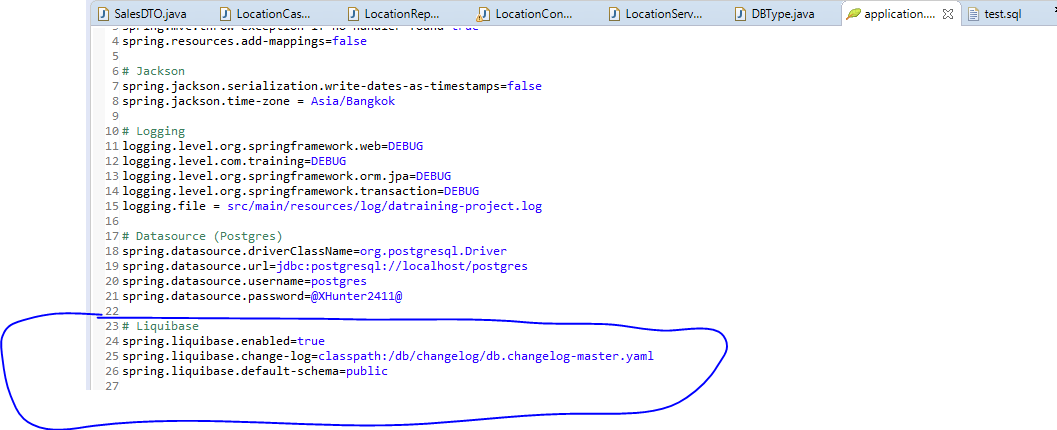
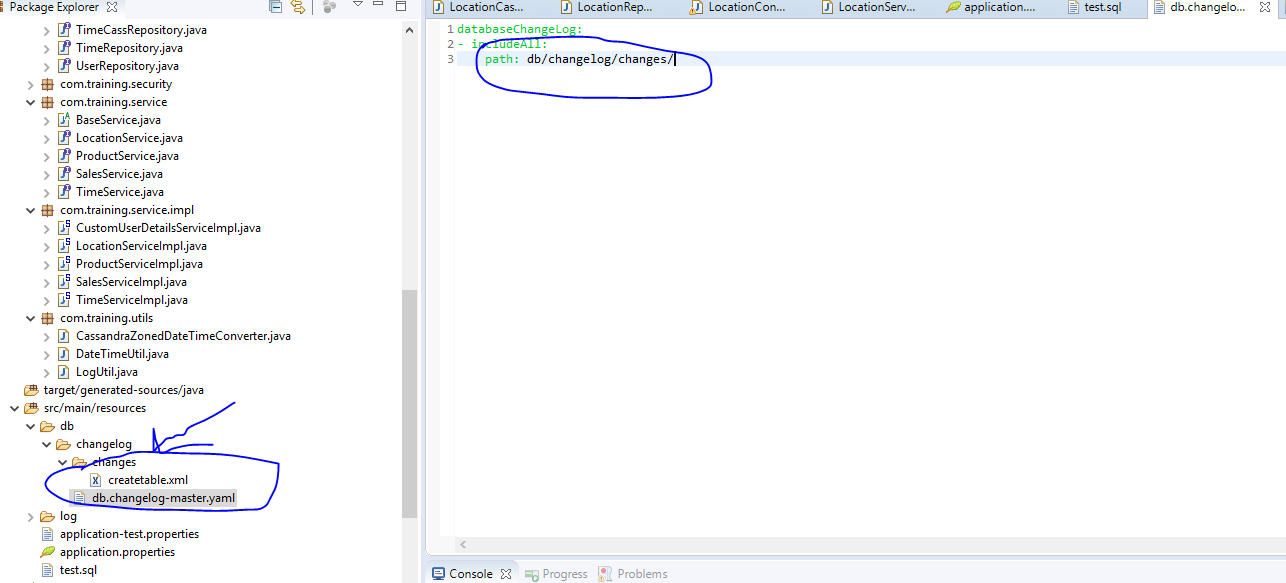
Tạo ứng dụng web theo cấu trúc 3 lớp

Lấy dữ liệu từ Cassandra và lưu xuống Postgresql

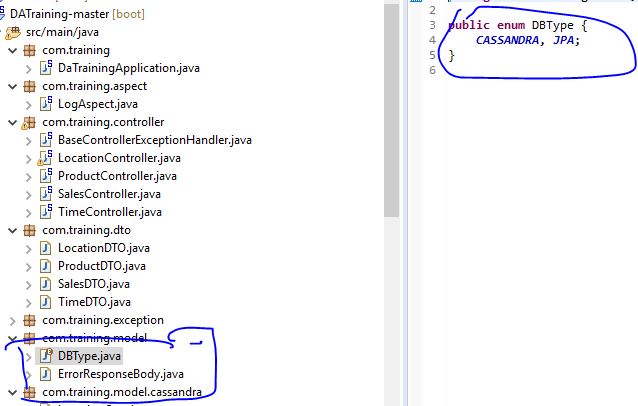
Sử dụng spring boot, maven, thymeleaf, spring data JPA

