\* Hibernate framework là một framework cho persistence layer : chịu trách nhiệm mở kết nối, truy xuất và lưu trữ data vào CSDL.

A. Kiến trúc.

1. Persintance object

-> Là các POJO map với bảng trong CSDL quan hệ.

2. Configuration

- Là 1 tập tin chứa thông tin cấu hình và thuộc tính yêu cầu của hibernate.

- Được khởi tạo đầu tiên và chỉ khởi tạo 1 lần duy nhất khi chạy ứng dụng.

- Cung cấp 2 thành phần chính :

+ Database Connection : được xử lý thông qua một hoặc nhiều tệp cấu hình được hibernate hỗ trợ.

+ Class Mapping Setup : tạo ra các kết nối giữa các lớp java và bảng trong RDBMS.

3. Session Factory

- Là interface giúp tạo ra các session kết nối đến database bằng cách đọc các cấu hình trong Configuration.

- Là đối tượng nặng nên sẽ được tạo ra trong lúc khởi động ứng dụng và lưu giữ để sử dụng sau này.

- Là một đối tượng Thread-safe và được sử dụng bởi tất cả các luồng trong chương trình.

- 1 database tương đương với 1 session factory -> nếu chương trình sử dụng 2 loại database thì cần khởi tạo 2 session factory.

4. Hibernate Session

- Đối tượng session được tạo ra bởi session factory và sẽ tạo một kết nối đến CSDL.

- Đối tượng session là nhẹ và được thiết kế để tạo ra instance mỗi khi tương tác với CSDL -> Dữ liệu được lưu và truy xuất thông qua một session instance.

- Session không phải là đối tượng thread-safe -> không nên mở session trong thời gian dài và chúng cần tạo ra và đóng lại khi cần thiết.

5. Transaction

- Một transaction đại diện cho 1 đơn vị làm việc với CSDL.

- Các transaction trong hibernate được quản lý bởi transaction manager, jdbc transaction, jta transaction.

- Transaction đảm bảo tính toàn vẹn của session với CSDL -> nếu có lỗi xảy ra trong transaction thì tất cả các tác vụ đều thất bại.

6. Query

-> Các đối tượng query sử dụng chuỗi truy vấn native SQL hoặc HQL để tương tác với CSDL.

7. Criteria

-> Đối tượng criteria được sử dụng để tạo và thực hiện các tiêu chí truy vấn để lấy các đối tượng từ CSDL.

B. Hibernate Cache

1. Phân loại

a. First level cache : cache ở cấp độ session

- Được enable mặc định trong hibernate -> không thể disable.

- Phạm vi của bộ nhớ first level cache là ở session

-> Khi session bị đóng lại, bộ nhớ cache của session đó sẽ bị mất vĩnh viễn.

-> Các đối tượng session không thể truy cập bộ nhớ cache của nhau, của ai người nấy dùng.

- Khi truy vấn 1 entity từ CSDL lần đầu tiên, nó sẽ được lưu trữ vào first level cache của session -> nếu truy vấn lại cùng 1 entity và cùng session thì dữ liệu sẽ được lấy từ first level cache thay vì thực hiện lại câu truy vấn.

- Function :

* Evict(entity) : xóa entity trong first level cache -> entity sẽ được lấy từ CSDL ở lần truy vấn tiếp theo.
* Clear() : xóa toàn bộ dữ liệu trong first level cache.
* Contains() : kiểm tra xem entity đã được lưu ở trong first level chưa.

b. Second level cache : cache ở cấp độ session factory

- Là một bộ nhớ cache tùy chọn, defaul là disable.

- Hibernate cung cấp 1 interface CacheProvider -> bên thứ 3 cần implements để cung cấp hibernate với 1 xử lý để cài đặt bộ nhớ cache.

- Setup L2 cache được thực hiện trong 2 bước :

+ Define cache concurency strategy nào được sử dụng.

+ Sử dụng cache provider để cấu hình các attribute cho cache expiration và cache vật lý.

