# Giới thiệu về Hibernate

Hibernate là một framework phát triển ứng dụng trong lĩnh vực lập trình Java.

Được sử dụng để giải quyết vấn đề liên quan đến lưu trữ và truy xuất dữ liệu từ cơ sở dữ liệu.

Vì sao nên dùng Hibernate?

• Ánh xạ đối tượng quan hệ - Object Relational Mapping (ORM)

• Quản lý mối quan hệ - Relationship Management

• Tối ưu hóa truy vấn - Query Optimization

ORM : có thể ánh xạ các lớp biến thành cơ sở dữ liệu

# JPA là gì?

Jakarta Persistence API (JPA) (bắt đầu từ Java EE9) (trước đây được hiểu là Java Persistence API)

Là một API tiêu chuẩn cho việc phát triển các framework ORM. Nó bao gồm các Interface và các phương thức yêu cầu hiện thực để sử dụng.

# Những lợi ích của JPA

Không bị phụ thuộc vào 1 nhà cung cấp cụ thể.

Dễ bảo trì, code linh động.

Nếu có một nhà cung cấp không còn tồn tại thì sao?

Tạo ra user trong CSDL

+ username

+ From host

+ password

+ Phân quyền

Query:

Tao user

CREATE USER 'spring'@'%' IDENTIFIED BY 'spring';

GRANT ALL PRIVILEGES ON \* . \* TO 'spring'@'%';

Tao Database

CREATE DATABASE

IF NOT EXISTS `QuanLySinhVien`

CHARACTER SET utf8mb4

COLLATE utf8mb4\_general\_ci;

Tạo Table

CREATE TABLE `sinhvien` (

`id` int NOT NULL AUTO\_INCREMENT,

`ho\_dem` varchar(45) DEFAULT NULL,

`ten` varchar(45) DEFAULT NULL,

`email` varchar(45) DEFAULT NULL,

PRIMARY KEY (`id`)

)

ENGINE=INNODB

AUTO\_INCREMENT=1

DEFAULT CHARSET=UTF8MB4

COLLATE=UTF8MB4\_UNICODE\_CI;

Them data

INSERT INTO `sinhvien` (`ho\_dem`, `ten`, `email`)

VALUES

('Nguyễn', 'Văn A', 'vana@gmail.com'),

('Trần', 'Thị B', 'thib@gmail.com'),

('Lê', 'Văn C', 'vanc@gmail.com'),

('Phạm', 'Thị D', 'thid@gmail.com'),

('Huỳnh', 'Văn E', 'vane@gmail.com'),

('Nguyễn', 'Thị F', 'thif@gmail.com'),

('Trần', 'Văn G', 'vang@gmail.com'),

('Lê', 'Thị H', 'thih@gmail.com'),

('Phạm', 'Văn I', 'vani@gmail.com'),

('Huỳnh', 'Thị K', 'thik@gmail.com');

EntityManager

● Với EntityManager, chúng ta có thể:

1. Lưu trữ đối tượng vào cơ sở dữ liệu bằng cách sử dụng phương thức persist().

2. Lấy đối tượng từ cơ sở dữ liệu bằng cách sử dụng phương thức find() hoặc các truy vấn (query methods).

3. Cập nhật đối tượng bằng cách sử dụng phương thức merge().

4. Xóa đối tượng bằng cách sử dụng phương thức remove().

Thực hành tạo dự án Spring Boot có sử dụng Hibernate

● Lưu ý các dependency:

○ Liên quan Databse mà bạn đang dùng, ví dụ như mysql-connector-j

○ spring-boot-stater-data-jpa

# Thực hành tạo dự án Spring Boot có sử dụng Hibernate

● Chúng ta cần cấu hình các thông tin để kết nối:

● spring.datasource.url=jdbc:mysql://localhost:3306/db\_name

● spring.datasource.username=db\_username

● spring.datasource.password=db\_password

● #spring boot tự động phát hiện ra driver-class

● #spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

=============================== JPQL ====================================

# JPQL (Java Persistence Query Language)

Là một ngôn ngữ truy vấn dựa trên đối tượng được sử dụng trong JPA (Java Persistence API). JPQL cho phép bạn truy vấn đối tượng trong cơ sở dữ liệu một cách độc lập với cơ sở dữ liệu cụ thể.

