CAREERFOUNDRY

Python for Web Developers Learning Journal

Exercise 2.1: Getting Started with Django

Learning Goals

- Explain MVT architecture and compare it with MVC
- Summarize Django's benefits and drawbacks
- Install and get started with Django

Reflection Questions

1. Suppose you're a web developer in a company and need to decide if you'll use vanilla (plain) Python for a project, or a framework like Django instead. What are the advantages and drawbacks of each?
   Vanilla Python: Using plain Python means writing code from scratch without using any pre-built frameworks. It gives you full control over your code and can be simpler for small projects. However, it can be more time-consuming and complex for larger projects

because you have to handle everything yourself, like routing, database management, and security

Django (Framework): Django is a web framework for Python that provides a lot of pre-built tools and features for web development. It can speed up development time significantly by providing ready-to-use components for things like database management, authentication, and URL routing. However, it may have a steeper learning curve for beginners, and you might have to adapt to its conventions and structure.

2. In your own words, what is the most significant advantage of Model View Template (MVT) architecture over Model View Controller (MVC) architecture?
   MVT vs. MVC: The biggets advantage of the Model-View-Template (MVT) architecture over Model-View-Controller (MVC) is its simplicity. In MVT, the template layer handles the user interface and presentation logic, which makes it easier to design and manage the frontend. In contrast, MVC requires a more complex setup with separate components for the model, view, and controller, which can be more challenging to understand and maintain, especially for beginners

3. Now that you've had an introduction to the Django framework, write down three goals you have for yourself and your learning process during this Achievement. You can reflect on the following questions if it helps:

   - What do you want to learn about Django?
     I'm excited to see how its framework is formatted and what features it provides

   - What do you want to get out of this Achievement?
     Again, I want to continue to learn about something new.

   - Where or what do you see yourself working on after you complete this Achievement?
     I will likely touchup some of my portfolio then pick up a few YouTube tutorials on Javascript, React, and Python so that I can get better at them.

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.*)

   Analyze the Company's Website:

   Understand the structure of the website, including its pages, navigation, and content.
   Identify the main components, such as the homepage, about page, products/services pages, contact page, etc.

   Map Website Components to Django Terms:

   URLs: Each page of the website corresponds to a URL. In Django, URLs are defined in the urls.py file using the path() function.

   Views: Views in Django are Python functions or classes that handle requests and return responses. Each page of the website would have a corresponding view function or class.

   Templates: Templates in Django are HTML files that define the structure of a page. Each page of the website would have a corresponding template.

   Models: Models in Django define the structure of the database. If the website has dynamic content (e.g., blog posts, products), each type of content would be represented by a model.

   Static Files: Static files such as CSS, JavaScript, and images are served by Django's staticfiles app and stored in the static directory of the project.

   Media Files: Media files such as user-uploaded images are stored in the media directory of the project and served using Django's MEDIA_URL setting.

   Admin Interface: Django provides an admin interface for managing site content. Models registered with the admin interface can be managed through the admin site.

   Create Django Project and App:

   Create a new Django project using django-admin startproject.
   Create a new Django app using python manage.py startapp.
   Define URL patterns, views, models, templates, static files, and media files as needed based on the analysis of the company's website.
   Implement Django Functionality:

Define URL patterns in the urls.py file of the project and app to map URLs to views.
Write view functions or classes to handle requests and return responses.
Create templates for each page of the website.
Define models for dynamic content if applicable.
Configure static and media files settings in the project's settings file.

2. In your own words, describe the steps you would take to deploy a basic Django application locally on your system.

   Create a new Django project using django-admin startproject.
   Create a new Django app using python manage.py startapp.
   Create superuser admin
   Define URL patterns, views, models, templates, static files, and media files as needed
   Implement Django Functionality:

   Define URL patterns in the urls.py file of the project and app to map URLs to views.
   Create templates for each page of the website.

