1. Feign có cache để lưu trữ dữ liệu những lần gọi service trước đó. Vì vậy đừng lấy làm khó hiểu, khi service đích hoặc cân bằng tải tắt đi rồi, mà service A gọi api trước đó vẫn có thể lấy được thông tin của service B (thực ra đây là dữ liệu cũ, đã được lưu trong cache).
2. spring-boot-maven-plugin đã chứa maven-compiler-plugin.
3. Khi project chứa cả 2 dependency là:

<groupId>org.mapstruct</groupId>

<artifactId>mapstruct-processor</artifactId>

Và:

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

Thì trong file .pom ta phải khai báo như sau:

1. <build>
2. <plugins>
3. <plugin>
4. <groupId>org.apache.maven.plugins</groupId>
5. <artifactId>maven-compiler-plugin</artifactId>
6. <version>3.8.1</version>
7. <configuration>
8. <annotationProcessorPaths>
9. <path>
10. <groupId>org.mapstruct</groupId>
11. <artifactId>mapstruct-processor</artifactId>
12. <version>${org.mapstruct.version}</version>
13. </path>
14. <path>
15. <groupId>org.projectlombok</groupId>
16. <artifactId>lombok</artifactId>
17. <version>${org.projectlombok.version}</version>
18. </path>
19. <path>
20. <groupId>org.projectlombok</groupId>
21. <artifactId>lombok-mapstruct-binding</artifactId>
22. <version>${lombok-mapstruct-binding.version}</version>
23. </path>
24. </annotationProcessorPaths>
25. </configuration>
26. </plugin>
27. </plugins>
28. </build>

🡺 Nếu không sẽ dẫn đến các lỗi thiếu lombok, kiểu như: không có phương thức get và set khi sử dụng @Getter và @Setter.

1. Cài đặt route cho gateway ghi nhớ:

Spring.cloud.gateway.routes[n].uri=lb://{service-id}

* Service-id phải trùng với service-id đã đăng ký với Registration. Nếu không bộ định tuyến sẽ không phát hiện ra và thông báo lỗi 503 – Server Unavailable và log gateway sẽ ghi là: No service available for server: <tên\_sai>

1. BigDecimal.devide(BigDecimal).setScale() khác BigDecimal.devide(BigDecimal, scale, RoundingMode) ở chỗ:

Biểu thức 1 là chia xong -> nhận về BigDecimal rồi trình bày lại BigDecimal đó theo công thức ta quy định.

Biểu thức 2 là vừa chia vừa trình bày lại BigDecimal.

* Biểu thức 1 có thể bị bug khi phép chia có phần thập phân dài vô cực và không thể làm tròn.

1. @AfterReturnning

Please note that it is not possible to return a totally different reference when using after returning advice. (không thể thay đổi giá trị trả về).

[Declaring Advice :: Spring Framework](https://docs.spring.io/spring-framework/reference/core/aop/ataspectj/advice.html#aop-advice-after-returning)

* Muốn thay đổi giá trị trả về ta sử dụng @Around và đối số ProceedingJoinpoint.process() thì mới thay đổi được giá trị trả về.
* **Lưu ý**: chỉ thay đổi được nội dung giá trị trả về, chứ không thay đổi được **kiểu** trả về.

9. Lỗi:

object references an unsaved transient instance - save the transient instance before flushing : vanhoang.project.entity.CommentEntity.commentor -> vanhoang.project.entity.UserEntity

🡺 Lý do xảy ra lỗi này là do: Đối tượng liên kết chưa có trong database;

10. Phép toán với bit: Not x (~x) sẽ thực hiện đảo bit 0 -> 1 và 1 -> 0. Kết quả của số sau đảo được tính bằng phép toán: **bù 2**. (Tất cả số âm **nhị phân** trong java đều chuyển sang hệ cơ số 10 bằng phép bù 2)

10. Trạng thái của entity và vấn đề sử dụng phương thức save() và findAll() của JpaRepository<T, ID>() :

Lời đầu tiên, ở đây ta đang xét đến cấu trúc JPA được Hibernate cài đặt dựa trên thiết kết JPA.

**Trạng thái của entity**:

* Hibernate quản lý các entity được **liên kết** và entity có các trạng thái thể hiện tình trạng liên kết giữa entity và dữ liệu trong database.

