CMC UNIVERSITY

Aspire to Inspire the Digital World

CHƯƠNG 5: KẾT NỐI CSDL

Ngô Việt Anh

Khoa CNTT&TT, Trường Đại học CMC

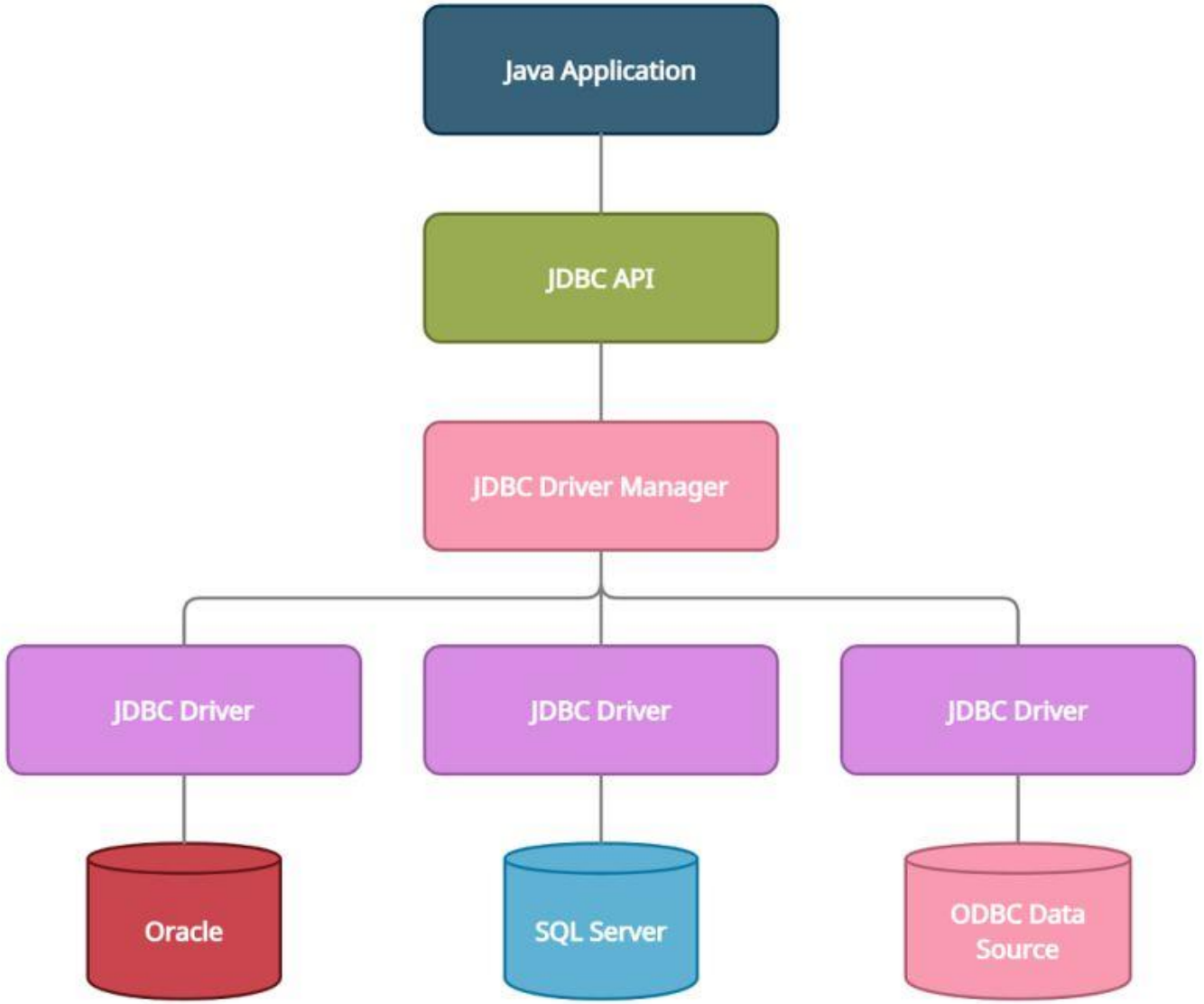
Hà Nội, tháng 8 năm 2024

Giới thiệu

- Trong Java, có nhiều phương pháp để kết nối với cơ sở dữ liệu (CSDL). Các phương pháp này thường sử dụng JDBC (Java Database Connectivity), là API chính để giao tiếp giữa Java và các hệ quản trị cơ sở dữ liệu (DBMS). Dưới đây là một số phương pháp phổ biến để kết nối với CSDL trong Java:
  - Sử dụng JDBC (Java Database Connectivity)
  - Sử dụng Connection Pooling với JDBC
  - Sử dụng ORM Framework (Hibernate, JPA)
  - Sử dụng Spring Data JPA