\* Cache Concurrency Strategy

- Là bộ điều chỉnh có trách nhiệm lưu trữ các mục dữ liệu trong bộ nhớ cache và lấy chúng từ bộ nhớ cache.

- Khi kích hoạt L2 cache -> cần phải quyết định chiến lược truy cập đồng thời vào bộ nhớ cache để sử dụng :

* READ\_ONLY :
  + Sử dụng cho các đối tượng đọc nhưng không bao giờ cập nhật.
  + Là chiến lược có hiệu suất tốt nhất vì không có quá trình tải để kiểm tra xem đối tượng có được cập nhật trong CSDL hay không.
* NONSTRICT\_READ\_WRITE :
  + Sử dụng cho các đối tượng hầu như không thay đổi.
  + Trong trường hợp đối tượng không nhất quán giữa cache và CSDL thì cũng không gây ra vấn đề gì cho ứng dụng.
* READ\_WRITE :
  + Sử dụng cho các đối tượng liên tục được cập nhật bằng hibernate -> nếu sử dụng các phần mềm khác để update dữ liệu thì hibernate sẽ không thể biết được cache bị cũ.
  + Chiến lược này đảm bảo tính nhất quán dữ liệu giữa cache và CSDL thông qua soft lock.
  + Khi 1 đối tượng được cache bị update. Một soft lock được lưu lại trong cache cho entity và nó sẽ được giải phóng khi transaction được commit.
* TRANSACTIONAL :
  + Cung cấp bộ đệm transaction đầy đủ :
    - 1 thay đổi trong entity được lưu trữ trong cache được đảm bảo khôi phục trong cả CSDL và bộ đệm trong cùng 1 transaction.

\* Cache Provider :

- Phân loại :

* EH Cache :
  + Có thể cache trong ram hoặc trên ổ cứng và clustered caching.
  + Hỗ trợ bộ nhớ cache kết quả truy vấn hibernate tùy chọn.
* OS Cache :
  + Hỗ trợ cache vào ram hoặc ổ cứng trong 1 JVM duy nhất.
  + Có một tập hợp đầy đủ các chính sách hết hạn và hỗ trợ bộ nhớ truy vấn.
* Swarm Cache :
  + Là 1 bộ nhớ cache cluster dựa trên Jgroups.
  + Không hỗ trợ bộ nhớ cache truy vấn hibernate.
* JBoss Cache :
  + Là 1 bộ nhớ cache được sao chép hoàn toàn hợp lệ được transaction dựa trên thư viện đa nhóm Jgroups.
  + Hỗ trợ nhân bản hoặc hủy bỏ hiệu lực.
  + Giao tiếp đồng bộ hoặc bất đồng bộ.
  + Optimistic, pessimistic locking.
  + Bộ nhớ cache truy vấn hibernate được hỗ trợ tạm thời.

\* Mỗi loại Cache Provider tương thích với những Cache Concurrency Strategy khác nhay :



\* Phương thức hoạt động L2 Cache khi session load 1 entity :

* Tìm trong L1 cache :
  + Nếu tìm thấy entity trong cache -> trả về kết quả.
  + Nếu không thực hiện bước tiếp theo.
* Tìm trong L2 cache :
  + Nếu tìm thấy entity trong L2 cache :
    - Lưu entity sang L1 cache để các lần load sau sẽ return từ L1 cache.
    - Trả về kết quả.
  + Nếu không tìm thấy trong L2 cache :
    - Thực hiện truy vấn vào CSDL.
    - Lưu trữ entity tại L1 và L2 cache.
    - Trả về kết quả.
* Nếu việc sửa đổi entity được thực hiện qua API hibernate session thì việc cập nhật cache ở L2 sẽ phụ thuộc vào cache provider nào được triển khai.

\* Kích hoạt L2 cache :

- hibernate.cache.use\_second\_level\_cache : có sử dụng L2 cache hay không.

- hibernate.cache.region.factory\_class : chỉ định tên lớp Region Factory của Cache Provider.

- hibernate.cache.use\_query\_cache : sử dụng query cache hay không.