Exercise 2.2: Django Project Set Up

Learning Goals

- Describe the basic structure of a Django project
- Summarize the difference between projects and apps
- Create a Django project and run it locally
- Create a superuser for a Django web application

Reflection Questions

1. **Suppose you're in an interview. The interviewer gives you their company's website as an example, asking you to convert the website and its different parts into Django terms. How would you proceed? For this question, you can think about your dream company and look at their website for reference.**
   (*Hint: In the Exercise, you saw the example of the CareerFoundry website in the Project and Apps section.*)
   A Django project is a complete application structure consisting of multiple modules (apps), configurations files, and a database. A Django app is a module with specific functionality.
   Let's consider the Rakuten Kobo website (https://www.kobo.com/). In Django terms, the website is the project. Inside the project we have individual modules with particular functions (e.g. create an account, sign in, eBooks, audiobooks, Kobo Plus, blog, etc.). In Django terms, these modules are the apps. For example, the Kobo Plus lets you subscribe Kobo Plus eBooks, audiobooks or both, as well as check the available eBooks and audiobooks of Kobo Plus.
   To create the project structure in Django I need to go the terminal, active the virtual environment (where I installed Django) and execute execute one of the following commands, according to my OS: django-admin startproject kobo (Mac/Linux) or django-admin.exe startproject kobo (Windows). This command creates a directory (project) with the name kobo in my current working directory. To create the apps I need to run the following command: django-admin.exe startapp [app name] (or py manage.py startapp [app name]).

2. **In your own words, describe the steps you would take to deploy a basic Django application locally on your system.**
   To deploy a Django application locally on my system I need to:
   1. Run migrations which creates the necessary database tables (i.e., prepares the backend for the application). In Django, migration refers to making changes to the database schema.
   2. Run the server which deploys the application to localhost and gives you a web link to access the application from the browser.

   To run migrations I need to go to the root directory of my project and enter one of the following commands in the terminal, according to my OS: python manage.py migrate (Mac/Linux) or py

manage.py migrate (Windows). Running migrations for the first time will create my database (Django's default database is SQLite). This command applies the migrations. I should see that the status of everything is OK and an additional dq.sqlite3 file now appears under the root folder. Any future change to my database will require me to run migrations again.
To run server I need to go to the root directory of my project and enter one of the following commands in the terminal, according to my OS: python manage.py runserver (Mac/Linux) or py manage.py runserver (Windows). By default the server will run on port 8000. I can copy the link that appears in the terminal and past it into the browser. If all goes well, the landing page's message will confirm to me that the Django server install worked successfully.

3. **Do some research about the Django admin site and write down how you'd use it during your web application development.**
   The Django admin site reads metadata from my models to provide a quick, model-centric interface where trusted users can manage content of my site. The automatic admin interface is not intended for building the frontend; it's an internal management tool (i.e. just for use by admins, or people internal to your organization).
   The Django admin site can use models to build a site area that I can use to create, view, update and delete records. To add the models to the Django admin site I have to register them. This can save me a lot of time during development because it's easier to test my models and get a feel for whether I have the right data. The Django admin site can also be useful for managing data in production, depending on the type of website.