JDBC (Java Database Connectivity)

- JDBC là cách trực tiếp và phổ biến nhất để kết nối với cơ sở dữ liệu trong Java. Nó cung cấp các lớp và phương thức cần thiết để thực hiện các thao tác cơ bản như kết nối, thực thi truy vấn SQL, xử lý kết quả và cập nhật dữ liệu.
- Các bước cơ bản để kết nối với cơ sở dữ liệu bằng JDBC:
  - Tải driver JDBC: Mỗi hệ quản trị cơ sở dữ liệu (MySQL, Oracle, PostgreSQL, SQL Server, v.v.) đều có driver riêng.
  - Tạo kết nối với cơ sở dữ liệu: Sử dụng lớp DriverManager để tạo kết nối.
  - Tạo câu lệnh SQL: Sử dụng các đối tượng như Statement hoặc PreparedStatement.
  - Thực thi truy vấn và xử lý kết quả: Sử dụng đối tượng ResultSet để xử lý kết quả trả về.
  - Đóng kết nối: Giải phóng tài nguyên khi không còn cần thiết.


Java Application
JDBC API
JDBC Driver Manager
JDBC Driver
JDBC Driver
JDBC Driver
Oracle
SQL Server
ODBC Data Source

- Cung cấp các lớp, phương thức và giao diện khác nhau giúp giao tiếp dễ dàng với cơ sở dữ liệu. Nó cũng cung cấp hai gói chứa nền tàng Java SE và Java EE để thể hiện các khả năng WORA (write one run anywhere). Ngoài ra còn có một tiêu chuẩn trong API JDBC để kết nối cơ sở dữ liệu với ứng dụng khách.
- Một số lớp và giao diện quan trọng được định nghĩa trong API JDBC như sau:
  - DriverManager
  - Driver
  - Connection
  - Statement
  - PreparedStatement
  - CallableStatement
  - ResultSet
  - SQL data

Ví dụ JDBC

- Kết nối CSDL MySQL, liệt kê toàn bộ danh danh sinh viên trong bảng Student

Connection Pooling với JDBC

- Connection Pooling là kỹ thuật sử dụng một tập hợp các kết nối đã được mở sẵn và tái sử dụng chúng, thay vì tạo mới mỗi lần cần kết nối với cơ sở dữ liệu. Điều này giúp tăng hiệu suất và giảm chi phí khi quản lý kết nối.
- Các thư viện như HikariCP, Apache DBCP, và C3P0 cung cấp giải pháp Connection Pooling phổ biến.

Ví dụ sử dụng Connection Pooling với JDBC

- ORM (Object-Relational Mapping) là một kỹ thuật chuyển đổi giữa các đối tượng trong lập trình hướng đối tượng (Object-Oriented Programming) và bảng trong cơ sở dữ liệu. Các framework ORM phổ biến trong Java như Hibernate và Java Persistence API (JPA) giúp đơn giản hóa quá trình tương tác với CSDL mà không cần viết nhiều mã SQL.

Ví dụ sử dụng ORM sử dụng Hibernate

- Trong ví dụ này, chúng ta sẽ xây dựng một ứng dụng Java sử dụng JPA để quản lý thông tin về Sách trong một thư viện. Ứng dụng sẽ sử dụng JPA để tương tác với cơ sở dữ liệu mà không cần viết trực tiếp các câu truy vấn SQL, giúp đơn giản hóa quá trình xử lý dữ liệu.
- Các chức năng chính của ứng dụng:
  - Thêm sách mới vào thư viện.
  - Lấy danh sách tất cả các sách.
  - Tìm kiếm sách theo ID.
  - Cập nhật thông tin sách.
  - Xóa sách theo ID.

