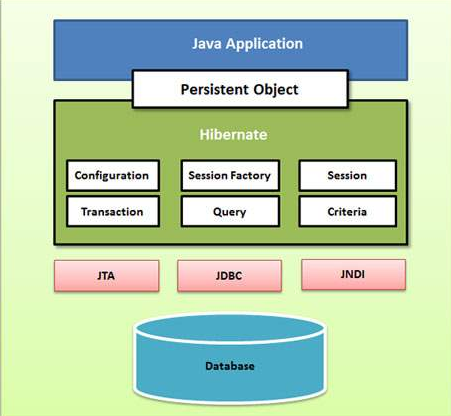
**Hibernate**

* Là một giải pháp ORM, nằm giữa Java objects và database server giúp đơn giản hóa sự phát triển của ứng dụng java để tương tác vs cơ sở dữ liệu

1. Lợi ích của Hibernate

* Mã nguồn mở nhẹ
* Hiệu suất nhanh: có hai bộ nhớ cache, gồm bộ nhớ cache cấp 1 và bộ nhớ cache cấp 2, cấp 1 đc bật bằng lệnh mặc định
* Truy vấn CSDL độc lập: Nó tạo ra các truy vấn cơ sở dữ liệu độc lập vì thế nên không cần phải viết các truy vấn cơ sở dữ liệu cụ thể, nếu có cơ sở dữ liệu bị thay đổi, thì k cần thay đổi truy vấn
* Tạo bảng tự động: nó cung cấp các phương tiện để tạo bảng tự động
* Đơn giản các lệnh join phức tạp: có thể lấy dữ liệu nhiều bảng một cách dễ dàng

1. Kiến trúc của Hibernate



* Java Application
* Hibernate framework
* Database
* Backhand api

1. Các thành phần trong kiến trúc Hibernate

* Configuration: là đối tượng đầu tiên đc tạo, chỉ tạo một lần khi khởi tạo ứng dụng, có hai thành phần chính:

+ database connection: Thao tác này xử lý thông qua môt hoặc nhiều tệp cấu hình ( hibernate.properties và hibernate.cfg.xml)

+ class Mapping setup: tạo ra kết nối giữa các lớp java và các bảng cơ sở dữ liệu

* SessionFatory: sd để có kết nối vs csdl, là một đối tượng an toàn và được sử dụng bởi tất cả các luồng của ứng dụng, cần một đối tượng cho mỗi csdl
* Session: đc sử dụng để có đc một kết nối vật lý vs một csdl, nó không phải luồng an toàn
* Transaction: đại diện cho một đơn vị làm việc vs CSDL, được xử lý bởi trình quản lý transaction và transaction
* Query: sử dụng chuỗi truy vấn SQL hoặc HQL để lấy dữ liệu từ cơ sở dữ liệu và tạo các đối tượng
* Criteria: để tạo và thực hiện các truy vấn các tiêu chí hướng đối tượng để lấy các đối tượng

1. Cấu hình

* Hibernate.dialect : làm cho hibernate tạo ra SQL thích hợp cho csdl đã chọn
* Hibernate.connection.driver\_class: JDBC driver class
* Hibernate.connection.url: JDBC URL của csdl
* Hibernate.connection.username: username của csdl
* Hibernate.connection.password : password
* Hibernate.connection.pool\_size: giới hạn số kết nối đang chờ
* Hibernate.connection.autocommit: cho phép chế độ autocommit để được sử dụng cho JDBC connection

1. Mapping file: dung để ánh xạ một hoặc nhiều lớp trong java với các bảng trong database, có thể tạo bằng tay hoặc dung tool, XDoclet, Middlegen, AndroMDA

* Các phần tử của file mapping:

+ hibernate-mapping: chứa tất cả các phần tử

+ <class> để ánh xạ cụ thể từ các lớp java sang bảng csdl

+ <meta> dung để tạo ra mô tả các lớp

+ <id> ánh xạ thuộc tính id duy nhất tới khóa chính

+ <generator> đc sử dụng để tạo tự động giá trị cho khóa chính

+< property> ánh xạ một thuộc tính thành một cột của csdl, …

1. Mapping Type

* Là kiểu dữ liệu ánh xạ Hibernate, có thể dịch từ kiểu dữ liệu java qua SQL và ngược lại

