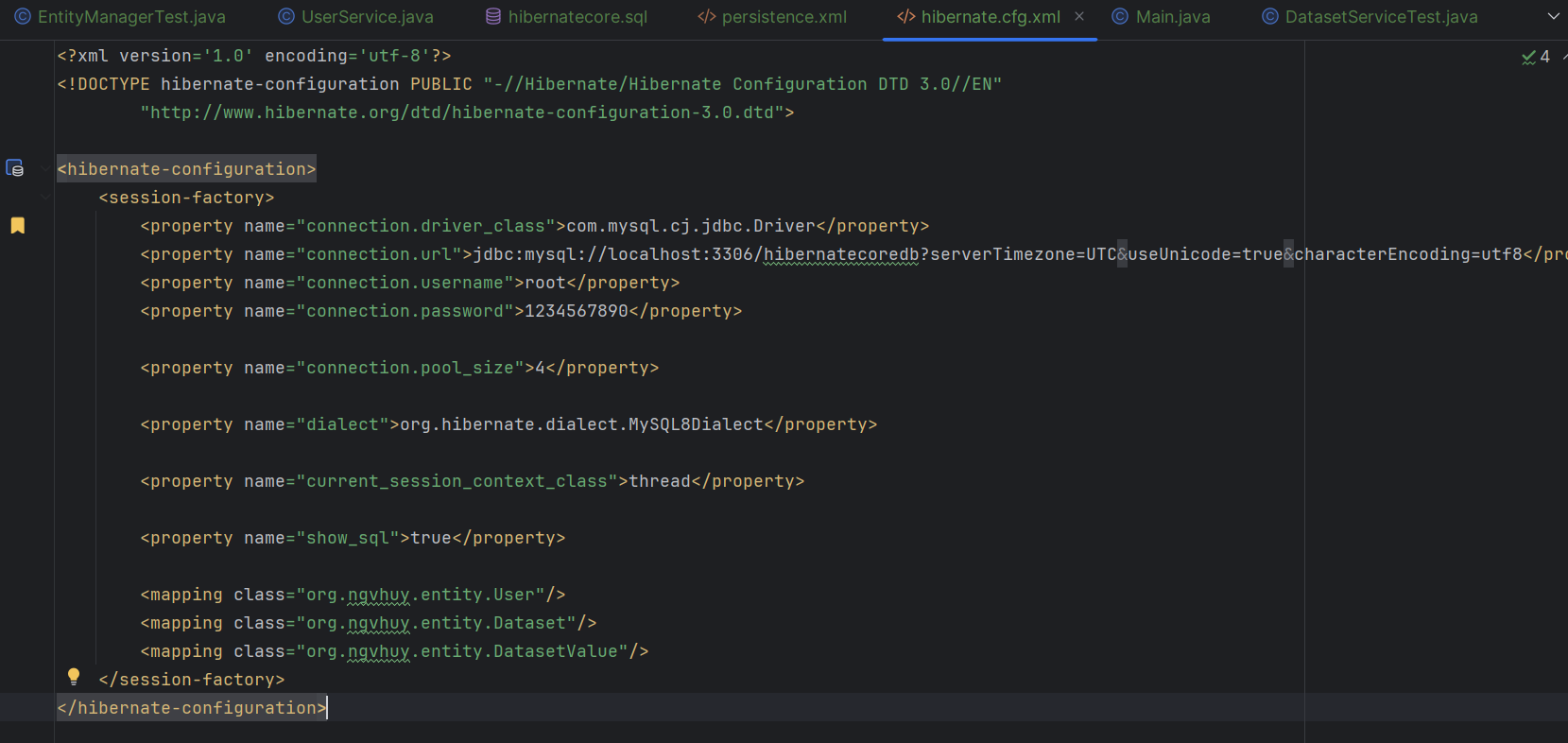
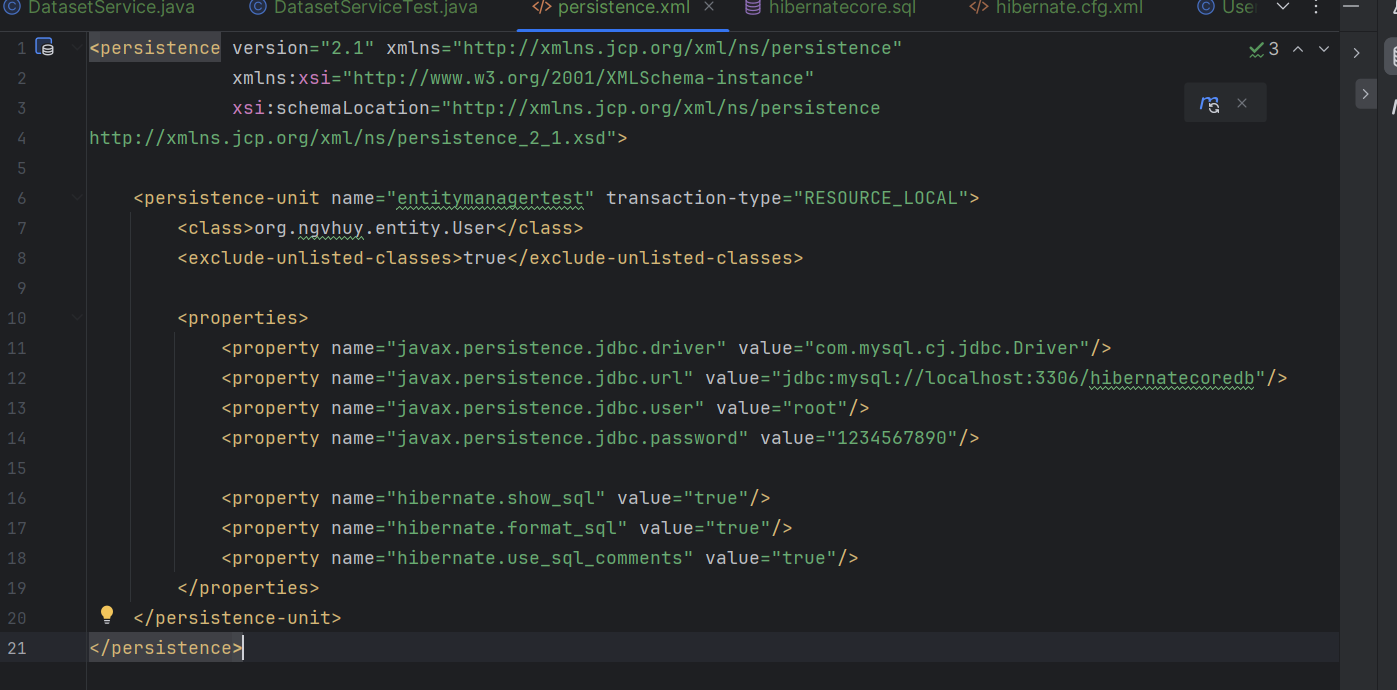
1. Lời mở đầu:

* Các bài lab dùng để thực hành, từ đó có cái nhìn sâu về hibernate
* File doc này chỉ mô tả một phần những gì đã tìm hiểu và test, có những thứ chưa thể diễn tả trong doc vì còn phải dành thời gian cho việc tìm hiểu thứ khác
* File doc này chưa phải kiến thức cuối cùng, nghĩa là vẫn sẽ được bổ sung về sau
* File doc này chỉ mô tả các bài lab, không mô tả về các kiến thức khác (kiến trúc hibernate, vòng đời entity, kiến trúc jpa,…), chúng sẽ được mô tả ở file doc khác

