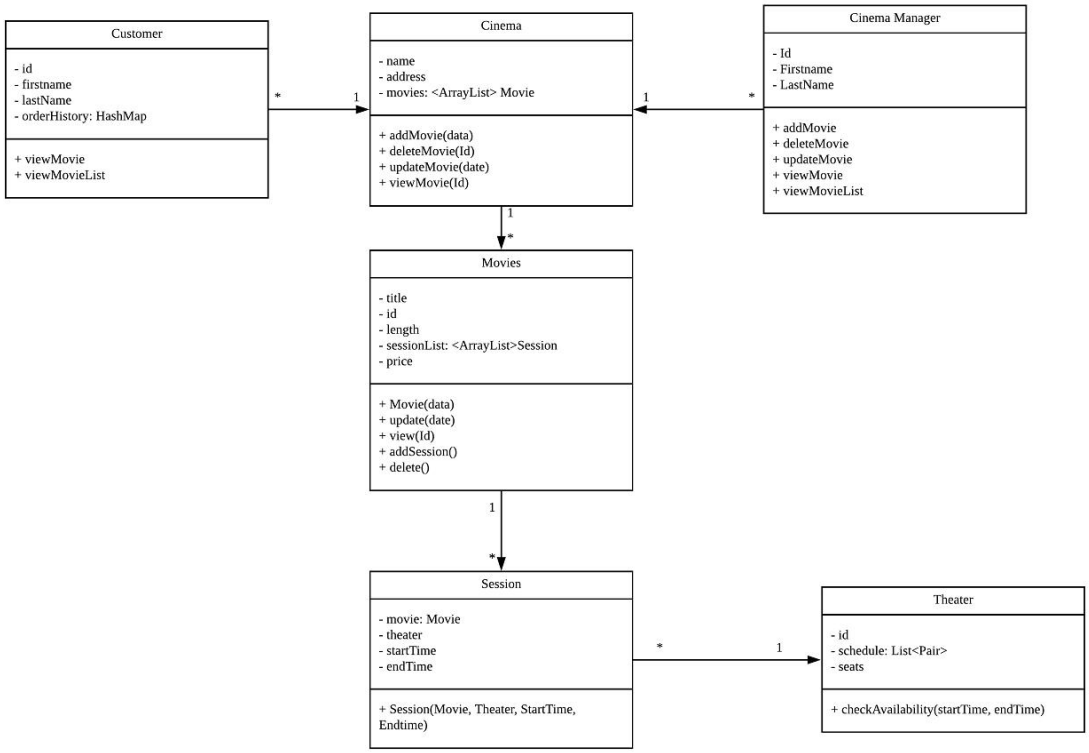
Student Name: Linyuan Zhao
Haoyang Cui
Student Number:947588

Domain Class E-R diagram:



The domain class ER diagram above demonstrates our design for domain model:

1. Cinema: contains the information of a cinema(name, address, cinema_id). it has one-to-many relationships with movie class, cinema Manager class and customer class.
2. Movie: contain basic information of a movie(movie_Id, name, price and length). It has a one-to-many relationship with session class.
3. Customer: contain basic information of a customer(first name, last name, username, password).
4. Cinema manager: similar with customer class.
5. Session: contains session_Id, seat_id available seat and movie_id. It has a one-to-one relationship with timeRange.
6. TimeRange: contains startTime and endTime.