+ Các kiểu dữ liệu nguyên thủy

+ Các kiểu date and time

+ Các kiểu Binary và đối tượng lớn

1. Lớp persistent: là các lớp có các đối tượng hoặc các thể hiện được lưu trữ trong bảng csdl, các lớp này tuân theo một số quy tắc đơn giản là POJO:

* Cần có construsctor mặc định
* Phải có ID để dễ xác định, ánh xạ tới một khóa chính của một bảng csdl
* Tất cả thuộc tính phải có setter và getter
* Là non-final hoặc đc implements từ interface có các phương thức public
* Các lớp k đc extend hoặc implement một vài lớp

1. Session trong Hibernate

* Nó được sử dụng để có được một kết nối vật lý với một cơ sở dữ liệu, nó được tạo ra và đóng lại sau khi sử dụng
* Các chức năng chính là create, update, read, delete
* Tồn tại ở một trong ba trạng thái tại một thời điểm nhất định:

+ transient: một thể hiện của persistent không liên qua tới session, không có đại diện trong csdl, không có giá trọ định danh

+ persistent: tạo bằng cách liên kết nó với một session, có một đại diện trong csdl, một giá trị định danh và đc liên kết vs một session

+ detached: đi đóng sission, persistent trở thành detached

1. O/R mappings:

Chú ý:

* Collections Mappings

+ java.util.Set : được ánh xạ với một phần tử <set> và được khởi tạo với java.util.HashSet

+ java.util.SortedSet: được ánh xạ với một phần tử <set> và được khởi tạo với java.util.TreeSet. Thuộc tính sort để định nghĩa một tập đc sắp xếp hoặc không

+ java.util.List được ánh xạ với một phần tử <list> và được khởi tạo với java.util.ArrayList

+ java.util.Map: được ánh xạ với một phần tử <map> và được khởi tạo với java.util.HashMap

+ java.util.SortedMap: được ánh xạ với một phần tử <map> và được khởi tạo với java.util.TreeMap. Thuộc tính sort để định nghĩa một tập đc sắp xếp hoặc không

Mảng được hỗ trợ với <primitive-array> cho các kiểu giá trị Java nguyên thủy và <array> cho các kiểu dữ liệu khác

* Association mappings

Có 4 kiểu liên kết: many – to – One, one –to – one, one – to many, many – to – many.

+ One to one (một người chỉ đứng một vị trí trong lớp, một vị trí chỉ chứ một người) cách tốt nhất để map trong quan hệ one to one là sử dụng @MapsID. Theo cách này thì bạn không cần tới mối quan hệ hai chiều mà có thể lấy đc cái con từ sử dụng định danh của cái cha. Cột id sẽ bao gồm cả Pk và fk

+ OnetoMany: ( ví dụ một lớp có nhiều sv, một sv chỉ thuộc một lớp) map các lớp con thành collection của lớp cha, có thể dùng list hoặc dùng set, có hai cách là unidirectional và bidirectional.

* Unidirectional @OneToMany

Sẽ có ba bảng, sử dụng hai khóa ngoài

* Bidirectional @OneToMany

Cách tốt nhấp để map one to many là dựa vào many to one, cha sẽ dùng one to many, con sẽ dùng many to one,

+ ManyToMany: ( một sv nhiều lp….) bảng giữa chứa hai khóa ngoài

* Component mappings

1. Annotations

* Dùng các Annotations để bổ sung thay thế mapping XML ( nếu có cả hai thì XML sẽ ghi đè lên annotation)

+ lợi tích hơn sơ với dùng XML : viết ngắn gọn hơn, tích hợp trong code; cos th

* @Entity Annotation: chú thích Entity vào lớp để đánh dấu lớp này như một Entity Bean, do đó nó phải có một constructor không có đối số mà có kiểu public
* @Table Annotation: chỉ định các chi tiết của bảng sẽ được sử dụng để lưu trữ thực thể trong csdl
* @Id và @GeneratedValue annotation: id sẽ tự động xác định tạo primary key, nhưng có thể ghi đè bằng cách áp dụng chú thích @GeneratedValue có hai tham số strategy và generator
* @Column Annotation : dùng để chỉ định chi tiết các cột mà trường hoặc thuốc tính sẽ được ánh xạ

