Report

The concert-booking system is designed to three parts: the client service, common and server.

On the server side, multiple resource classes are used for different http services. This increases the scalability of the system because it is more flexible to add different http methods or introduce new resource classes to the system compared to using one single resource class. Since HTTP is a stateless protocol, cookies are used to store the client state on the client side. However, in order to authenticate the user for some of the methods in resource classes, cookies are used as parameters and each time, which requires a method, which needs to be used for all the resource classes. Therefore, all the resource classes need to extends an abstract class called ServiceResource, which stores the common method for making cookies and fields that need to be used in the resource classes. This reduces redundancy code in the system, which, in other words, increases the flexibility of the system.

Domain model is separated from the DTO classes, which is different to one of our labs. The domain model classes are mainly for the database and the DTO classes are the objects that need to transfer to the clients.

In order to decrease the complexity of the domain model, only one domain model class called Reservation is created, which is corresponded to the BookingDTO class and ReservationDTO class. An indicator in the Reservation class is used to indicate whether the reservation is confirmed or not. If it is confirmed, it will be mapped to BookingDTO when transfer back to the client, otherwise, ReservationDTO will be passed back to the user. In order to indicate whether the seat is reserved or not, a Reservation field is added to the seat class. If the seat is reserved (or booked), that field will have a reference of the reservation object, otherwise, it will remain null. When the reservation is expired, this field will change back to null again. In order to avoid the situation where the first client is in the process of reservation and the second client breaks in and reserves the seat first, Optimistic lock is used to lock the seat and reservation during the process of reservation.

Since AWS service is a third-party service and assuming each client has its generated Acess key and Acess Secret key, it should not be put in the server side of the system. An AWS class is created to deal with the AWS service such as download the image from Amazon server and store the images in the client side in order to avoid wasting of memory in the server. Since there is no test case for the functionality of getting image from Amazon server, a simple test case is added to check whether the downloaded image is the same name as the system downloaded. The image object can be visualised when a break point is toggled and it can be displayed by Intellij IDE.

To increase the usability of the system, a config class is added to the common package for storing the path that needs to use for the resource classes to decrease the number of magic numbers in the system. Whenever a new service is added into the system, the path of the new resource class or the new methods in the resource class can be added to the Config class, which increases the scalability of the system.