Learning Objectives: Setting Up Django

Learners will be able to...

- **Create a Django project and app**
- **Create an initial database**
- **Create a URL pattern that returns a specific class-based view**

info

Make Sure You Know

You are comfortable entering simple commands in the terminal and using Conda for package management. You are also familiar with Python, including classes.

Limitations

This Django project is a simple site that makes use of a database. Topics like forms and user authentication are not covered. You will use the Django Admin instead.

Django Setup

Movie Review Project

In this second project, we will repeat many of the same tasks as before. Setting up a Django project is formulaic in many ways. Create a project, add an app (or several), setup your URL patterns, define your views, create your templates, etc. This is all standard when starting a new Django project.

For this project, we are going to create a simple website containing movie reviews. The big difference with this project is the inclusion of models, which means we are going to store information in a database. We are also going to create a Django user; a superuser to be precise.

This project ++will not++ cover authenticating Django users or forms to enter information. Instead, this will all be handled by the Django Admin. But, we are getting ahead of ourselves. Let's start by preparing our system for this Django project.

Installing Django

Start by creating the `django` virtual environment.

```
conda create --name django -y
```

Then activate the newly created environment.

```
conda activate django
```

Install Django as well as Black. Black is a Python code formatter. The project describes itself as being uncompromising, which means it is going to make changes to the code that might seem unusual. For instance, Black defaults to double quotes over single quotes. You can read more about the Black code style here. Black is very popular in the Django community, so we will be using this tool to format our code. Both packages are installed from the Conda Forge channel to get their latest versions.

```
conda install -c conda-forge django black -y
```

Starting Our Django Project

Now that we have a virtual environment and Django installed, it is time to create our project. Run the following command to create our Django project. We are going to name it `movie_review`.

```
django-admin startproject movie_review .
```

Use the `tree` command to print a graphical representation of our current directory.

```
tree
```

What was once an empty directory is now contains the directory `movie_review`, which itself contains five files. We also have a `manage.py` file. This is the benefit of using a framework. It knows that you need these directories and files for a proper Django project, so it creates them for you.

```
.
├── manage.py
└── movie_review
│   ├── asgi.py
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
```

Each file in the `movie_review` directory serves a specific purpose:

- `asgi.py` - this is an optional file used with an asynchronous server gateway interface.
- `__init__.py`- this indicates the directory is part of a Python package.
- `settings.py` - this file controls the settings for our Django project.
- `urls.py` - this file tells Django which page to load when a user visits our site.
- `wsgi.py` - the web server gateway interface is used when we host our site on a production server.

The `manage.py` file is not technically part of the Django project since it is outside the `movie_review` directory. However, this file is very important. We will use it when we need to make migrations to the database, run our development server, etc.

If we activated the `django` virtual environment at the beginning of a page, we want to get into the habit of deactivating it at the end of a page.

```
conda deactivate
```

Codio-Specific Settings

Running Django On Codio

info

Django and Codio

The changes on this page are done to allow Django to run on the Codio platform. By default, Django has strict security features that keep Codio from embedding Django.

The code samples below **only** apply for Codio. You **would not** make these changes when running Django outside of Codio.

Before setting up Django to run on Codio, you first need to import the `os` module.

```
import os
```

The two biggest obstacles in getting Django to run inside Codio are recognizing the unique host name for each Codio project and cookies. The changes below will tell Django the exact host name of the Codio project and alter how Django handles cookies. Make sure your settings match the ones below.

```
ALLOWED_HOSTS = ['*']
X_FRAME_OPTIONS = 'ALLOW-FROM ' +
        os.environ.get('CODIO_HOSTNAME') + '-8000.codio.io'
CSRF_COOKIE_SAMESITE = None
CSRF_TRUSTED_ORIGINS = ['https://' +
        os.environ.get('CODIO_HOSTNAME') + '-8000.codio.io']
CSRF_COOKIE_SECURE = True
SESSION_COOKIE_SECURE = True
CSRF_COOKIE_SAMESITE = 'None'
SESSION_COOKIE_SAMESITE = 'None'
```

Scroll down a bit and look for the definition of the variable `MIDDLEWARE`. Comment out the two lines that refer to `csrf`. CSRF stands for cross-site request forgery, which is a way for a malicious actor to force a user to perform an unintended action. Django normally has CSRF protections in

place, but we need to disable them. There are serious security implications to doing so, however your project on Codio is not publicly available on the internet. Comment out the two lines in the `MIDDLEWARE` setting as shown below.

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
#     'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
#     'django.middleware.clickjacking.XFrameOptionsMiddleware',

]
```

Starting the Server

Once you finish making the changes to the `settings.py` file, we want to enforce the Black style guide. Start by opening the terminal with the link below. Then activate the `django` virtual environment.

```
conda activate django
```

Now run Black on the `settings.py` file. If you were to click on the tab for the settings file, you would see the newly formatted settings.

```
black movie_review/settings.py
```

Use the `manage.py` file to run the development server. We are going to specify `0.0.0.0` for the address of localhost and port 8000.