Cấu trúc thư mục

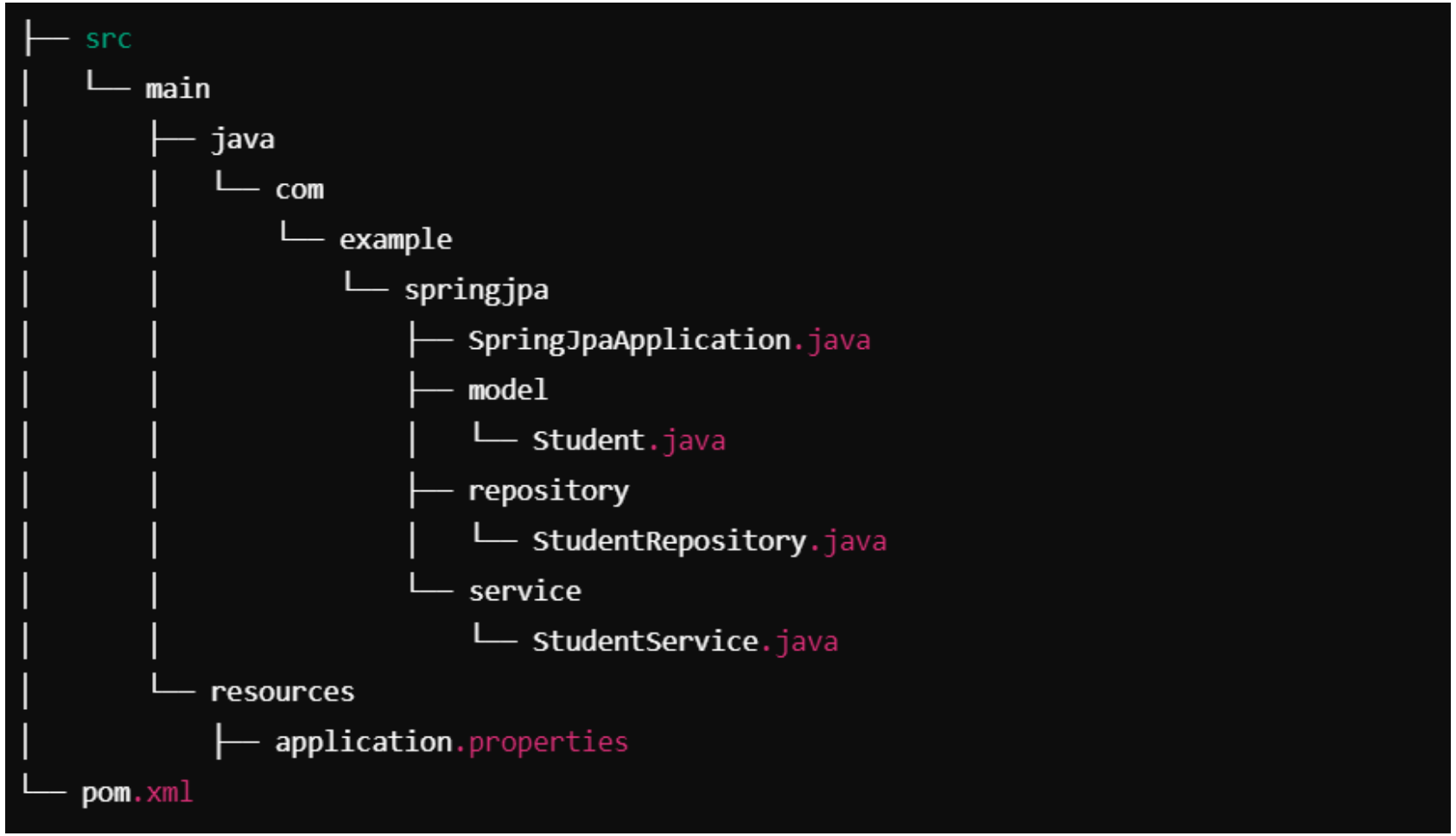

```
jpa-example/
├── src
│   └── main
│       ├── java
│       │   └── com
│       │       └── example
│       │           └── jpaexample
│       │               ├── JpaExampleApplication.java
│       │               ├── model
│       │               │   └── Book.java
│       │               ├── repository
│       │               │   └── BookRepository.java
│       │               └── service
│       │                   └── BookService.java
│       └── resources
│           └── META-INF
│               └── persistence.xml
└── pom.xml
```

- Spring Data JPA là một phần của Spring Framework, nó cung cấp các lớp và phương thức giúp thao tác với cơ sở dữ liệu dễ dàng hơn bằng cách sử dụng JPA. Bạn có thể sử dụng các phương thức của CRUDRepository hoặc JPARepository mà không cần viết nhiều mã.