High Level Architecture

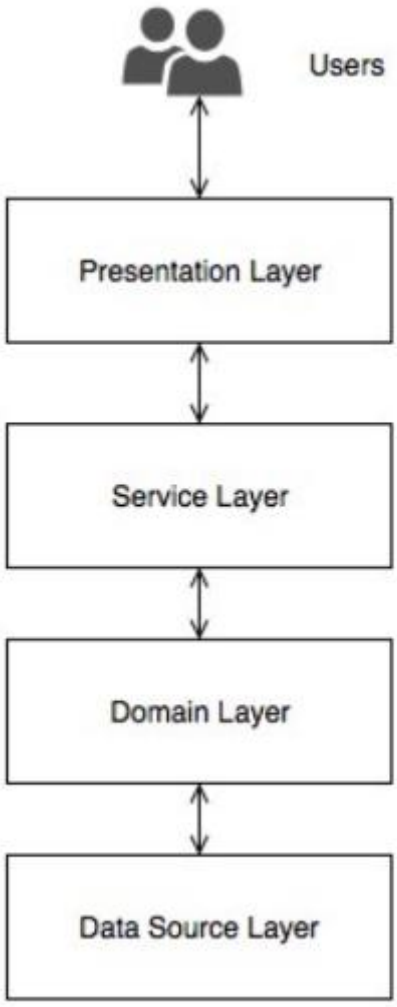
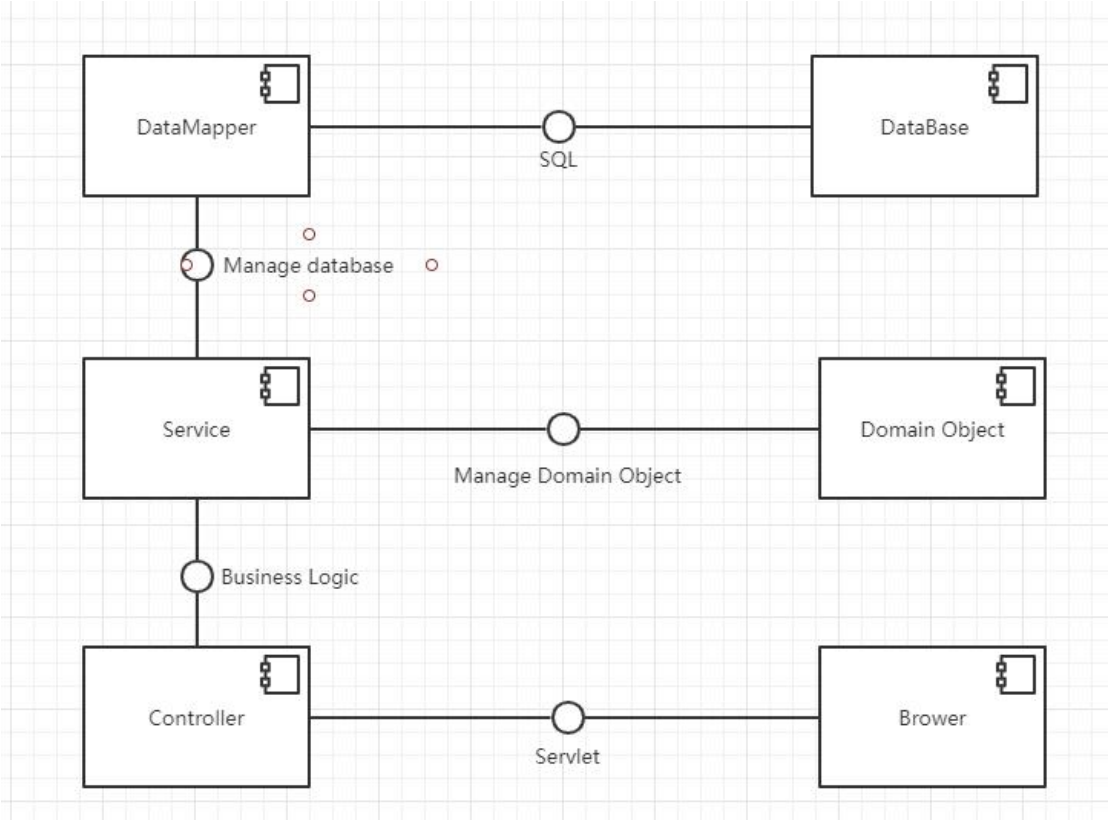


Figure 2. high level architecture

The system with a four-tier layered architecture is shown above, which are presentation layer, service layer, domain layer and data source layer from up to down. Individual layer does not have any overlaps with others but prepare interfaces to communicate, increasing the reusability and scalability.



Datamapper component manipulates database through SQL statement and provides an interface for Service component to let it manage the insert/deleted/update/find operations in the database. Service component converts results from database to domain object by the interface of Domain Object component. Controller component calls functions in Service component through its interface to deal with the business logic. Browser, which is the user interface, gets information from Controller component through servlet.

presentation layer

The presentation layer deals with the user interactions of the system. In the project, we decide to use front controller pattern, because we treat our project as a complex enterprise application. Using a single controller makes the configuration much easier compared to the page controller. On the other hand, this pattern provide an option of adding new command at runtime. Consequently, the source code in controller can be divided into two parts: (1) a handler called frontServlet, whose responsibilities are receiving the request and passing to different command class to process, for example: view movie command and delete movie command. (2) an abstract command class and command classes that extend it. Each command provides their own process by overriding the process method.