**Using liquibase to create database Postgres structure and initial needed data**



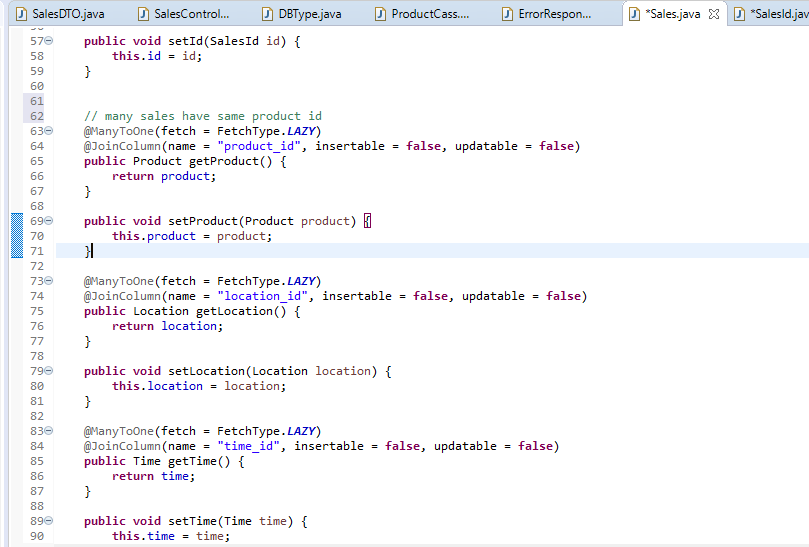


**Viết code có thể tái sử dụng được cho unit test, such as API request mapping url, enums values for type object..**



Three layer (controller, service, repository/dao).

Sử dụng LAZY Strategy cho object mapping.



### LAZY:

**LAZY** nói với **Hibernate** rằng, hãy tải dữ liệu một cách lười biếng.

Chẳng hạn bạn có một đối tượng **Account**, và gọi phương thức **getOpenBranch()** nó trả về một đối tượng **Branch**, đối tượng**Branch**chỉ trường (field) **branchId** được gán giá trị, các trường khác thì không.   
  
Thực tế **hibernate**chưa tải dữ liệu từ bản ghi tương ứng của bảng **BRANCH** vào đối tượng này. Nó chỉ thực hiện truy vấn dữ liệu khi bạn làm gì đó với đối tượng **Branch** vừa có được, chẳng hạn gọi phương thức **branch.getName()**.

### EAGER:

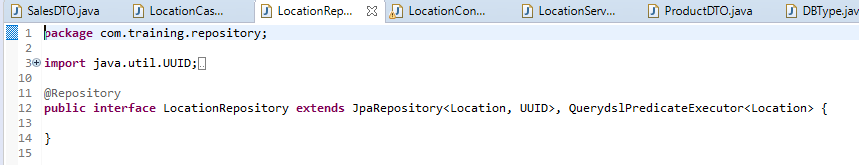
**EAGER**nói với **Hibernate**hãy truy vấn toàn bộ các cột của bảng liên quan.

Chẳng hạn bạn có đối tượng **Account**, và gọi phương thức **getOpenBranch()** trả về đối tượng **Branch**đã có sẵn các giá trị cho các trường ( **name**, **address**, ...). Thực tế dữ liệu của nó có được cùng trong 1 lần truy vấn với bảng **Account**.

*Bạn nên sử dụng****LAZY****thay vì****EAGER****vì lý do hiệu năng chương trình.*

<https://o7planning.org/vi/10225/tra-cuu-java-hibernate>

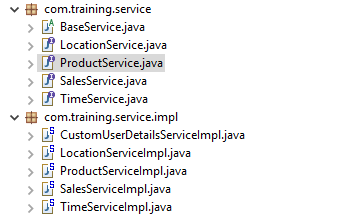
Sử dụng Spring repository để cài đặt lớp DAO

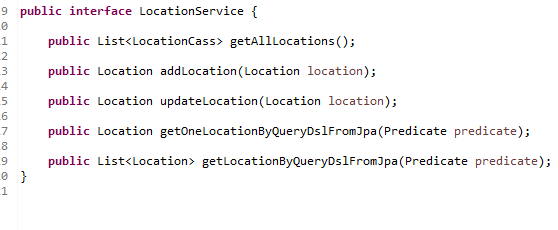


Đối với controller, chuyển đổi một model cụ thể thành DTO object trước khi trả về dữ liệu cho client.