**New (mới được tạo ra, transient, chưa được lưu xuống trong database):**

* Entity vừa được **khởi tạo (gọi contructor xong)**, chưa được liên kết với dữ liệu trong database.
* Để chuyển sang trạng thái **persistent** (bền bỉ), ta sử dụng phương thức persist() của entityManager hoặc các phương thức gọi đến nó gián tiếp khác

**Persistent (đã được liên kết với dữ liệu trong database, và vì xác định là dữ liệu mới 🡪 lưu mới, và được quản lý bởi Persistence Context):**

* Lúc này mọi thay đổi trên entity sẽ được ánh xạ xuống dữ liệu trong database.
* **Lưu ý:** Quá trình ánh xạ không diễn ra ngay tức thì mà sẽ đợi đến khi gọi phương thức **flush()** của **Session.** Hãy nghĩ như thế này: Để ánh xạ entity 🡪 dữ liệu, ta cần một bộ biên dịch, dịch các thay đổi thành SQL và “bộ biên dịch” này để tăng hiệu năng nó sẽ không tiến hành gửi luôn câu query đó xuống database mà đợi đến khi số lượng câu SQL đủ nhiều rồi mới gửi một thể như **Spring Batch**.

**Detach (Xóa liên kết, thoát khỏi sự quản lý của Persistence Context):**

* Mọi sự thay đổi trên entity từ giờ trở đi sẽ không còn được ánh xạ sang dữ liệu.
* Để tạo liên kết lại với dữ liệu ta có 2 cách:
  1. Sử dụng phương thức update() – được liên kết với 1 Session: phương thức này sẽ tạo lại liên kết giữa entity với dữ liệu (với điều kiện là dữ liệu chưa liên kết với entity khác).
  2. Sử dụng phương thức merge(): sao chép dữ liệu của entity sang một entity đang có liên kết với dữ liệu.

**Remove (Xóa entity 🡪 xóa dữ liệu)**

* Xóa entity, nhưng chưa xóa dữ liệu ngay mà chờ phương thức flush() của Session được gọi.

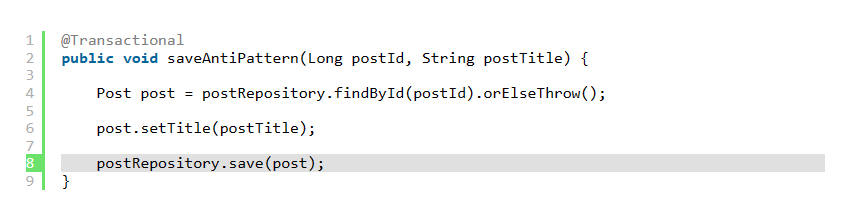
**Lý do không nên dùng phương thức save() của JpaRepository:**

Như đã đọc ở bên trên, thực tế không có cái gì gọi là chèn dữ liệu mà tất cả chỉ là sự **cập nhật** thông tin từ entity 🡪 sang dữ liệu.

Mà phương thức save() lại làm 2 việc:

1. Gọi phương thức persist() của entityManger nếu dữ liệu chưa có trong database hay entity được xác định là entity mới thông qua isNew() của EntityInformation.
2. Gọi phương thức merger() của entityManger nếu dữ liệu đã có trong database hay entity được xác định là entity cũ thông qua isNew() của entityInformation.

🡺 Nếu 1 entity cũ hay đã có liên kết với dữ liệu mà tiến hành gọi save() 🡪 dẫn đến thừa bước làm việc và tiến hành merge lại dữ liệu không cần thiết. Như trường hợp dưới đây là không nên:



**Lý do tại sao không nên sử dụng phương thức findAll() của JpaRepository:**

* Bởi vì trừ trường hợp bảng vô cùng đặc biệt giành riêng cho 1 cá nhân, 1 đối tượng, … còn không gần như ta không bao giờ cần lấy toàn bộ dữ liệu từ 1 bảng rồi đưa thẳng nên FE mà sẽ tiến hành lọc tiếp ở BE 🡪 dữ liệu trao đổi giữa BE và database là vô cùng lớn 🡪 tốn tài nguyên.