是否需要这两个图？

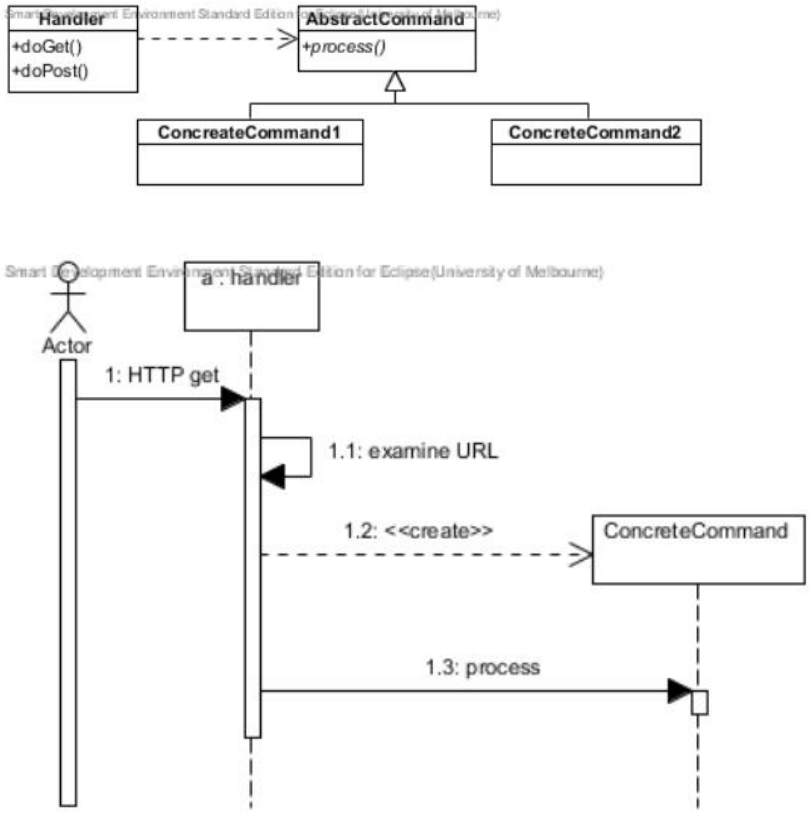
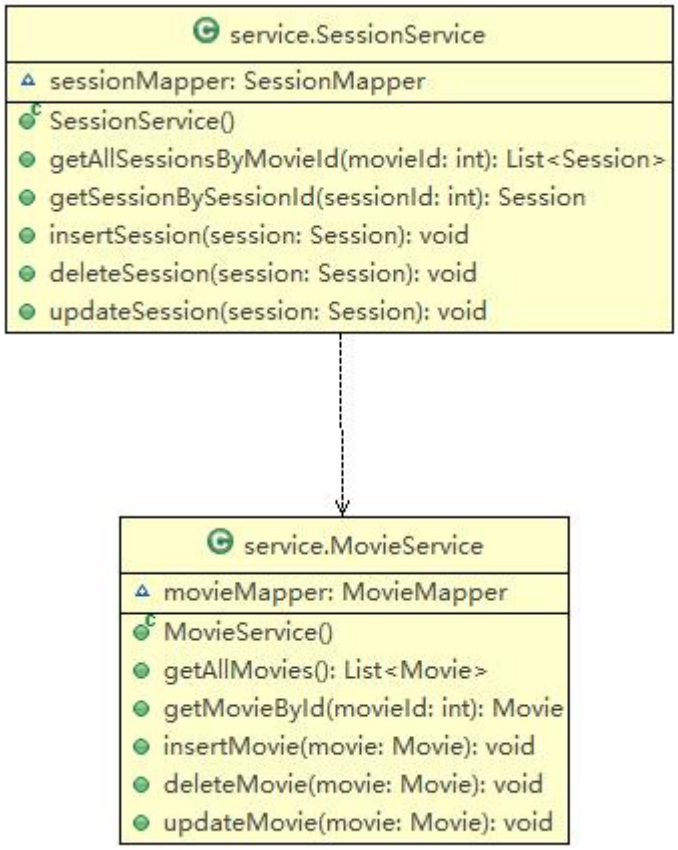


Figure 6.6: Behaviour of the *Front Controller* design pattern.

The pattern used for handling the view in our project currently is template view. In this phase, we are focusing on business logic and serializing all components. As a result, we have to sacrifice maintainability and testability. At the next sprint, we are going to improve this part to transform view.

Service Layer



SessionService is responsible for handling creating, updating, and deleting a schedule.
MovieService is responsible for the operations about movies, such as creating a movie, view all the movies, edit the basic information of a movie, delete a movie.

Next sprint:
We are going to create

1. CustomerService to handle the request to view all users and orders in database.
2. AuthenticationService to check whether the input username and password is valid.

