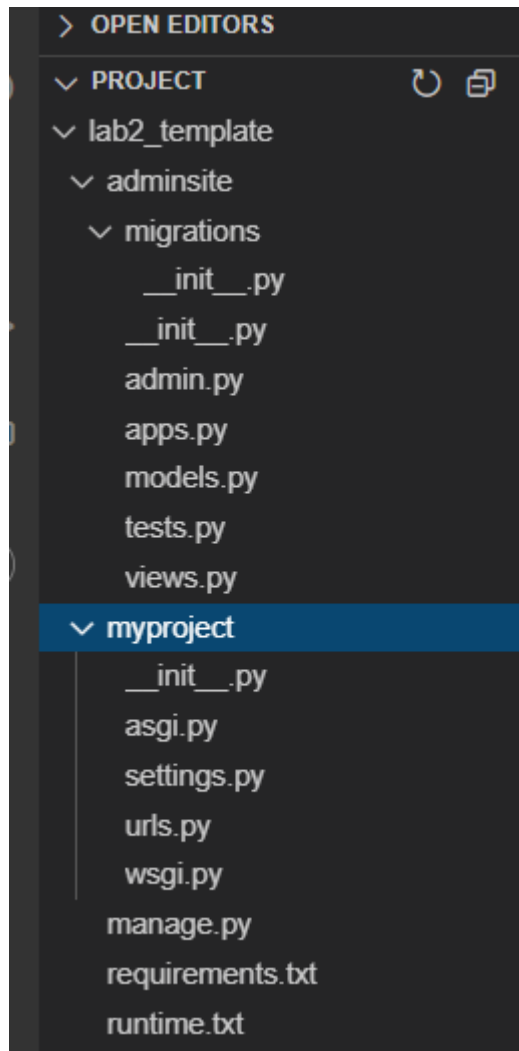
If the terminal was not open, go to `Terminal > New Terminal` and make sure your current Theia directory is `/home/project`.



- Run the following command-lines to download a code template for this lab

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

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  
`cd lab2_template`

Let's set up a virtual environment which has all the packages we need.

```
pip install virtualenv
virtualenv djangoenv
source djangoenv/bin/activate
```

```
pip install -r requirements.txt
```

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

- Open `settings.py` file located in `myproject` directory 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

```
python3 manage.py makemigrations
```

- and run migration

```
python3 manage.py migrate
```

## 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 this command to create a new superuser:

```
python3 manage.py createsuperuser
```

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

Superuser created successfully.

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

- Start the development server

```
python3 manage.py runserver
```

- 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.

The screenshot displays the Cloud IDE interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The left sidebar contains a list of icons and menu items. The 'Launch Application' menu item is highlighted with a red box. The main panel shows the 'Launch Your Application' dialog box, which includes a message about selecting or entering a port number, a text input field for the 'Application Port' (also highlighted with a red box), and a button labeled 'Your Application'.

File Edit Selection View Go Run Terminal Help

SKILLS NETWORK TOO... ?

> DATABASES

> BIG DATA

> CLOUD

OTHER

Launch Application

Open Link inside IDE

Open Cloud IDE with Open...

Open Cloud IDE Basic

Suggest a feature or submi...

About Cloud IDE

Launch Application X

# Launch Your Application

To open any application in the browser, please select or enter the port number below.

Application Port

Your Application

- 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.

# Django administration

## Site administration

### AUTHENTICATION AND AUTHORIZATION

**Groups**



**Users**



## 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

```
admin.site.register(Course)
admin.site.register(Instructor)
```

- 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 is `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:**

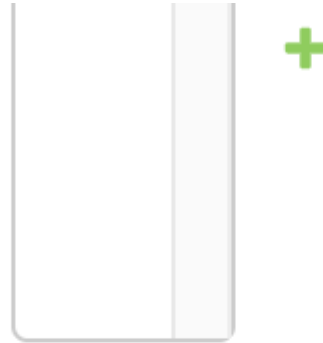
2020-11-29

Today | 

Note: You are 5 hours behind server time.

**Instructors:**

root



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

**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:

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

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

```
admin.site.register(Course, CourseAdmin)
```

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.

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

▼ Click here to see solution

```
class InstructorAdmin(admin.ModelAdmin):  
    fields = ['user', 'full_time']  
admin.site.register(Instructor, InstructorAdmin)
```

## 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`:

```
class LessonInline(admin.StackedInline):  
    model = Lesson  
    extra = 5
```

- And update `CourseAdmin` class by adding a `inlines` list

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

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:****Content:**

## Summary

In this lab, you have learned to use the Django admin site for managing the models defined in the `models.py` file of your Django app, to customize the admin site to include the subset of model fields, and to create `Inline` classes for adding related objects on the same page.

**Yan Luo**

**© IBM Corporation. All rights reserved.**