Ví dụ sử dụng Spring JPA

- Trong ví dụ này, chúng ta sẽ xây dựng một ứng dụng Spring Boot sử dụng Spring Data JPA để quản lý dữ liệu của sinh viên trong một cơ sở dữ liệu MySQL. Ứng dụng sẽ có các chức năng:
  - Thêm sinh viên.
  - Lấy danh sách tất cả sinh viên.
  - Lấy thông tin sinh viên theo ID.
  - Cập nhật thông tin sinh viên.
  - Xóa sinh viên theo ID.

Cấu trúc thư mục




CMC UNIVERSITY
Aspire to Inspire the Digital World
ORM & JPA


Code
ORM
DATABASE

- **ORM** (Object-Relational Mapping) là một kỹ thuật lập trình cho phép ánh xạ (mapping) các đối tượng trong lập trình hướng đối tượng (OOP) với các bảng trong cơ sở dữ liệu quan hệ. Thông qua ORM, lập trình viên có thể làm việc với các đối tượng trong code mà không cần phải viết các câu truy vấn SQL phức tạp.

Ví dụ

- Ví dụ về ORM:
  - Trong lập trình hướng đối tượng, chúng ta có một lớp Student với các thuộc tính như id, name, và age.
  - Trong cơ sở dữ liệu, chúng ta có một bảng students với các cột tương tự: id, name, và age.
- ORM sẽ giúp ánh xạ lớp Student với bảng students. Khi bạn muốn lưu trữ một đối tượng Student vào cơ sở dữ liệu, ORM sẽ tự động chuyển đổi tượng này thành một câu lệnh SQL và thực thi nó.

Ưu điểm của ORM

- Tăng hiệu quả lập trình: Lập trình viên có thể tập trung vào các đối tượng và logic của ứng dụng thay vì xử lý chi tiết việc giao tiếp với cơ sở dữ liệu.
- Tự động hóa các tác vụ CRUD: ORM tự động hóa việc tạo, đọc, cập nhật và xóa (CRUD) dữ liệu.
- Tránh lỗi: ORM giúp tránh lỗi phát sinh khi chuyển đổi giữa các kiểu dữ liệu của đối tượng Java và kiểu dữ liệu trong cơ sở dữ liệu.
- Tối ưu hóa hiệu suất: Một số công cụ ORM có cơ chế caching giúp giảm số lượng truy vấn đến cơ sở dữ liệu, từ đó tăng hiệu suất ứng dụng.

Nhược điểm của ORM

- Hiệu suất: Trong một số trường hợp, ORM có thể chậm hơn khi thực hiện các thao tác phức tạp so với việc viết SQL thủ công.
- Phức tạp cho các truy vấn đặc thù: ORM có thể không phù hợp cho các truy vấn SQL phức tạp, đặc biệt là khi bạn cần tối ưu hóa truy vấn cho một loại cơ sở dữ liệu cụ thể.