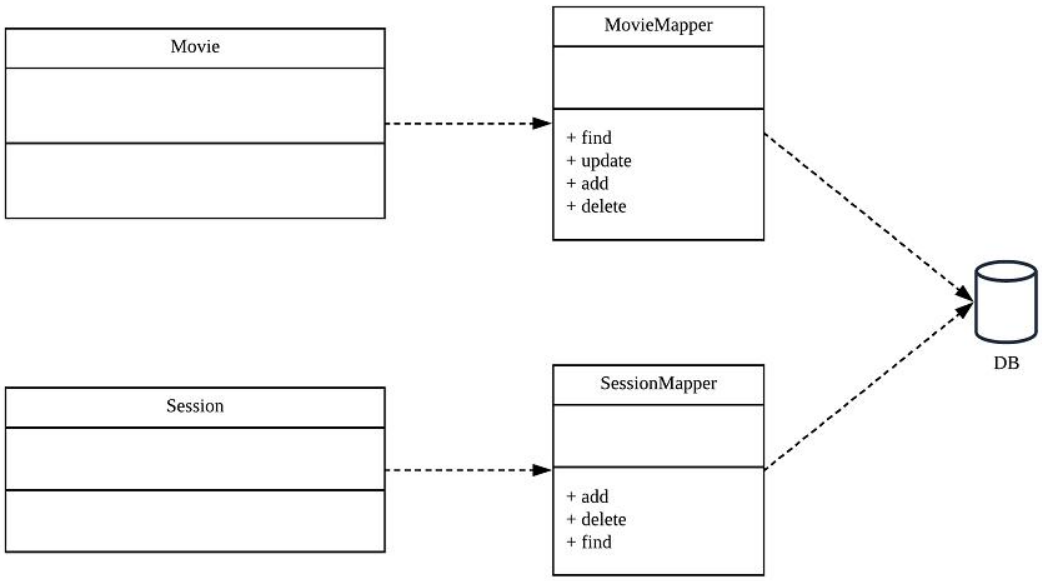
Domain Layer

1. lazy load

Data Source Layer

A figure here to show data source layer

1. Data Mapper



This pattern is used as the architectural design in the data source layer. In consideration of maintainability and reusability, we selected domain model pattern in our system. Thus, we chose data mapper pattern for data source layer because it is compatible with domain model pattern. The six data mappers in the data source layer connects the database with the website. Since they all have insert/update/delete operations to database, they implement the interface from dataMapper

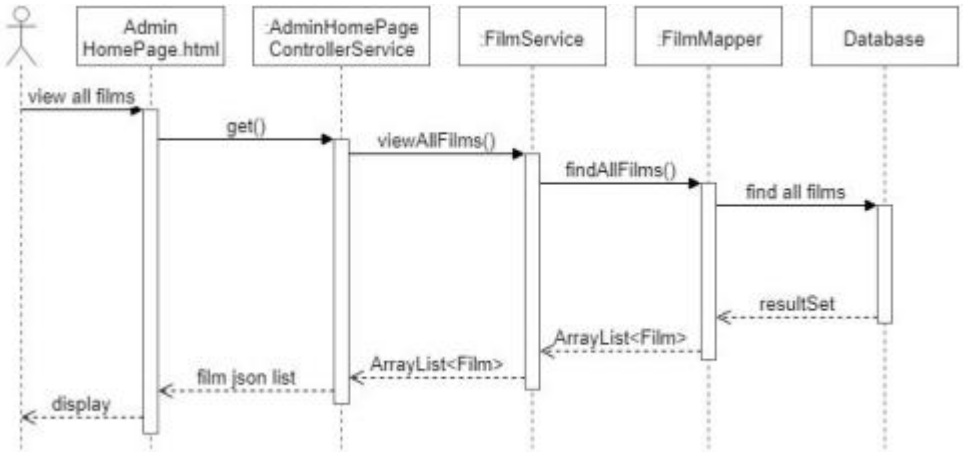
2. Identity Map

Identity map pattern is applied to make sure every domain object is read from database only once.

3. Unit of Work

We used unit of work pattern to process the edit of movie, because the admin can make multiple create, update and delete operations to the movie table in the database in one edit, using unit of work pattern would save some cost of multiple connection to the database.

Interaction diagram



Code and depolyment

Github:
https://github.com/HaoyangCui0830/SWEN90007-Project-OnlineMovieTicketBookingSystem

Heroku: https://online-movie-ticket-booking.herokuapp.com/