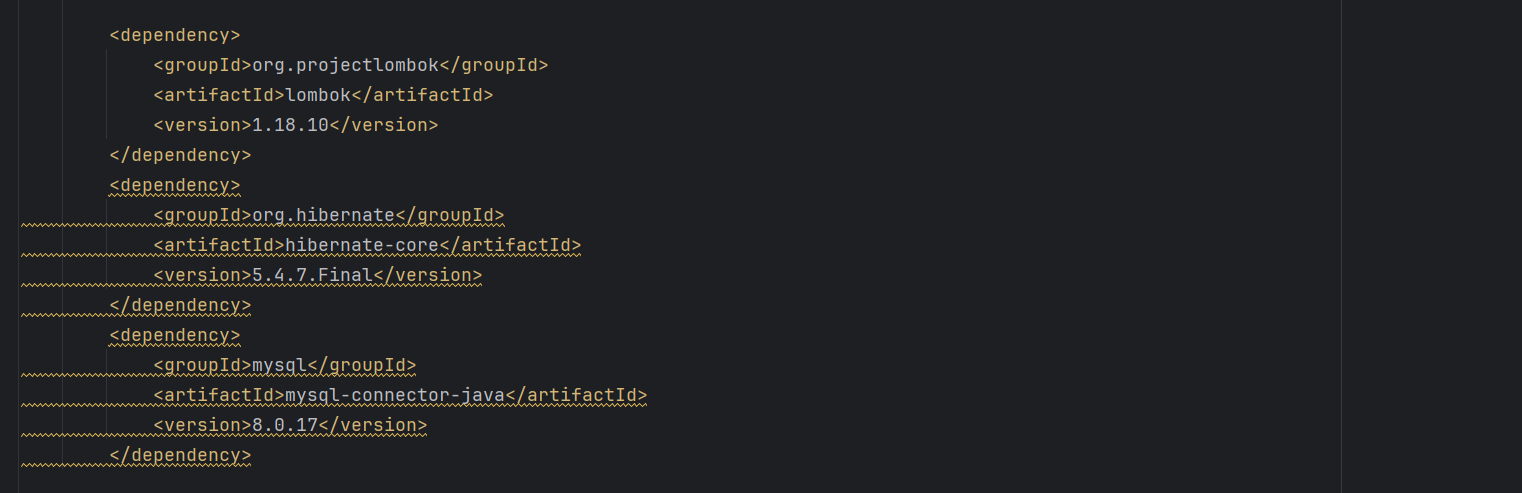
1. Thực hành:
   1. Cấu hình:
      1. hibernate.cfg.xml



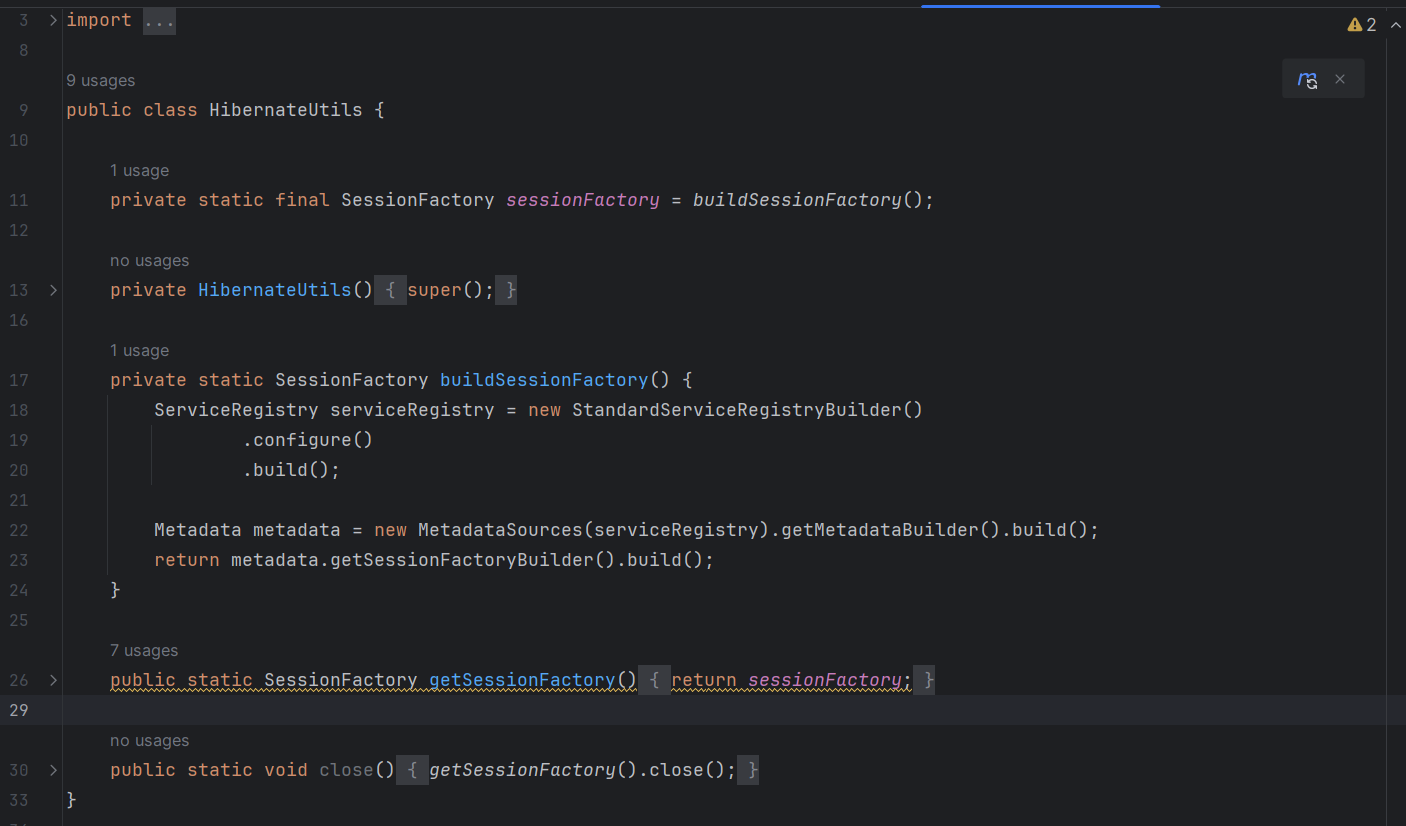
* + 1. persistence.xml (với EntityManager)