+ name: chỉ định tên cột nào tỏng db map vs tên trường đc chú thích

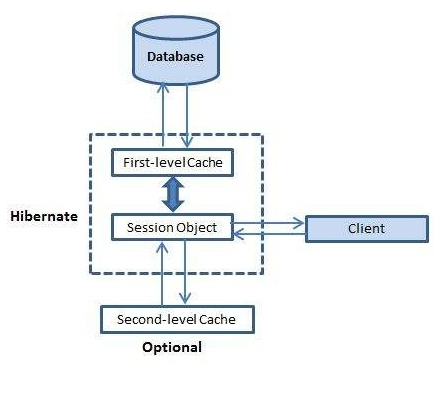
+ length: kích thước của cột được sử dụng để ánh xạ một giá trị đặt biệt cho một giá trị String

+ Nullable cho phép cột được đánh dấu không Null khi schema được tạo ra

+ Unique cho phép cột được đánh dấu chỉ chứa các giá trị duy nhất

1. Bộ nhớ cache

* Có ba cấp là bộ nhớ cấp 1, bộ nhớ cấp 2, bộ nhớ cấp truy vấn



1. First-level cache

Là một session cache và nó là một phần bắt buộc, phần mà tất cả các request đều phải truyền qua. Session object này giữ một object trong các session trước khi commit lên DB. Nếu có nhiều hoạt động update được gọi cho 1 object, thì hibernate sẽ tạo ra một khoảng trễ đủ dài để giảm số lượng câu SQL update đưa ra. Nếu session này bị đóng thì tất cả object trong session bị mất và nó sẽ được persist hoặc update vào DB

1. Seccon-level cache (???)

Là một bộ nhớ cache tùy chọn và bộ nhớ thứ nhất sẽ luôn đc thực hiện trước khi đặt một object vào bộ nhớ thứ hai. Nó được cấu hình trên cơ sở mỗi lớp và mỗi collection và chịu trách nhiệm chính trong việc lưu trữ các đối tượng trong các session

+ Thành phần quan trọng của bộ nhớ cấp 2 là concurrency strategies. Một concurrency strategies là một thành phần phụ trách việc lưu trữ các item cả dữ liệu vào cache đang được enable. Các loại concurrency strategy được sử dụng sau:

* Transactional: sử dụng strategy này hầu hết cho việc đọc dữ liệu để tránh việc mất dữ liệu trong các transactions xảy ra đồng thời. Strategy này ít dung cho update
* Read – wite: Strategy khá giống transactional strategy, nó cũng được sử dụng hầu hết cho việc đọc data
* Nonstrict – read – write: Strategy này không đảm bảo đồng chất giữa cache và database. Sử dụng strategy này nếu data ít khi bị thay đổi và nó không còn đc sử dụng
* Read- only: Strategy này thích hợp cho dữ liệu mà không bao giờ bị thay đổi, chỉ sử dụng cho việc reference dữ liệu

Cache provider :

* EHCache:
* OSCache:
* warmCache:
* JBoss Cache:

1. Query level cache (???)

Thực hiện bộ nhớ cache cho các kết quả truy vấn kết hợp chặt chẽ với bộ nhớ cache cấp hai. Đây là tính năng chọn và yêu cầu thêm hai vùng bộ nhớ vật lý giữ kết quả truy vấn và các timestamps khi một table được update lần cuối

1. Ngôn ngữ truy vấn HQL: nó thực hiện trên các đối tượng persistent và thuộc tính của chúng  
   ex:

|  |
| --- |
| String hql = "FROM Employee";  Query query = session.createQuery(hql);  List results = query.list(); |

+ FROM : load toàn bộ đối tượng trong bộ nhớ

+ AS: gán bí danh cho lớp

+ SELECT:

+ WHERE: thu hẹp nơi lưu trữ

+ ORDER BY: sắp xếp kết quả truy vấn

+ GROUP BY: nhóm các kết quả theo các thuộc tính

+UPDATE

+ DELETE

+ INSERT

1. Native SQL

Nó thực hiện trên truy vấn csdl, truyền vào chuỗi chứa truy vấn sql và thực trực tiếp vs entity

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | String sql = "SELECT first\_name, salary FROM EMPLOYEE"; |  |  | | --- | | SQLQuery query = session.createSQLQuery(sql); |  |  | | --- | | query.setResultTransformer(Criteria.ALIAS\_TO\_ENTITY\_MAP); |  |  | | --- | | List results = query.list(); | |