My Anime List - MVC Project Documentation

Azmain Inquaid Haque Mostakima Yasmin Rashi
230218 230227

November 30, 2024

Contents

1 Introduction

This project, **My Anime List (MVC)**, demonstrates the implementation of the Model-View-Controller (MVC) architectural pattern. The MVC pattern is a design paradigm that separates the application logic into three interconnected components:

1. **Model:** Manages the data and business logic of the application.
2. **View:** Handles the presentation layer, displaying data to the user.
3. **Controller:** Acts as an intermediary, processing user inputs and coordinating between the Model and View.

This separation ensures modularity, scalability, and ease of testing, making the codebase maintainable and organized.

2 Project Structure

The project is organized as follows:

```
My_Anime_List(MVC)/
+-- src/
|   +-- model/
|   |   +-- Anime.java
|   +-- repository/
|   |   +-- AnimeRepository.java
|   +-- service/
|   |   +-- AnimeService.java
|   +-- filter/
|   |   +-- FilterStrategy.java
|   |   +-- GenreFilter.java
|   |   +-- RatingFilter.java
|   +-- display/
|   |   +-- Displayable.java
|   |   +-- AnimeDisplayService.java
|   +-- Main.java
+-- README.md
```

3 Understanding MVC and Its Implementation

The MVC pattern in this project is implemented as follows:

3.1 Model

The **Model** is responsible for representing the application's data. In this project:

- `Anime.java` defines the structure of an anime entity, including attributes such as title, genre, episodes, and rating.
- The `AnimeRepository.java` acts as a database, managing storage, retrieval, and basic CRUD (Create, Read, Update, Delete) operations for anime objects.

3.2 View

The **View** is responsible for presenting data to the user. In this project:

- `Displayable.java` is an interface defining methods for displaying anime data.
- `AnimeDisplayService.java` implements the `Displayable` interface and provides methods to display a list of anime or details about a specific anime in a user-friendly format.

3.3 Controller

The **Controller** processes user inputs and coordinates interactions between the Model and the View. In this project:

- `AnimeService.java` contains business logic for managing anime data, such as filtering anime by genre or rating, and interacts with both the `AnimeRepository` (Model) and the `AnimeDisplayService` (View).

4 Advantages of Using MVC

- **Separation of Concerns:** Each component has a well-defined role, reducing complexity.
- **Reusability:** Components such as filters or display services can be reused across different parts of the application.
- **Ease of Testing:** Each layer can be tested independently.
- **Scalability:** Adding new features, such as new filters or display formats, is straightforward.

5 Code Examples

5.1 Adding Anime to the Repository

Listing 1: Adding anime in Main.java

```
AnimeRepository animeRepository = new AnimeRepository();
AnimeService animeService = new AnimeService(animeRepository);
Anime anime = new Anime("Attack on Titan", "Action", 75, 9.0);
animeService.addAnime(anime);
```

5.2 Filtering Anime by Genre

Listing 2: Filtering anime by genre in Main.java

```
GenreFilter genreFilter = new GenreFilter("Action");
List<Anime> actionAnime = animeService.filterAnime(genreFilter);
AnimeDisplayService displayService = new AnimeDisplayService();
displayService.display(actionAnime);
```

6 Conclusion

This project showcases a clean implementation of the MVC pattern in Java, demonstrating how to build a modular and maintainable application. The use of interfaces and classes adhering to the Single Responsibility Principle further enhances the code quality.

Feel free to explore, test, and extend this project as you learn more about software design principles and patterns!