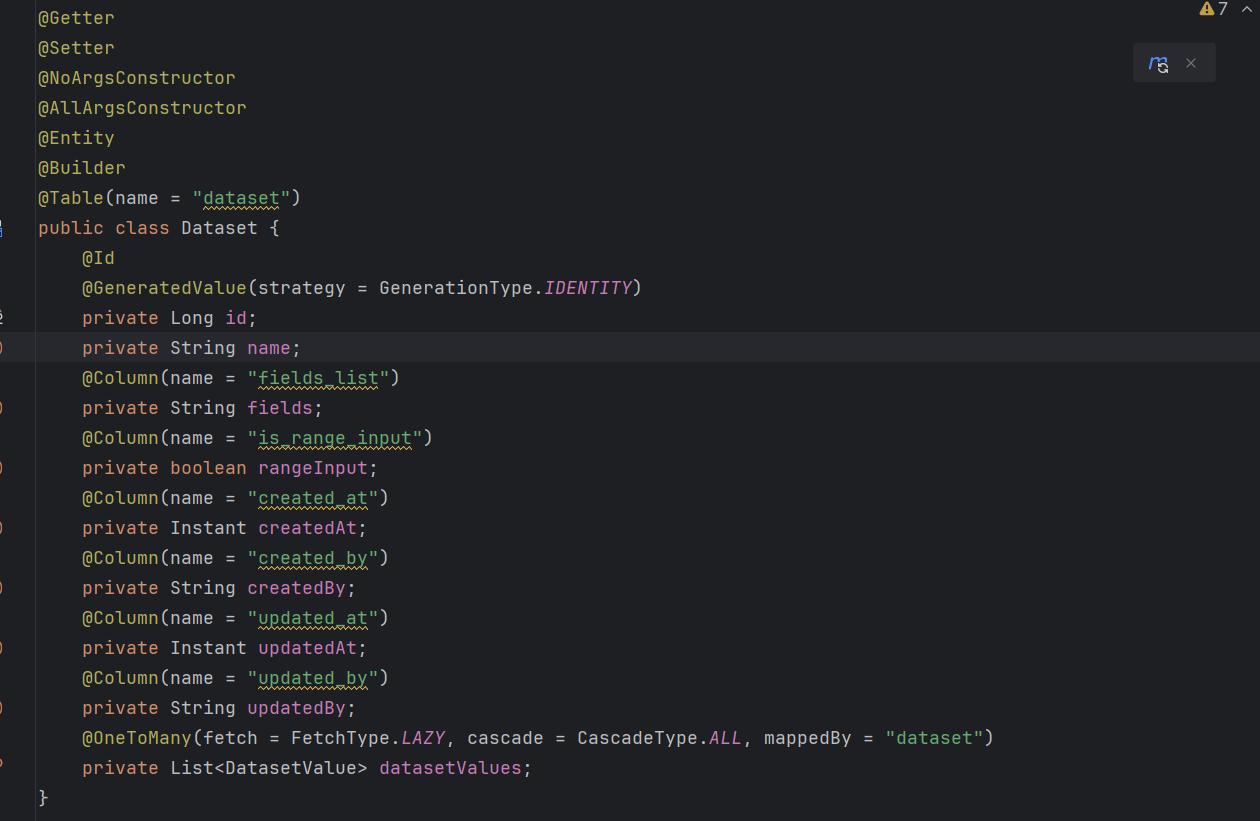
* + 1. pom.xml

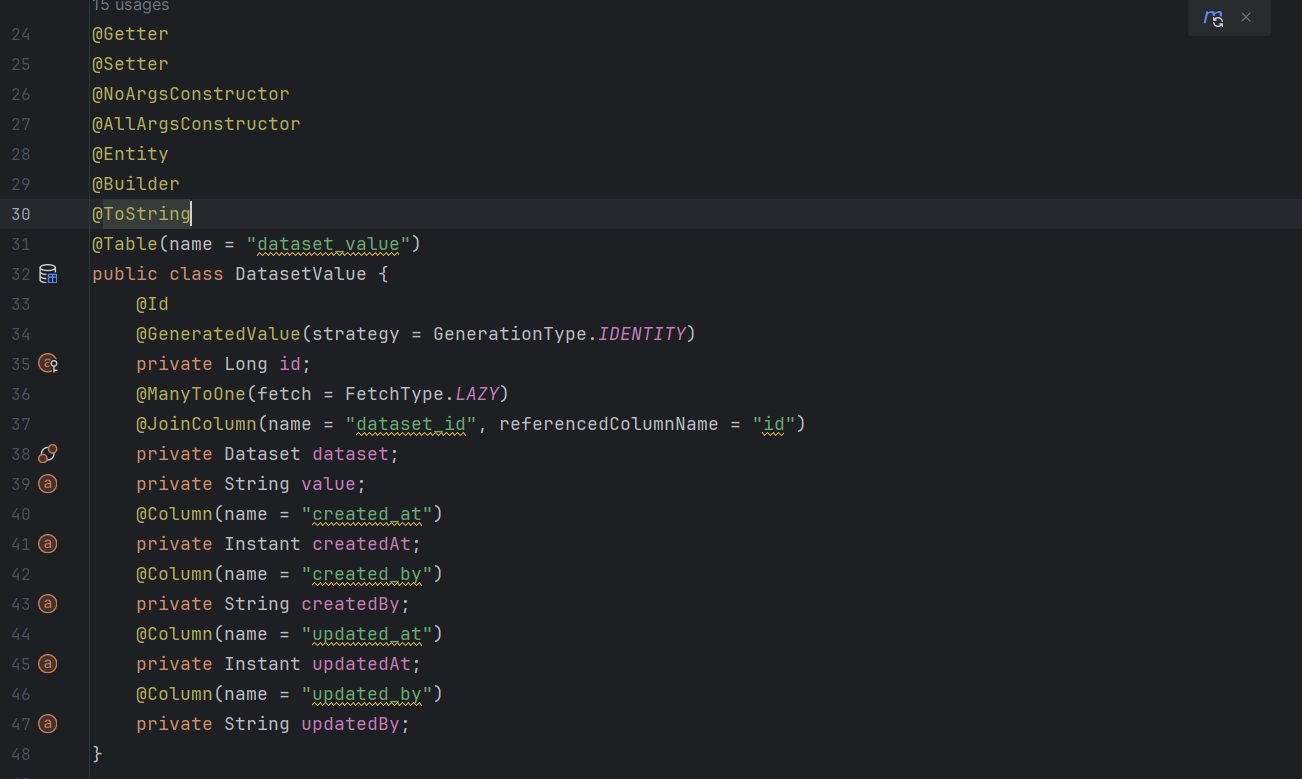


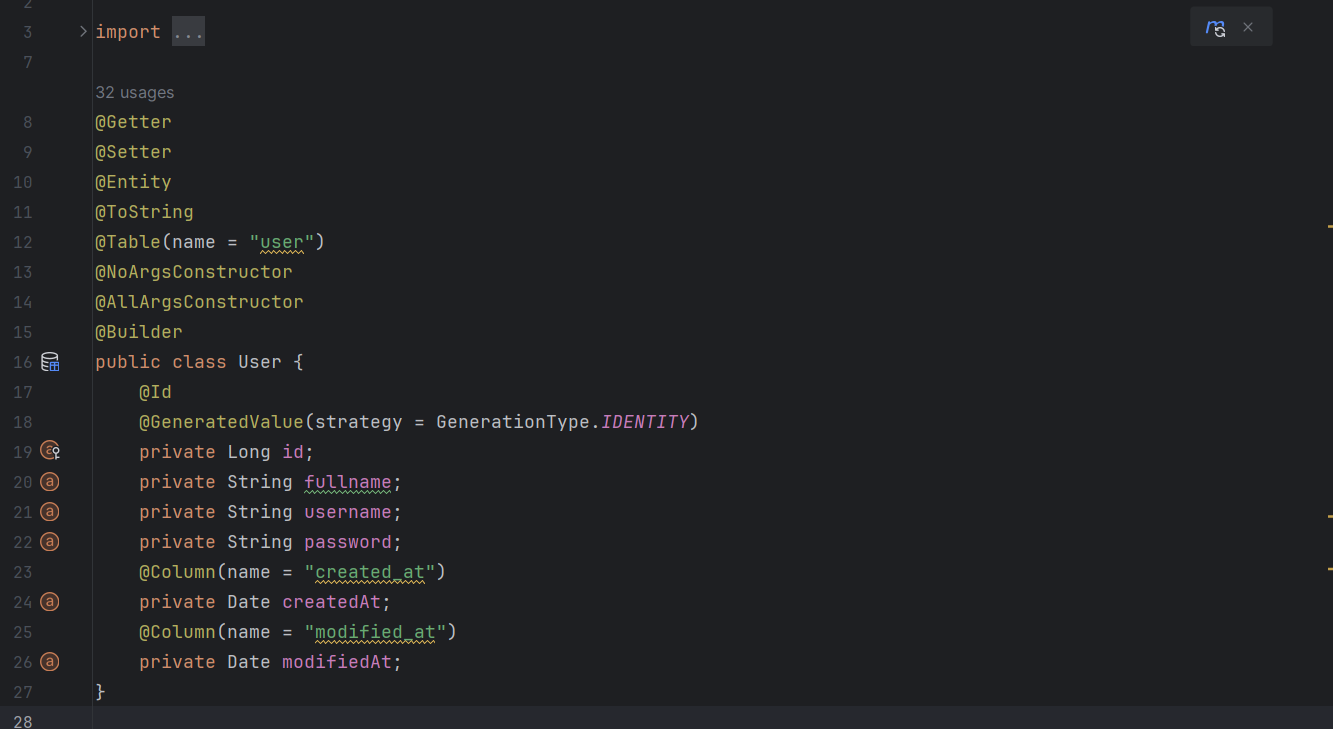
* + 1. org.ngvhuy.config.HibernateUtils



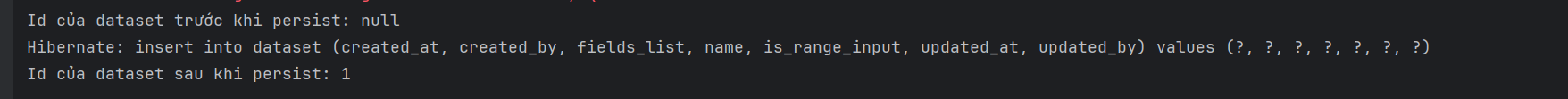
* + 1. Các class entity:







* 1. Thực hành:
     1. Save

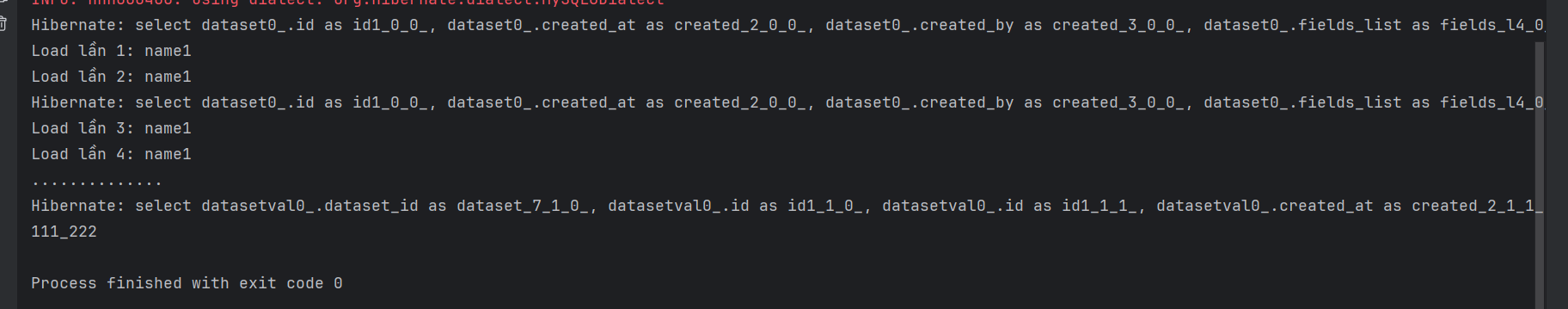


Kịch bản:

* Tạo mới 1 entity
* Kiểm tra id của nó
* Persist entity đó
* Kiểm tra id sau đó

Nhận xét:

* Trước khi persist thì entity ở trạng thái transient và chưa có id, sau khi persist thì entity ở trạng thái persistence, ở trạng thái này dữ liệu được ánh xạ tới 1 row trong db và được gán id.
* Khi entity được persist thì entity được lưu vào first level cache, kể từ đó việc load() theo id sẽ không truy vấn đến db (nếu chưa clear cache)
  + 1. Clear cache, Lazy load – Load entity cha xong load entity con



Kịch bản:

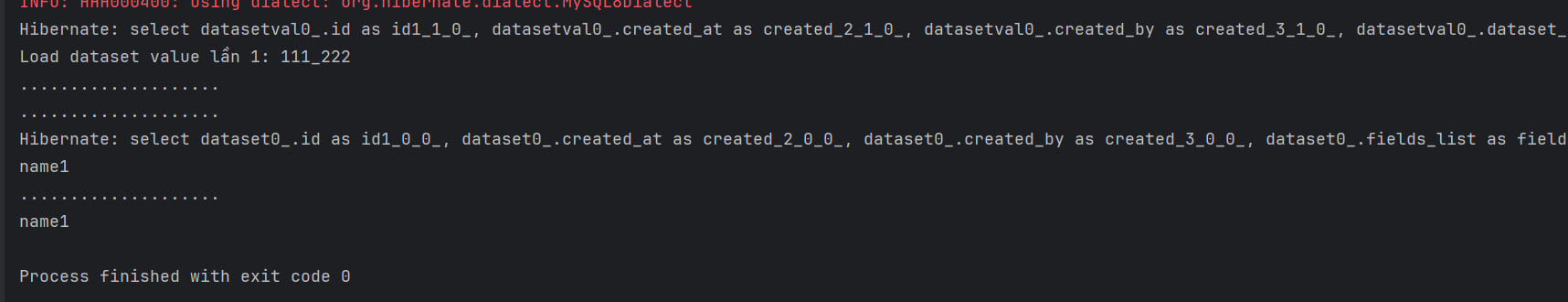
* Load entity cha lần 1
* Load entity cha lần 2
* Xóa cache
* Load entity cha lần 3
* Load entity cha lần 4
* Từ entity cha gọi entity con
* Gọi thuộc tính của entity con

Kết quả:

* Khi load entity cha lần 1 thì thực hiện truy vấn tới db, và entity đó được lưu vào cache
* Khi load entity cha lần 2 thì dữ liệu được lấy từ cache
* Sau khi xóa cache, ta thực hiện load entity cha lần 3, khi này dữ liệu được lấy từ db
* Khi ta từ entity cha được load ra mà gọi tới entity con của nó, chương trình vẫn không truy vấn tới db
* Khi ta gọi thuộc tính của entity con, chương trình khi này thực hiện truy vấn sql tới db

Nhận xét:

* Khi load entity từ trong db, chúng sẽ được lưu vào first level cache
* Sau khi clear cache thì việc load entity phải thực hiện lại truy vấn tới db
* Đối với Lazy load, việc gọi .getDatasetValue() chương trình **chưa** load dữ liệu từ db, mà phải thực hiện các action trên datasetValue thì mới dẫn tới truy vấn tới db.
  + 1. Lazy load - Load entity con xong load entity cha:



Kịch bản:

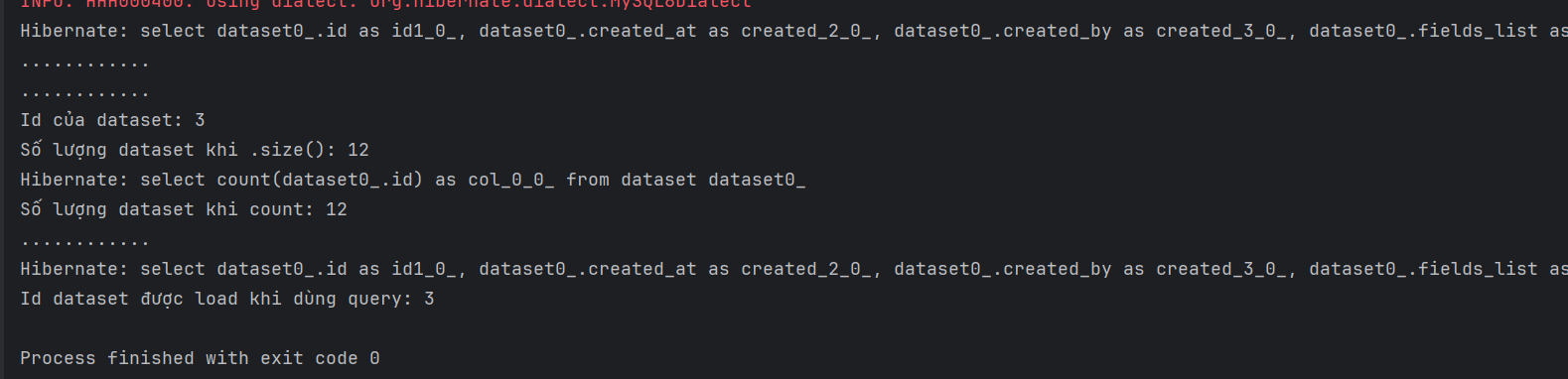
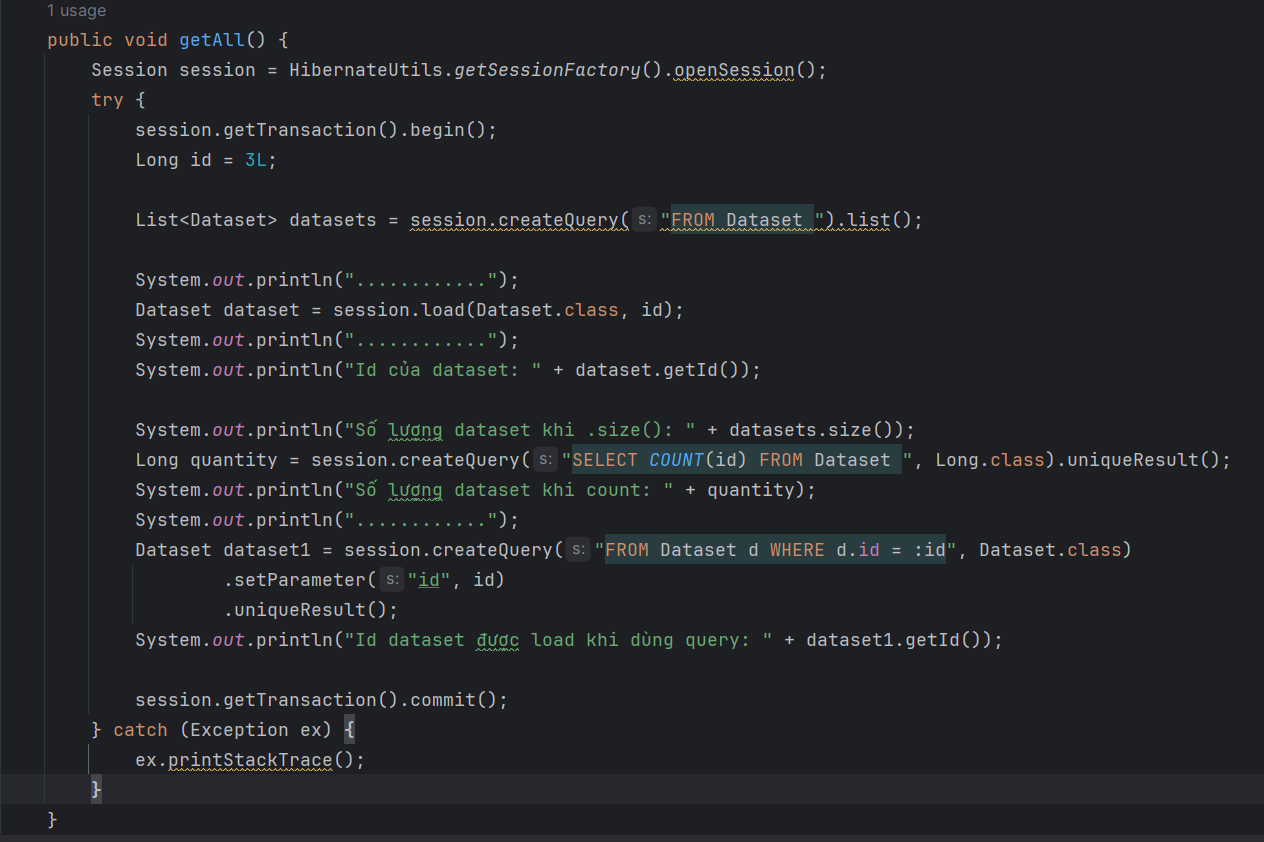
* Load entity con
* Lấy ra entity cha từ entity con
* Load lại entity cha đó với phương thức load
* Gọi phương thức getName() từ entity cha

Kết quả:

* Chương trình thực hiện truy vấn tới db khi load entity con lần 1
* Chương trình không thực hiện truy vấn tới db khi lấy entity cha từ entity con cũng như load entity cha bằng phương thức load
* Chương trình thực hiện truy vấn tới db khi gọi đến thuộc tính của entity cha

Nhận xét:

* Khi load entity con, session thực hiện truy vấn tới db
* Sau đó từ entity con lấy ra entity cha thì sẽ lấy ra một entity “ảo”, entity này **chỉ có id**, dẫn tới việc load entity cha theo id sẽ không truy vấn tới db, nhưng khi get các thuộc tính của entity cha thì lúc đó sẽ thực hiện truy vấn tới db.
  + 1. List All, createQuery



Kịch bản:

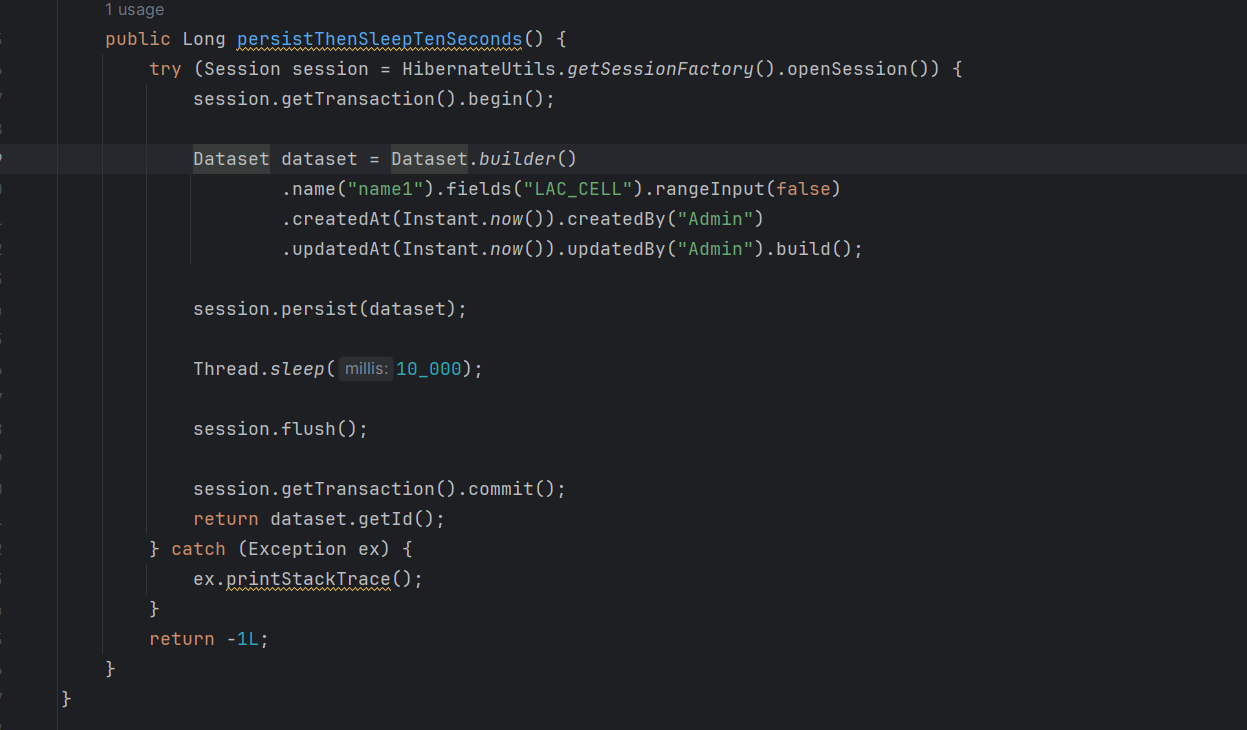
* Load ra tất cả entity từ trong db (Sử dụng createQuery)
* Load 1 entity bất kỳ theo id bằng phương thức load
* Đếm số lượng entity sử dụng .size()
* Đếm số lượng entity sử dụng hàm COUNT(id)
* Thực hiện load lại chính entity vừa load bên trên nhưng sử dụng createQuery

Kết quả:

* Chương trình thực hiện truy vấn tới db khi lấy ra toàn bộ entity
* Chương trình không thực hiện truy vấn tới db khi lấy ra entity theo id
* Chương trình không thực hiện truy vấn db khi sử dụng .size()
* Chương trình thực hiện truy vấn tới db khi sử dụng COUNT(id)
* Chương trình thực hiện truy vấn tới db khi dùng createQuery để lấy ra entity

Nhận xét:

* Khi load nhiều (tất cả) entity từ db, thì chúng đều được lưu vào first level cache, việc load lại entity nào trong số đó đều không dẫn đến truy vấn tới db
* Hibernate sẽ không load từ trong cache đối với sql query trực tiếp
  + 1. Life cycle của entity



Kịch bản:

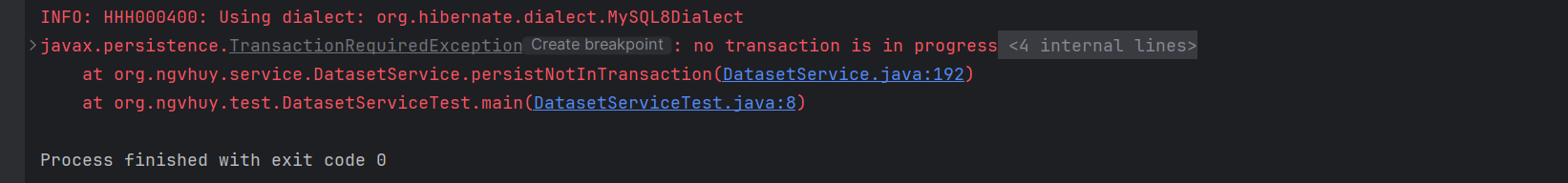
* Tạo 1 entity mới
* Persist nó
* Thread sleep trong 10s
* Kiểm tra thay đổi trong db
* Hết 10s thì sesion.flush()
* Kiểm tra lại thay đổi trong db

Kết quả:

* Khi persist xong, trong khoảng thời gian thread sleep thì dữ liệu trong db chưa được cập nhật
* Khi sesion.flush() thì dữ liệu trong db được cập nhật

Nhận xét:

* Khi persist, thông tin của entity chưa được cập nhật vào db ngay, mà phải flush (session.commit, session.close). Ví dụ này chứng minh cho cơ chế quản lý life cycle của entity hibernate
  + 1. Thực hiện save, update, get mà không có transaction:

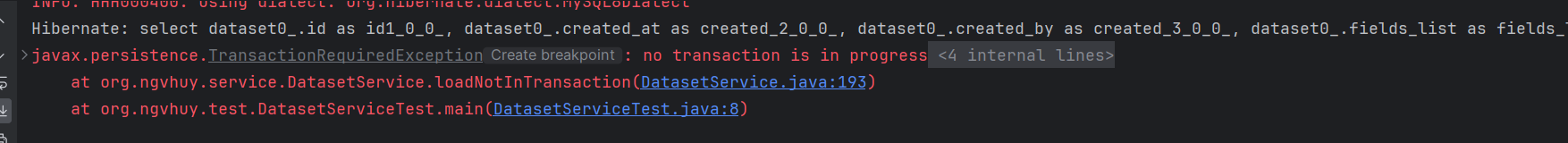


Kịch bản 1:

* Save một entity mới mà không đặt trong một transaction

Kết quả:

* Chương trình bị lỗi (no transaction is in progress)