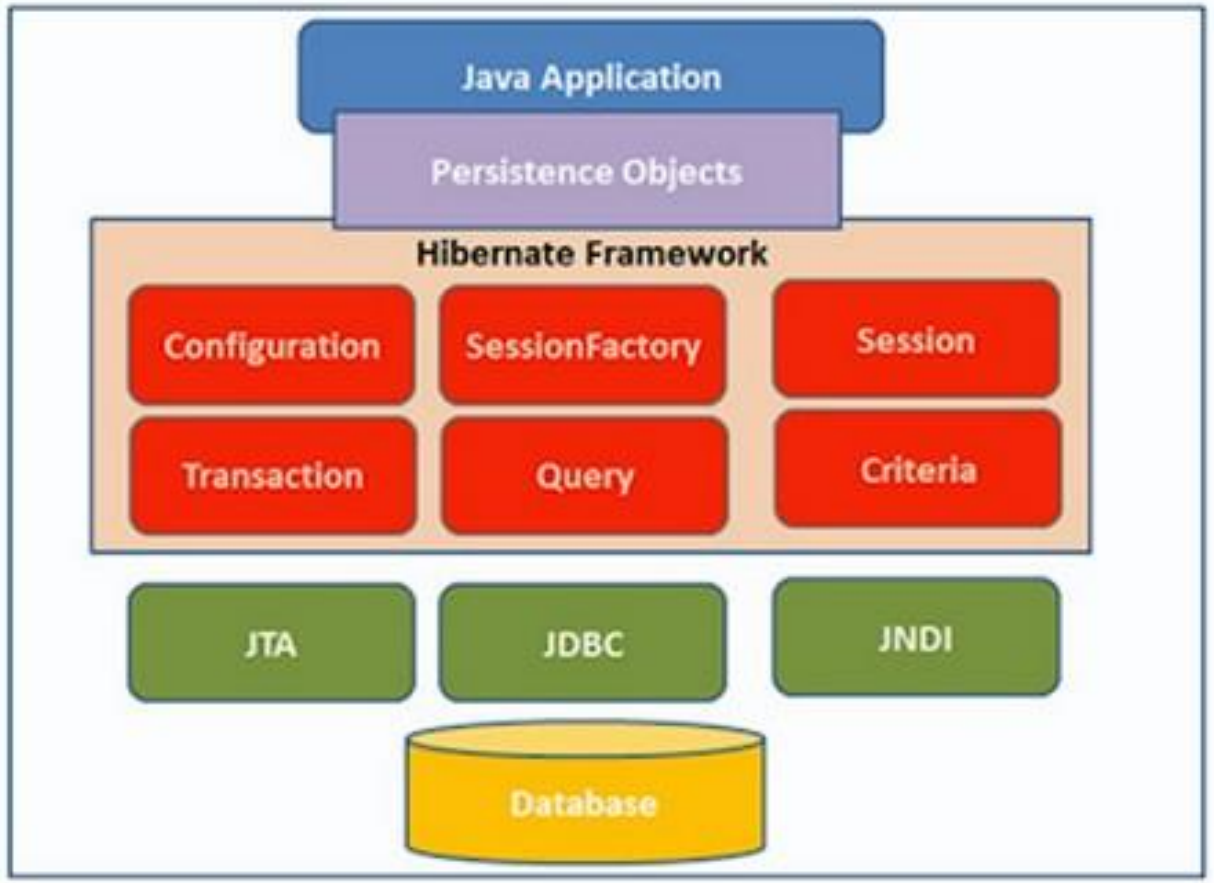
- **JPA** là một chuẩn API (Application Programming Interface) trong Java để làm việc với dữ liệu trong các hệ quản trị cơ sở dữ liệu. JPA là một phần của **Java EE (Enterprise Edition)** và nó được thiết kế để đơn giản hóa việc sử dụng ORM trong Java.
  - JPA không phải là một ORM framework thực tế, mà là một specification (đặc tả kỹ thuật). Nó định nghĩa các tiêu chuẩn mà một ORM framework cần phải tuân thủ.
  - Các ORM framework phổ biến như Hibernate, EclipseLink, và TopLink đều là các ORM framework cụ thể triển khai JPA.
  - JPA giúp lập trình viên Java dễ dàng tương tác với cơ sở dữ liệu mà không cần phải viết các câu lệnh SQL trực tiếp.

- Entity: Là một lớp Java đại diện cho một bảng trong cơ sở dữ liệu. Mỗi đối tượng của lớp này tương ứng với một bản ghi (record) trong bảng.
- EntityManager: Đối tượng này giúp quản lý các entity, cung cấp các chức năng tạo, đọc, cập nhật và xóa dữ liệu trong cơ sở dữ liệu.
- Persistence Context: Là một bối cảnh chứa các entity đã được quản lý bởi EntityManager. Các thay đổi trên entity sẽ được theo dõi và đồng bộ với cơ sở dữ liệu khi cần thiết.
- JPQL (Java Persistence Query Language): Đây là ngôn ngữ truy vấn được JPA sử dụng, nó tương tự như SQL nhưng làm việc với các đối tượng Java thay vì các bảng và cột trong cơ sở dữ liệu.

