

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 offers full control over architecture and implementation, with no framework overhead, making it suitable for simple projects and learning core concepts. However, it requires building features like authentication, admin panels, and ORM from scratch, which slows development and increases maintenance. Security must be implemented manually, increasing risk.

Django accelerates development with built-in features (admin panel, authentication, ORM) and strong security defaults. It provides a scalable architecture, clear conventions, and extensive documentation. The trade-offs are a steeper learning curve, an opinionated structure that limits flexibility, and potential overkill for very simple projects.

Decision Factor: Use Django for full-featured web applications that need rapid development, security, and scalability. Use vanilla Python for simple scripts, APIs with specific requirements, or when maximum control is needed.

2. In your own words, what is the most significant advantage of Model View Template (MVT) architecture over Model View Controller (MVC) architecture?

The main advantage of Model-View-Template (MVT) over Model-View-Controller (MVC) is the separation of presentation logic from business logic. In MVT, the Template handles only display, while the View contains the business logic that prepares data. This makes templates simpler and easier to maintain, and keeps business logic in Python rather than mixed into markup. The framework manages the controller layer, reducing boilerplate and letting developers focus on models, views, and templates.

3. Now that you've had an introduction to Django framework, write down 3 goals you have for yourself and your learning process during this Achievement. You can reflect on the following questions if it helps you. What do you want to learn about Django? What do you want to get out of this Achievement? Where or what do you see yourself working on after you complete this Achievement?

- a. What I want to learn about Django: To create a complete Django app with models, views, templates, and user authentication. Understand how components interact and deploy it to demonstrate end-to-end capability.
- b. What I want to get out of this achievement: Master Django's ORM and database management by learning to design models, use migrations, and perform efficient queries. Apply this to build data-driven features and understand Django's database abstraction.
- c. What I see myself working on after completing this achievement: Building a Django application that leverages the framework's strengths—such as a local community resource hub, a collaborative learning platform, or a neighborhood marketplace. Focusing on features that showcase Django's capabilities: user authentication and permissions for different user roles, an admin interface for content moderation, complex model relationships (many-to-many, foreign keys), and real-time features or API integration. The goal is to create something that solves a real problem in my community.