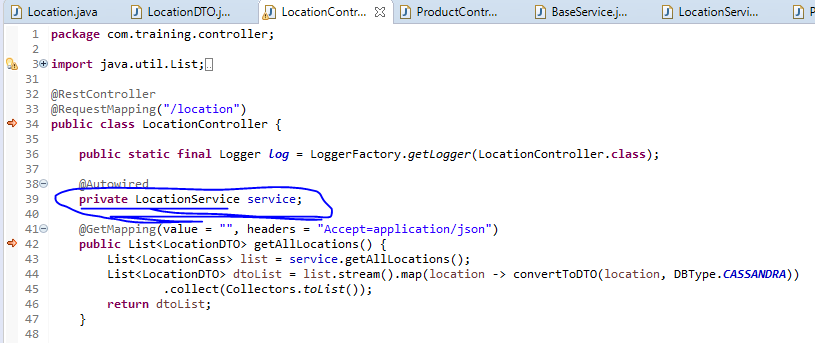
Khai báo một interface cho mỗi class service, sử dụng interface trong controller thay vì một lớp service cụ thể.



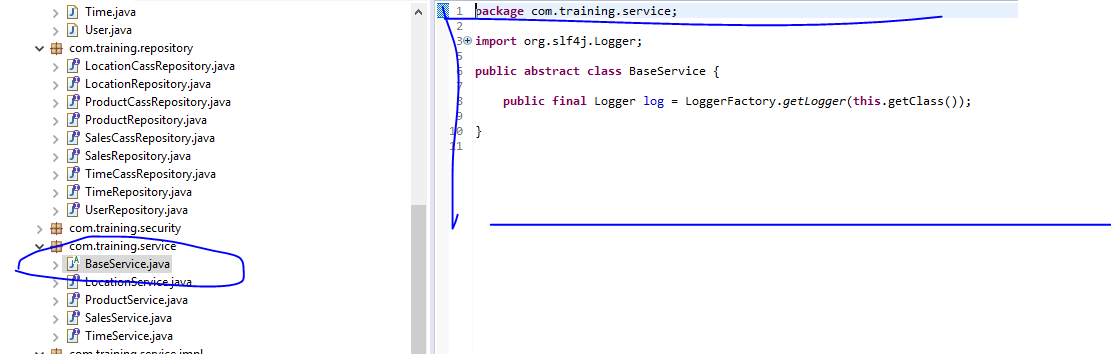




Sử dụng interface thay vì service trong controller.

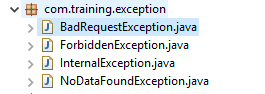


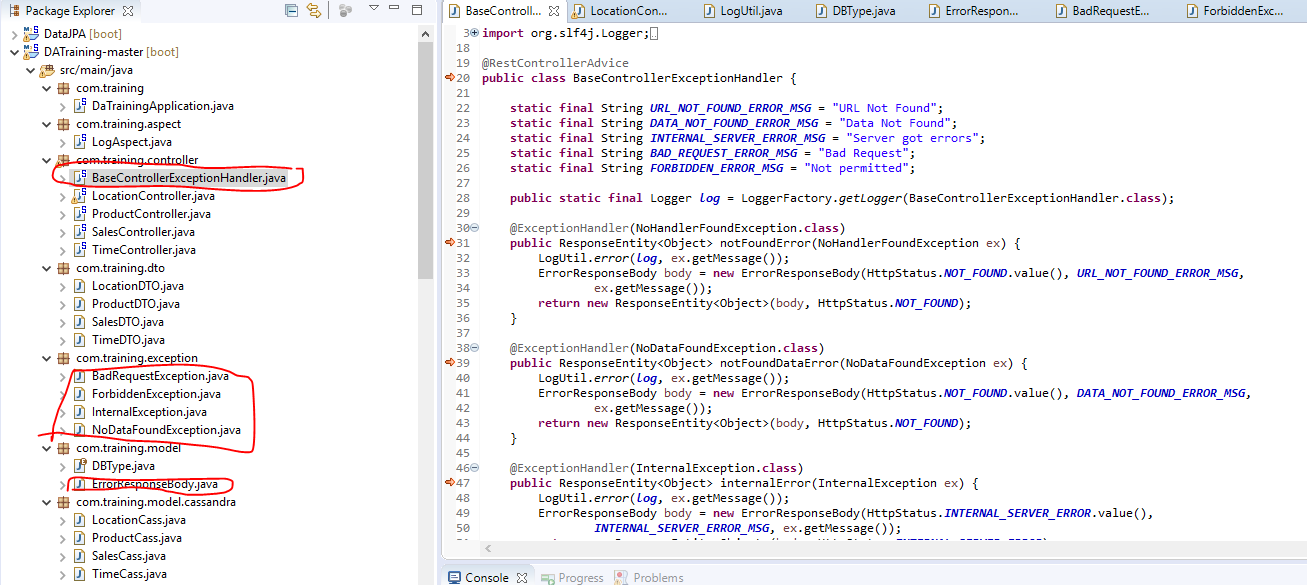
Khai báo một BASE SERVICE để quản lý các thuộc tính tồn tại trong tất cả các SERVICE con