```
python manage.py runserver 0.0.0.0:8000
```

Finally, open the preview of Django running on the development server with the link below. You should see the default success message from Django. We see this message because all we did was start a Django project and change a few settings. There is no website to show, so we get the success message instead.

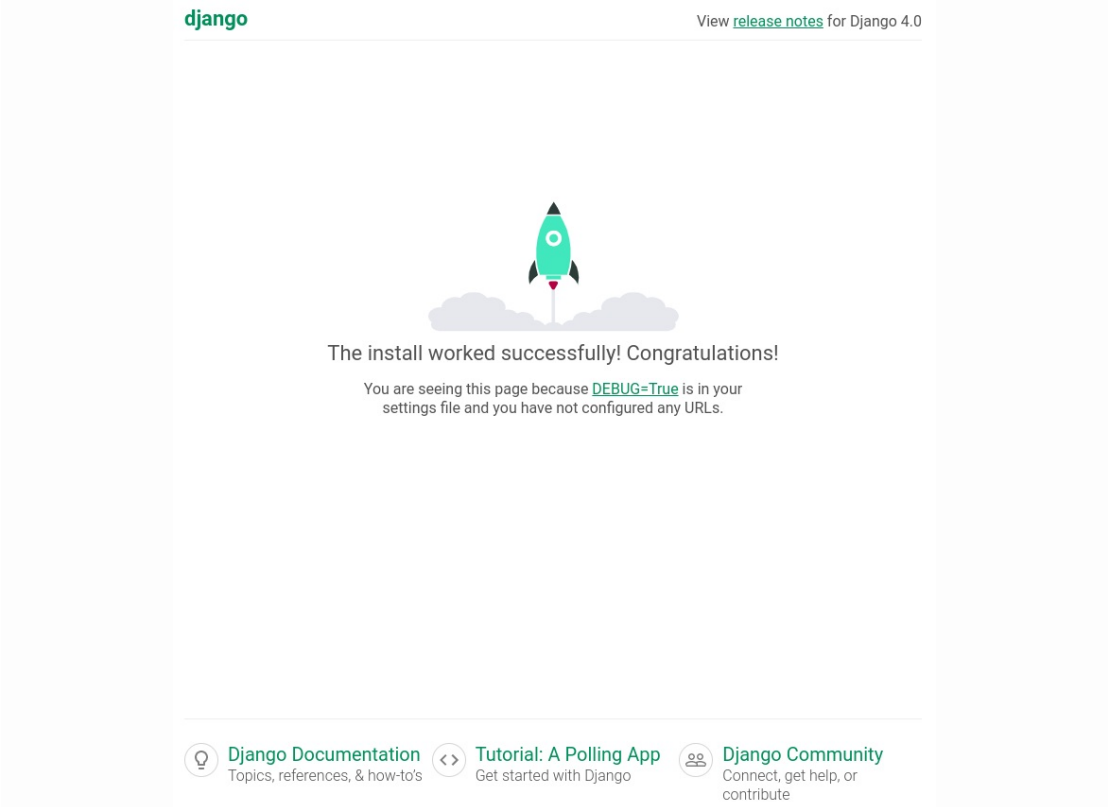

Image depicts the success message from Django that the install is working.

Reminder: these changes only apply to working with Django on Codio. **Do not** make these changes to a project you plan on making available on the internet.

Django Apps

Creating a Django App

We have the skeleton of a Django project. A project is comprised of several Django apps. Each app should perform a specific function of the overall project. Combine all the apps together, and you have a Django project.

Our second Django project is going to build off the ideas from the previous one — views, templates, URL patterns, and styling with Bootstrap. In addition, we will add a model for the database.

Start by activating the `django` virtual environment.

```
conda activate django
```

To add an app to a Django project, use the `manage.py` file with the `startapp` argument. In addition, give this new app a name.

```
python manage.py startapp reviews
```

Next, let's create our initial database with the `migrate` command. We have not yet declared our model (which controls the database's structure), so the database will use Django's default settings.

```
python manage.py migrate
```

If you enter the `tree` command in the terminal, you will see the new structure of our Django project. In addition to the `movie_review` directory, we now have the `reviews` directory. This new directory is our app.

```
.
├── db.sqlite3
├── manage.py
├── movie_review
│   ├── asgi.py
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-310.pyc
│   │   ├── settings.cpython-310.pyc
│   │   ├── urls.cpython-310.pyc
│   │   └── wsgi.cpython-310.pyc
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
└── reviews
    ├── admin.py
    ├── apps.py
    ├── __init__.py
    ├── migrations
    │   └── __init__.py
    ├── models.py
    ├── tests.py
    └── views.py
```

Django automatically creates any files needed by the app. Each file has a specific purpose:

- `admin.py` - handles configuration of the built-in Django Admin feature
- `apps.py` - handles configuration for the app itself
- `migrations` - this directory keeps track of changes to models so the database stays in sync
- `models.py` - handles the definition of database models
- `tests.py` - handles any tests specific to the app
- `views.py` - handles any HTTP requests and responses for the app

In addition to the files in the `reviews` app, the `db.sqlite3` file is our database. We will talk more about this when we declare our model.

Adding the App

Django makes it really easy to add an app to a project. The problem, however, is that the project "does not know" about the newly created app. We need to add our `reviews` app to the list of installed apps for the Django project. Open `settings.py` (the project settings) with the link below. Then add our languages app to the end of the list.

```
INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    'reviews.apps.ReviewsConfig',
]
```

`ReviewsConfig` is a class in the `apps.py`file inside the `reviews` directory (our new Django app). Django created this file and its contents during the app creation process. We do not need to make any changes to this file. We just need to make our Django project aware of it.

Open the terminal and run the Black formatter to make sure our code has a consistent style.

```
black movie_review/settings.py
```

Deactivate the virtual environment.

```
conda deactivate
```