Các chú thích (Annotations) trong JPA

- JPA sử dụng nhiều annotations để định nghĩa các entity và ánh xạ các thuộc tính của chúng với các cột trong cơ sở dữ liệu. Dưới đây là một số annotations phổ biến:
  - @Entity: Xác định rằng lớp này là một thực thể (entity) và sẽ được ánh xạ với một bảng trong cơ sở dữ liệu.
  - @Table(name = "table_name"): Định nghĩa bảng cụ thể mà entity được ánh xạ đến.
  - @Id: Đánh dấu thuộc tính này là khóa chính của bảng.
  - @GeneratedValue(strategy = GenerationType.IDENTITY): Chỉ định rằng giá trị của trường này sẽ được tạo tự động (ví dụ: auto-increment trong cơ sở dữ liệu).
  - @Column(name = "column_name"): Chỉ định tên cột trong cơ sở dữ liệu mà thuộc tính của entity sẽ được ánh xạ.

Hibernate Architecture



Hibernate

- Hibernate là một Object-Relational Mapping (ORM) framework mã nguồn mở phổ biến trong Java, giúp chuyển đổi và ánh xạ dữ liệu giữa các đối tượng trong lập trình hướng đối tượng (OOP) và các bảng trong cơ sở dữ liệu quan hệ.
- Hibernate triển khai đầy đủ JPA (Java Persistence API) - một tiêu chuẩn ORM trong Java, nhưng ngoài các tính năng của JPA, Hibernate còn cung cấp nhiều tính năng bổ sung mạnh mẽ.

Các tính năng chính của Hibernate

1. Object-Relational Mapping (ORM):
   - Hibernate cung cấp khả năng ánh xạ giữa các lớp Java và các bảng trong cơ sở dữ liệu. Mỗi thuộc tính của lớp Java có thể tương ứng với một cột trong bảng.
2. Tự động tạo truy vấn SQL:
   - Với Hibernate, bạn không cần viết thủ công các truy vấn SQL. Hibernate sẽ tự động chuyển đổi các thao tác trên đối tượng thành các câu lệnh SQL tương ứng.
3. Hỗ trợ các thao tác CRUD (Create, Read, Update, Delete):
   - Hibernate cung cấp các phương thức đơn giản để thực hiện các thao tác CRUD mà không cần phải viết câu lệnh SQL.

4. Caching (Bộ nhớ đệm):Hibernate cung cấp các cơ chế caching để giảm thiểu truy vấn cơ sở dữ liệu, từ đó tăng hiệu suất ứng dụng. Hibernate hỗ trợ First Level Cache (mặc định) và Second Level Cache.
5. Lazy Loading (Tải dữ liệu theo yêu cầu):Hibernate hỗ trợ Lazy Loading, giúp chỉ tải dữ liệu khi cần thiết. Điều này giúp giảm thiểu số lượng dữ liệu không cần thiết được tải về từ cơ sở dữ liệu.
6. Inheritance Mapping (Ánh xạ kế thừa):Hibernate hỗ trợ ánh xạ các lớp kế thừa trong lập trình hướng đối tượng với các bảng trong cơ sở dữ liệu. Có nhiều chiến lược ánh xạ như Single Table, Joined Table, hoặc Table per Class.

7. Transaction Management (Quản lý giao dịch):Hibernate tích hợp sẵn việc quản lý giao dịch và đảm bảo tính nhất quán của dữ liệu khi có nhiều thao tác xảy ra đồng thời.
8. HQL (Hibernate Query Language):Hibernate cung cấp một ngôn ngữ truy vấn tương tự SQL nhưng làm việc với các đối tượng Java thay vì các bảng và cột của cơ sở dữ liệu. HQL là hướng đối tượng, cho phép truy vấn các đối tượng và thuộc tính của lớp.
9. Support for Database Independence:Hibernate có khả năng làm việc với nhiều loại cơ sở dữ liệu khác nhau (MySQL, PostgreSQL, Oracle, SQL Server, H2, HSQLDB, v.v.), và bạn có thể dễ dàng chuyển đổi giữa các cơ sở dữ liệu mà không cần phải thay đổi nhiều mã nguồn.