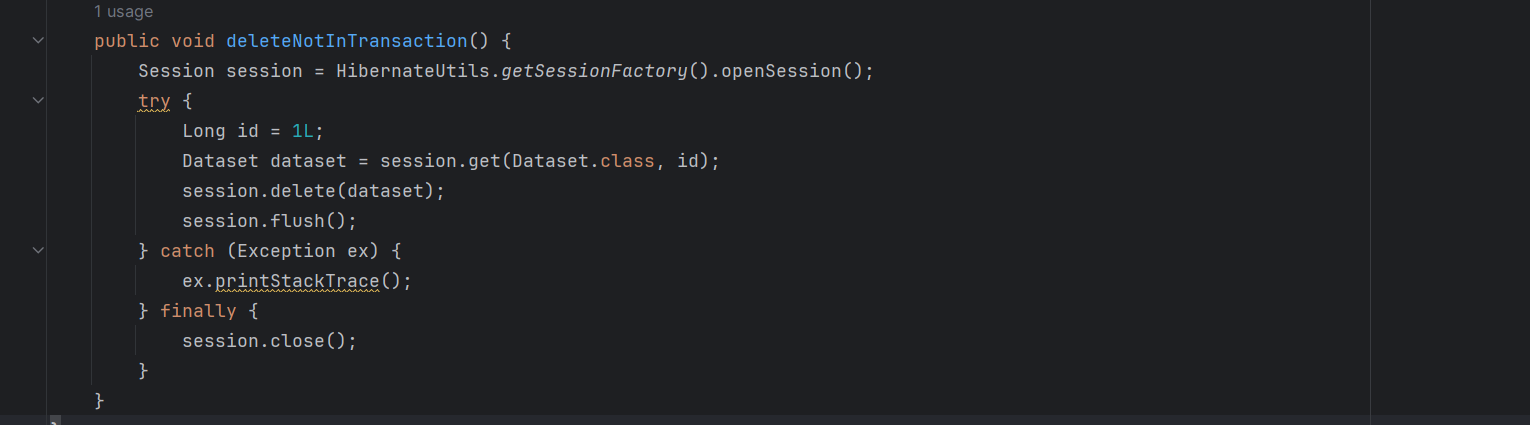
 

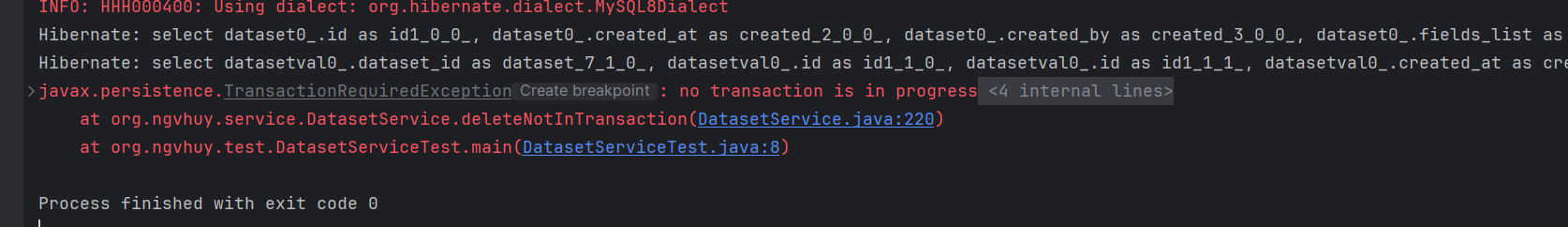
Kịch bản 2:

* Update một entity mà không đặt trong một transaction

Kết quả:

* Chương trình bị lỗi (no transaction is in progress)



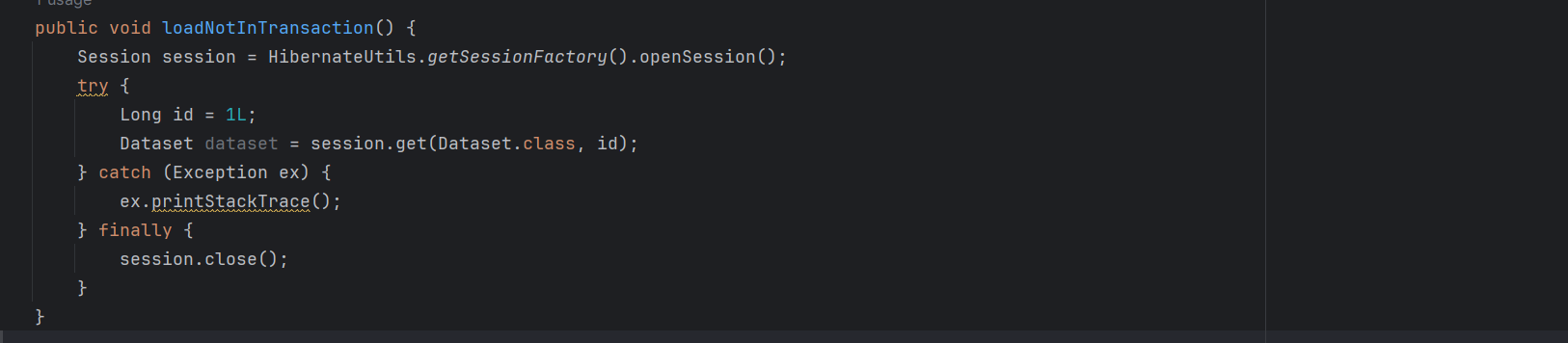
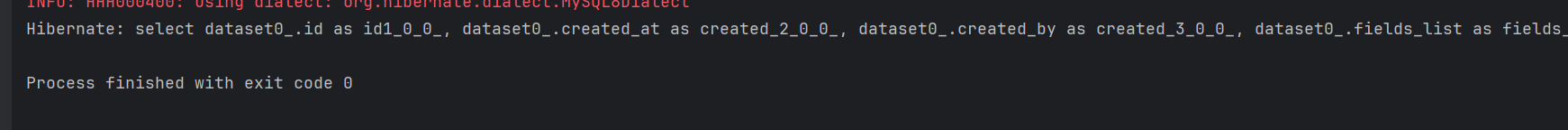


Kịch bản 3:

* Delete một entity không trong transaction

Kết quả:

* Chương trình bị lỗi (no transaction is in progress)

Kịch bản 4:

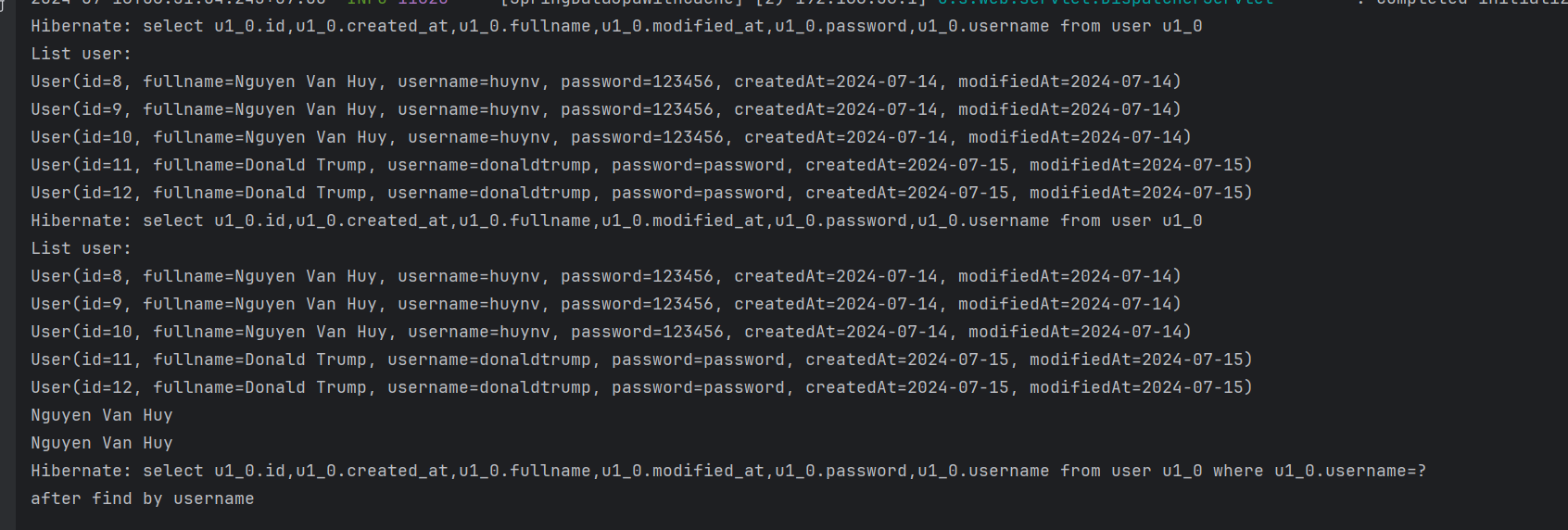
* Load một entity mà không đặt trong một transaction

Kết quả:

* Chương trình không bị lỗi

Nhận xét:

* Khi cập nhật dữ liệu trong db sử dụng session.flush() thì phải được đặt trong một tracsaction
  + 1. Spring data spa – findAll()



Kịch bản:

