07-Sep-2022 Team Meeting

Date: 7th Sep
Time: 4:15 - 6:15pm
Location: Tutorial

Attendees:

- Edward
- Haoxin
- Ray
- Yifei

Orange means it's discussion with Luke

Agenda

- Who should handle the conditions in complex queries?
    - Suppose we got a listing, product and category table each containing some information about a listing in the database
    - Should the listing mapper be responsible for querying every piece of information from all three DB tables (using SQL join command) or should the system make three separate querying in which only the corresponding mapper class could pull the data from the table it is responsible for?
    - Or should we create a separate service layer to deal with that?
    - I know each table should have a mapper that is responsible for pulling the data from the database
    - Listing contains Product and Category. 3 queries or 1 inner join
    - Luke: mapper should be 1 for 1. 1 class 1 mapper.
    - Ray: better to query everything in one go
    - Luke: Going across the 3 tables
    - Luke: It would be better to have 3 queries. If you always want to query three objects together - think about redesign the domain model design
    - Luke: Embed the product category into the
- Do we need error handling
- Database conn pool
    - Ray: We have a conn pool to keep track of how many connections we have. Should the db connection be closed each time
    - Luke: Recycle - not closing any connection
    - Ray: Eduardo mentioned that we should always close connection
    - Luke: Yes. If you a class that is created
    - Decision: Don't close, just give it back and reuse.

- Mapper class in content haviey
    - Ray: I think we are putting too much responsibility
    - Luke: The number of queries
    - Won't be peanlised by having too many methods
    - Luke: either is fine. Personally would split into different methods.
- Should we call mapper on JSP directly for GET?
    - Luke: please don't do that. You want backend do backend, and frontend do frontend. So please don't mix them up.
    - Edward: how do we need information
    - Luke: httpResponse method to send data to frontend

- Question to clarify with Luke
    - Can "Admin onboard sellers" be the same as "Admin add seller to a SellerGroup"?
        - Yes. Buyer frist -> onboard as seller by adding them into seller group. Can combined.
    - Can we preload Product + Category into db? YES!
- Question: update object: the entire object or just a single field?
    - Up to us. Each has pros and cons. You can have 100 methods to update individual code.
- Clarify the difference between Ghost/Virtual Proxy (Lazy Load). Hopefully make a decision on this with the team?
    - We just use whatever we want b/w Ghost and Virtual Proxy. . Don't use lazy initialisation though. Just use the Ghost.
- What's the difference between normal loading and lazy loading
    - "Load the next 10 users" that's just pagination + frontend logic. That's NOT lazy load.
    - Lazy loading means: for users, you just instantiate the user with their username ONLY. All other fields are NULL. Then, the Admin wants to read the user Ray, then they click into it, and the entire Ray object is loaded.
- Can you have a non-generic unit-of-work?
    - Generic class to do DataMapper.getMapper(Object)
    - Student.insert can be generic

```
public void commit() {
  for (DomainObject obj : newObjects) {
    DataMapper.getMapper(obj.getClass()).insert(obj);
  }
  for (DomainObject obj : dirtyObjects) {
    DataMapper.getMapper(obj.getClass()).update(obj);
  }
  for (DomainObject obj : deletedObjects) {
    DataMapper.getMapper(obj.getClass()).delete(obj);
  }
}
```

Sprint meeting:

- Use case we have covered:

Seller use cases

1. Create listing
   a. Auction (**Ray**)
   b. Fixed-price (**Ray** nearly)
2. View an order tied to seller group (**Edward** in progress)
3. Modify an order tied to seller group (**Edward** Order has foreign keys that we need to deal with -> reference)
   1. Decrease quantity
   2. Cancel an order

Buyer use cases

1. Search for a good (**Ray** - need help with frontend)
2. Purchase a good (Frontend firstMapper done. Just need to decide the frontend to decide the route)
3. View an order (**Edward** Done)
4. Modify an order (**Edward** sort of done - frontend)
   1. Increase/decrease quantity
   2. Cancel an order

Admin use cases

1. Onboard sellers (the same as add seller to a group - duplicate)
2. View all purchases/order (**Edward**)
3. Create seller group (**Haoxin**: frontend needs rework)
4. Add/remove seller to/from group (**Haoxin** remove needs further work)
5. Remove a listing (**Ray**)
6. Admin can view all users (**Yifei**)

- Things to do before the deadline:
  - What use cases we need to cover?

    - Transitions between pages?
    - Any more pattern decisions?
    - Error handling

Next week:

    - Diagrams
    - Deployment

Discussion

- See the **Agenda** section above, and the **To-Do** list section below.

To-Do list:

- Yifei:
    - Frontend work for editing the users profile
    - Login -> both frontend and backend
    - View all users page -> both frontend and backend
    - Authentication + authorisation framework
- Ray:
    - FixPriceListing mapper
        - Search
        - Create
        - Edit
    - Product mapper
    - ListingController
    - Frontend for listing
    -
- Edward:
    - Finish update/create order (controller + frontend)
    - Unit of work
- Haoxin:
    - Seller delete to get all sellers first
    - Frontend needs fixing
    - Lazy Load to be finalised
    - Rough sketches on frontend to be discussed on Friday
    - Exception handling (on Friday) Maybe not create our own exception - but at least SQL exceptions