Persistence Object

- Persistent Object được hiểu là một kho dùng để lưu trữ dữ liệu. Nó giúp tập trung dữ liệu được tải từ cơ sở dữ liệu đến ứng dụng và ngược lại. Các đối tượng Persistence cũng là các đối tượng POJO mà Hibernate map với các hệ thống quản trị cơ sở dữ liệu quan hệ.

Configuration

- Configuration là đối tượng đầu tiên được nhìn thấy khi khởi tạo một Hibernate. Người dùng chỉ cần tạo một lần là Configuration sẽ hiển thị mãi mãi trong ứng dụng. Configuration được hiểu là một tệp cấu hình có điều kiện hoặc thuộc tính mong muốn. Configuration bao gồm hai bộ phận như sau:
  - Database Connection: Đây là bộ phận để kết nối với cơ sở dữ liệu, Configuration sử dụng một hoặc nhiều tệp cấu hình có dạng hibernate.properties và hibernate.cfg.xml.
  - Class Mapping Setup: Đây là bộ phận chịu trách nhiệm tạo kết nối giữa Hibernate và cơ sở dữ liệu (database)

Session Factory

- Sessionfactory là giao diện kết nối với cơ sở dữ liệu thông qua việc đọc Configuration. Mỗi cơ sở dữ liệu đều có yêu cầu về một Sessionfactory riêng biệt. Vì vậy, nếu bạn muốn sử dụng nhiều cơ sở dữ liệu như MySQL hay Oracle cùng một lúc, bạn nên tạo một Sessionfactory riêng cho MySQL và một Sessionfactory khác cho Oracle. Tuy nhiên, Sessionfactory có đặc điểm là năng và an toàn vậy nên Sessionfactory thường được tạo trong quá quá trình khởi động và lưu trữ lại để dùng sau khi cần.

- Vì Sessionfactory là một đối tượng nặng và khó có thể sử dụng thường xuyên. Chính vì vậy mà Sessionfactory sẽ chủ động tạo ra các Hibernate với các thuộc tính đơn giản được sử dụng để thiết lập kết nối vật lý với cơ sở dữ liệu. Mặc dù vậy, Session không an toàn khi sử dụng nên người dùng chỉ nên tạo chúng khi cần và đóng ngay sau khi đã hoàn thành công việc.

- Transaction: được sử dụng như đại diện cho một đơn vị công việc trong cơ sở dữ liệu. Các transaction chịu trách nhiệm hoạt động bảo vệ tính toàn vẹn của các thao tác của cơ sở dữ liệu. Tất cả các tác vụ sẽ tự động chạy lại nếu quá trình không may bị lỗi. Transaction có thể được sử dụng làm giao diện hoặc mã code.
- Query: lấy thông tin từ cơ sở dữ liệu và khởi tạo đối tượng thông qua SQL (Native SQL) hay Hibernate Query Language (HQL).
- Criteria: Quá trình lấy dữ liệu từ cơ sở dữ liệu cũng cần một số điều kiện nhất định. Các tiêu chí được sử dụng làm đối tượng để tạo và thực hiện các yêu cầu truy vấn.

Thực hành

- Mô tả bài tập: Xây dựng một ứng dụng quản lý thư viện sách sử dụng Hibernate ORM. Chương trình sẽ có các chức năng:
  - Thêm sách mới vào cơ sở dữ liệu.
  - Lấy danh sách tất cả sách có trong cơ sở dữ liệu.
  - Tìm kiếm sách theo ID.
  - Cập nhật thông tin sách.
  - Xóa sách khỏi cơ sở dữ liệu theo ID.
- Yêu cầu:
  - Sử dụng cơ sở dữ liệu MySQL.
  - Hibernate sẽ được sử dụng để thực hiện các thao tác với cơ sở dữ liệu.
  - Bảng trong cơ sở dữ liệu sẽ có tên là books với các cột:
    - id: Khóa chính, tự động tăng (Auto-increment).
    - title: Tên sách.
    - author: Tác giả của sách.
    - published_year: Năm xuất bản.

CMC UNIVERSITY

Aspire to Inspire the Digital World

THANK YOU