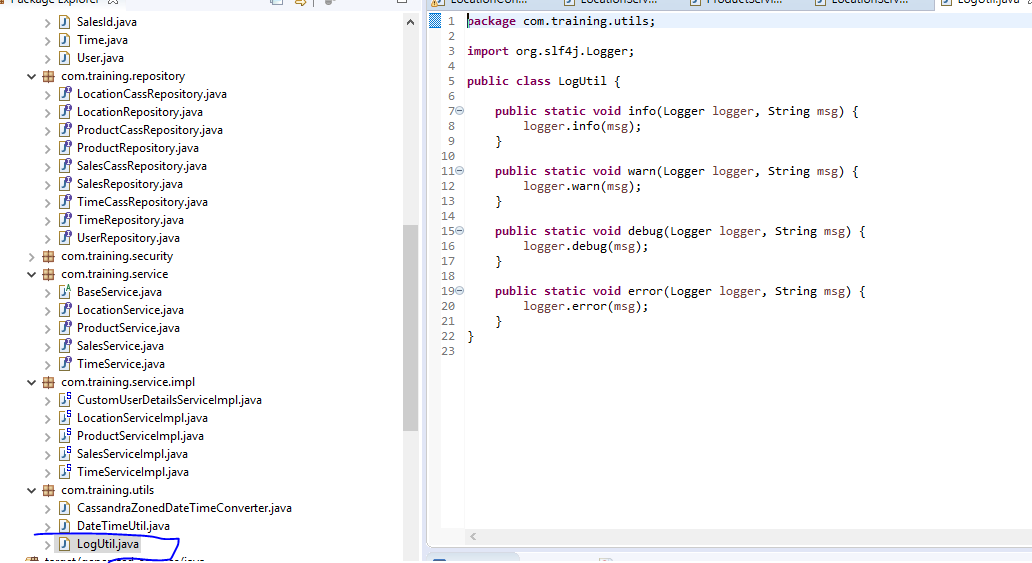
Declare a base controller to catch all exception-(@ExceptionHandler)

* NOT FOUND DATA EXCEPTION - given param not exist in the database,
* INTERNAL EXCEPTION - exception that occurred somewhere in the code,
* BAD REQUEST - given param is not matched to controller,
* FORBIDDEN - not authentication) and corresponding http status code to the client.





Khai báo LogUtil để debug, method lỗi, tạo logger cho từng controller cụ thể, dịch vụ cụ thể, sau đó sử dụng LogUtil bằng cách vượt qua bản ghi nhật ký cụ thể và / hoặc nhật ký thông báo / nhật ký thông báo lỗi (ex.getErrorMessage) vào tiện ích.

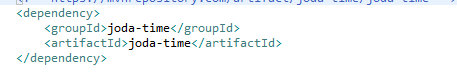


Đối với giá trị datetime, sử dụng Joda time để lấy giá trị với múi giờ UTC, khai báo một DateTimeUtil để có thể sử dụng lại và giảm thiểu thay đổi khi cần thiết

<http://www.joda.org/joda-time/>

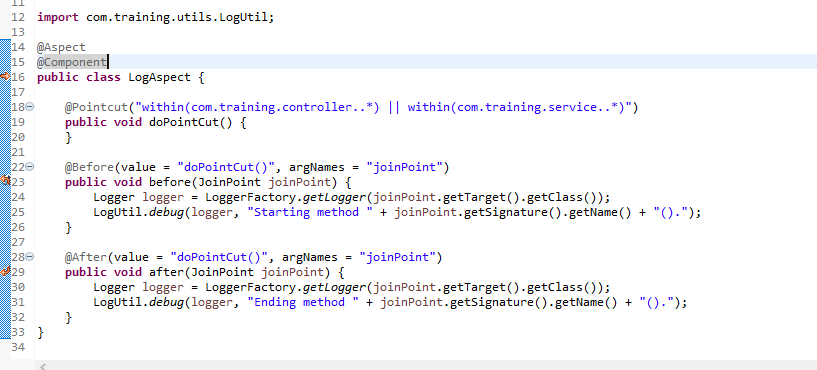
<http://www.joda.org/joda-time/userguide.html>

<https://viblo.asia/p/tim-hieu-aop-voi-aspectj-BYjv44VwvxpV>



Đối với mỗi method, log debug ở đầu và cuối của method.

Use library Aspectj



Đối với debug API runtime, tìm ra cách để ghi nhật ký thông báo gỡ lỗi và / hoặc thông báo lỗi cho khách hàng bằng cách lưu trữ thư trong response header (eX: log detail trace header).

