

## Giao việc giai đoạn 2

Viết các lớp DAO

+ Tạo các lớp DAO ( Data Access Object ) để đóng gói toàn bộ logic truy cập CSDL. Mọi tương tác đến CSDL đều phải sử dụng DAO, không bao giờ được truy cập trực tiếp

+ Các yêu cầu

- Tất cả các lớp DAO phải được đặt trong package `com.microshop.dao`
- Tên lớp DAO = tên lớp model + DAO. Ví dụ `NguoiDung` là code model của `NGUOIDUNG` trong csdl thì DAO của nó sẽ là `NguoiDungDAO`
- Mọi lớp DAO phải sử dụng lớp `DBContext.java` ( xem trong package `com.microshop.context` ) để lấy Connection từ pool
- Phải sử dụng `PreparedStatement` cho các câu lệnh SQL để chống lỗi SQL injection
- Phải sử dụng `try-with-resources` để đảm bảo ko rò rỉ tài nguyên hệ thống
- Các phương thức nên throw `SQLException` để tầng servlet có thể bắt và xử lý
- Các phương thức DAO phải trả về các đối tượng Model tương ứng

+ Cập nhật: 26/10 – Khoa có đề xuất dùng interfaces và generic cho các class DAO. Sau khi cân nhắc thì t thấy rất hợp lý và đã quyết định:

- Tạo interface `ReadOnlyDAO` cho các DAO có ít nhất thao tác `getAll`: lấy tất cả dữ liệu của model tương ứng và `getById`: lấy một bản ghi tương ứng theo khóa chính
- Tạo interface `CrudDAO` là extends của `ReadOnlyDAO` cho các DAO có ít nhất 5 chức năng, `getAll`, `getById`, `insert`, `update`, `delete`
- `ReadOnlyDAO` và `CrudDAO` đã được cập nhật trên repo nhánh `develop` tại package `dao`
- Deadline mới = sáng t5 = sáng 30/10

+ Giao việc chi tiết mới ( 26/10 )

- 🚩 **Danh sách các phương thức chỉ là tối thiểu, đề xuất thêm nếu ae thấy cần**
- 🚩 **Các class phải thực hiện implements đúng như đề xuất**
- 🚩 **Implements đã đảm bảo có các hàm trong interface, các hàm liệt kê dưới đây là các chức năng đặc biệt của DAO đó**

+ *Khoa: NguoIDungDAO, HangThanhVienDAO, DanhMucDAO*

- ✚ NguoIDungDAO implements CrudDAO<NguoiDung, Integer>
  - public NguoiDung getByTenDangNhap(String tenDangNhap) throws SQLException
  - public NguoiDung getByEmail(String email) throws SQLException

- ✚ HangThanhVienDAO implements ReadOnlyDAO<HangThanhVien, Integer>
  - Chỉ cần 2 hàm của interface

- ✚ DanhMucDAO implements ReadOnlyDAO<DanhMuc, Integer>

+ *Hải: TaiKhoanDAO, TaiKhoanLienQuanDAO, TaiKhoanFreeFireDAO, TaiKhoanRiotDAO*

- ✚ TaiKhoanDAO implements CrudDAO<TaiKhoan, Integer>
  - public List<TaiKhoan> getByMaDanhMuc(Integer maDanhMuc) throws SQLException
  - public List<TaiKhoan> getByTrangThai(String trangThai) throws SQLException
  - public void updateTrangThai(Integer maTaiKhoan, String trangThaiMoi) throws SQLException

- ✚ TaiKhoanLienQuanDAO implements CrudDAO<TaiKhoanLienQuan, Integer>




- ✚ TaiKhoanFreeFireDAO implements CrudDAO<TaiKhoanFreeFire, Integer>

- ✚ TaiKhoanRiotDAO implements CrudDAO<TaiKhoanRiot, Integer>

- ❖ Ở đây đối với TaiKhoanLienQuanDAO, TaiKhoanFreeFireDAO, TaiKhoanRiotDAO thì các hàm getAll, getById, insert, update, delete về mặt ý nghĩa vẫn dễ hiểu nhưng về mặt cài đặt thì sẽ có sự phức tạp.
- ❖ T sẽ đưa ra gợi ý cách làm của TaiKhoanLienQuan và Hải sẽ tham khảo để làm 2 cái còn lại.
  - getAll: mục đích là trả về tất cả các TaiKhoanLienQuan có trong CSDL, vậy nên ở đây mình sẽ cần kéo tất cả bản ghi trong TAIKHOAN\_LIENQUAN và phải thực hiện JOIN để lấy thông tin từ lớp cha (MaDanhMuc, GiaGoc, GiaBan, TrangThai, DiemNoiBat, LuotXem, NgayDang ). Khi này các bản ghi mới chứa đầy đủ thông tin của TaiKhoanLienQuan.
  - getById: làm tương tự như getAll, chỉ khác là mình JOIN cho một bản ghi

- insert: Đầu tiên cần bóc tách các thuộc tính của TaiKhoan trước, thực hiện gọi insert của TaiKhoanDAO, sau khi có được maTaiKhoan trả về thì mới có thể tạo TaiKhoanLienQuan rồi triển khai insert tiếp vào TAIKHOAN\_LIENQUAN
- update: làm tương tự, bóc tách các thuộc tính của TaiKhoan, thực hiện gọi update của TaiKhoanDAO trước sau đó thực hiện logic update vào TAIKHOAN\_LIENQUAN
- delete: Theo cơ chế “ON DELETE CASCADE” của khóa ngoại định nghĩa trong database\_schema thì mình sẽ chỉ cần gọi delete của TaiKhoanDAO thì tự động bản ghi của TAIKHOAN\_LIENQUAN tương ứng cũng sẽ bị xóa

#### + Thành: AnhTaiKhoanDAO, DonHangDAO, GameSteamDAO

-  AnhTaiKhoanDAO không implement interface
  - public List<AnhTaiKhoan> getByMaTaiKhoan (Integer maTaiKhoan) throws SQLException: Lấy tất cả ảnh của một tài khoản
  - public Integer insert (AnhTaiKhoan newImg) throws SQLException: Thêm một bản ghi ảnh, trả về maAnh nếu thành công, ngược lại trả về null
  - public boolean deleteByMaAnh(Integer maAnh) throws SQLException
  - public boolean deleteByMaTaiKhoan(Integer maTaiKhoan) throws SQLException: Xóa tất cả ảnh của một tài khoản
-  DonHangDAO implements CrudDAO<DonHang, Integer>
  - public List<DonHang> getByMaNguoiDung(Integer maNguoiDung) throws SQLException
  - public DonHang getByMaTaiKhoan(Integer maTaiKhoan) throws SQLException
  - public boolean updateTrangThai(Integer maDonHang, String trangThaiMoi, LocalDateTime thoiGianMua) throws SQLException
-  GameSteamDAO implements CrudDAO<GameSteam, Integer>
  - public List<GameSteam> fastGetAll() throws SQLException: Lấy tất cả trừ MoTaGame phục vụ mục đích hiển thị bên ngoài, khi nào khách bấm chi tiết thì mới cần dùng getAll gốc để kéo cả MoTaGame lên ( getAll theo interface là lấy tất cả GameSteam và các thuộc tính của chúng, trong đó bao gồm cả MoTa nên sẽ chậm )
  - public String getMoTaGame (Integer maGameSteam): trong trường hợp muốn né getAll mà chỉ lấy đúng mô tả thôi nhằm tăng tốc độ

+ Hưng: Còn lại