3. Do some research about the Django admin site and write down how you'd use it during your web application development.

   In web app development, I can use Django admin as an internal tool for managing the application's data and users. It's particularly useful for applications where content management is a key feature, such as blogs, content management systems (CMS), or internal tools for managing data. By using Django admin, I can quickly build a functional admin interface without having to write a lot of custom code, saving time and effort

Exercise 2.3: Django Models

Learning Goals

- Discuss Django models, the "M" part of Django's MVT architecture
- Create apps and models representing different parts of your web application
- Write and run automated tests

Reflection Questions

1. Do some research on Django models. In your own words, write down how Django models work and what their benefits are.

2. In your own words, explain why it is crucial to write test cases from the beginning of a project. You can take an example project to explain your answer.

Exercise 2.4: Django Views and Templates

Learning Goals

- Summarize the process of creating views, templates, and URLs
- Explain how the “V” and “T” parts of MVT architecture work
- Create a frontend page for your web application

Reflection Questions

1. Do some research on Django views. In your own words, use an example to explain how Django views work.

2. Imagine you’re working on a Django web development project, and you anticipate that you’ll have to reuse lots of code in various parts of the project. In this scenario, will you use Django function-based views or class-based views, and why?

3. Read Django’s documentation on the Django template language and make some notes on its basics.

Exercise 2.5: Django MVT Revisited

Learning Goals

- Add images to the model and display them on the frontend of your application
- Create complex views with access to the model
- Display records with views and templates

Reflection Questions

1. In your own words, explain Django static files and how Django handles them.

2. Look up the following two Django packages on Django's official documentation and/or other trusted sources. Write a brief description of each.

Package	Description
ListView	
DetailView	

3. You're now more than halfway through Achievement 2! Take a moment to reflect on your learning in the course so far. How is it going? What's something you're proud of so far? Is there something you're struggling with? What do you need more practice with? You can use these notes to guide your next mentor call.

Exercise 2.6: User Authentication in Django

Learning Goals

- Create authentication for your web application
- Use GET and POST methods
- Password protect your web application's views

Reflection Questions

1. In your own words, write down the importance of incorporating authentication into an application. You can take an example application to explain your answer.

2. In your own words, explain the steps you should take to create a login for your Django web application.

3. Look up the following three Django functions on Django's official documentation and/or other trusted sources and write a brief description of each.

Function	Description

authenticate()	
redirect()	
include()	

Exercise 2.7: Data Analysis and Visualization in Django

Learning Goals

- Work on elements of two-way communication like creating forms and buttons
- Implement search and visualization (reports/charts) features
- Use QuerySet API, DataFrames (with pandas), and plotting libraries (with matplotlib)

Reflection Questions

1. Consider your favorite website/application (you can also take CareerFoundry). Think about the various data that your favorite website/application collects. Write down how analyzing the collected data could help the website/application.

2. Read the Django official documentation on QuerySet API. Note down the different ways in which you can evaluate a QuerySet.

3. In the Exercise, you converted your QuerySet to DataFrame. Now do some research on the advantages and disadvantages of QuerySet and DataFrame, and explain the ways in which DataFrame is better for data processing.

Exercise 2.8: Deploying a Django Project

Learning Goals

- Enhance user experience and look and feel of your web application using CSS and JS

- Deploy your Django web application on a web server
- Curate project deliverables for your portfolio

Reflection Questions

1. Explain how you can use CSS and JavaScript in your Django web application.

2. In your own words, explain the steps you'd need to take to deploy your Django web application.

3. (Optional) Connect with a few Django web developers through LinkedIn or any other network. Ask them for their tips on creating a portfolio to showcase Python programming and Django skills. Think about which tips could help you improve your portfolio.

4. You've now finished Achievement 2 and, with it, the whole course! Take a moment to reflect on your learning:
    a. What went well during this Achievement?
    b. What's something you're proud of?
    c. What was the most challenging aspect of this Achievement?
    d. Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Django skills?

Well done—you've now completed the Learning Journal for the whole course.