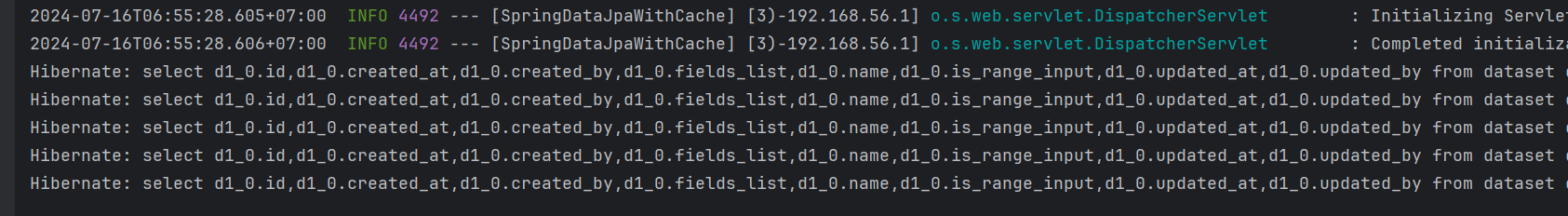
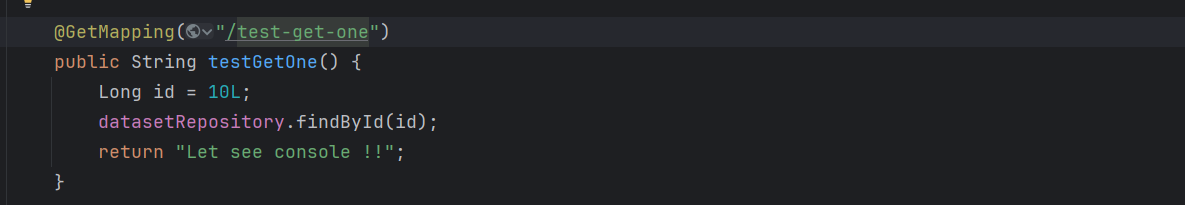
* Sử dụng repository, findAll entity lần 1
* Sử dụng repository, findAll entity lần 2
* Sử dụng repository, findById lần 1
* Sử dụng repository, findById lần 2
* Sử dụng repository, findByUsername

Kết quả:

* Chương trình thực hiện truy vấn tới db để thực hiện findAll lần 1
* Chương trình thực hiện truy vấn tới db để thực hiện findAll lần 2
* Chương trình không thực hiện truy vấn để findById lần 1
* Chương trình không thực hiện truy vấn để findById lần 2
* Chương trình thực hiện truy vấn để findByUsername

Nhận xét:

* Khi thực hiện findAll() thì tất cả entity sẽ được lưu vào first level cache, và sau đó việc findById sẽ được tìm trong cache trước tiên.
* Tuy nhiên việc findAll() lại thì lại truy vấn tới db
* Đối với các phương thức findBy khác (không phải tìm kiếm theo khóa chính) thì sẽ không tìm trong first level cache
  + 1. Spring data jpa – call api nhiều lần:



Kịch bản:

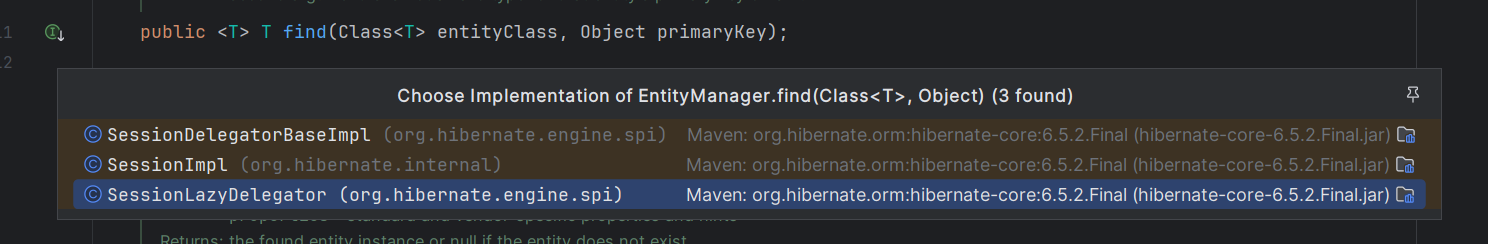
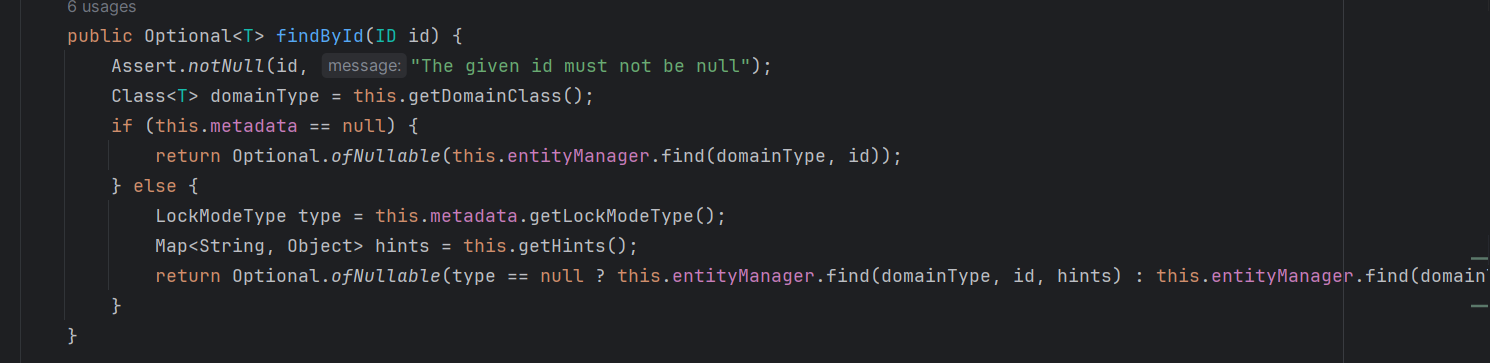
* Sử dụng postman để call api tới phương thức findById()
* Mục đích để kiểm tra hoạt động của first level cache với call api

Kết quả:

* Với việc thực hiện call api 5 lần, hibernate show sql 5 lần

Nhận xét:

* Mỗi lần call api là một session riêng, và chúng ko sử dụng chung first level cache với nhau
  + 1. Mối quan hệ giữa hibernate, jpa, spring data jpa:



Kịch bản:

* Thực hiện xem code triển khai của các phương thức của thư viện Spring Data Jpa

Kết quả:

* Các phương thức của Spring Data Jpa gọi đến các phương thức của entityManager, các phương thức này đều được triển khai bởi Hibernate, việc thao tác tới db vẫn được thông qua session

Nhận xét:

* Jpa là đặc tả - nghĩa là định nghĩa chung, là interface, được các công nghệ triển khai trong đó có hibernate
* Hibernate là công nghệ orm – nghĩa là có code, thuật toán bên trong
* Spring data jpa: là bộ thư viện xây dựng trên jpa, sử dụng hibernate để triển khai
* Các hàm của spring data jpa gọi tới EntityManager (1 phần của kiến trúc JPA), EntityManager lại gọi tới Session (1 phần của kiến trúc Hibernate)
  + 1. Transactional trong Spring Data Jpa

Nhận xét:

* Đối với các phương thức dẫn tới thay đổi dữ liệu trong db: save, delete,… Spring Data Jpa luôn đặt @Transactional lên trên hàm đó
* Đối với các phương thức read-only: get, find, exist,… thì không có @Transactional
* Tuy e chưa hiểu rõ bản chất cách hoạt động của @Transactional nhưng việc sử dụng @Transactional đúng với cách hibernate sử dụng transaction trong việc thao tác với dữ liệu.