A computer screen with white text

Description automatically generated

FROM > ON > JOIN > WHERE > GROUP BY > HAVING > SELECT > ORDER BY

Thứ tự thực thi của truy vấn SQL được xác định bởi các mệnh đề trong truy vấn, theo thứ tự sau:

**1. FROM:**

* Mệnh đề FROM xác định các bảng hoặc view mà dữ liệu sẽ được lấy ra.
* Các bảng được liệt kê trong FROM được kết nối với nhau theo thứ tự được liệt kê.
* Ví dụ: SELECT \* FROM Customers JOIN Orders ON Customers.CustomerID = Orders.CustomerID;

**2. ON:**

* Mệnh đề ON được sử dụng cùng với JOIN để xác định điều kiện liên kết giữa các bảng.
* Điều kiện ON được đánh giá sau khi các bảng đã được kết nối.
* Ví dụ: SELECT \* FROM Customers JOIN Orders ON Customers.CustomerID = Orders.CustomerID AND Orders.OrderDate > '2023-01-01';

**3. JOIN:**

* Mệnh đề JOIN được sử dụng để kết nối các bảng dựa trên các cột có giá trị chung.
* Có nhiều loại JOIN khác nhau, bao gồm INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN, v.v.
* Loại JOIN được sử dụng sẽ ảnh hưởng đến kết quả truy vấn.
* Ví dụ: SELECT \* FROM Customers INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID;

**4. WHERE:**

* Mệnh đề WHERE được sử dụng để lọc dữ liệu dựa trên các điều kiện cụ thể.
* Điều kiện WHERE được đánh giá sau khi các bảng đã được kết nối và lọc theo các mệnh đề trước.
* Ví dụ: SELECT \* FROM Customers WHERE Country = 'Vietnam';

**5. GROUP BY:**

* Mệnh đề GROUP BY được sử dụng để nhóm dữ liệu theo các cột cụ thể.
* Sau khi nhóm được tạo, các hàm tổng hợp (như SUM, COUNT, AVG, MIN, MAX) có thể được áp dụng cho mỗi nhóm.
* Ví dụ: SELECT Country, COUNT(\*) AS CustomerCount FROM Customers GROUP BY Country;

**6. HAVING:**

* Mệnh đề HAVING được sử dụng để lọc các nhóm dữ liệu dựa trên các điều kiện cụ thể.
* Điều kiện HAVING được đánh giá sau khi các nhóm đã được tạo.
* Ví dụ: SELECT Country, COUNT(\*) AS CustomerCount FROM Customers GROUP BY Country HAVING CustomerCount > 100;

**7. SELECT:**

* Mệnh đề SELECT xác định các cột dữ liệu mà bạn muốn truy xuất.
* Các cột được liệt kê trong SELECT được lấy từ các bảng đã được kết nối, lọc và nhóm theo các mệnh đề trước.
* Ví dụ: SELECT Customers.CustomerID, Customers.Name, Orders.OrderID, Orders.OrderDate FROM Customers JOIN Orders ON Customers.CustomerID = Orders.CustomerID;

**8. ORDER BY:**

* Mệnh đề ORDER BY được sử dụng để sắp xếp dữ liệu theo các cột cụ thể.
* Dữ liệu có thể được sắp xếp theo thứ tự tăng dần hoặc giảm dần.
* Ví dụ: SELECT \* FROM Customers ORDER BY Name DESC;

**Lưu ý:**

* Thứ tự thực thi của truy vấn SQL có thể được tối ưu hóa bởi bộ xử lý truy vấn dựa trên các yếu tố khác nhau, chẳng hạn như cấu trúc truy vấn, chỉ số và thống kê bảng.
* Việc hiểu thứ tự thực thi của truy vấn SQL rất quan trọng để viết các truy vấn hiệu quả và có thể dự đoán được kết quả.

Bạn có thể tham khảo thêm các tài liệu sau để tìm hiểu chi tiết hơn về thứ tự thực thi của truy vấn SQL:

* <https://viblo.asia/p/thu-tu-thuc-hien-cho-cau-lenh-sql-XL6lA8ogZek>
* <https://khoahoc.vietjack.com/question/955417/thu-tu-cua-cac-menh-de-trong-cau-lenh-sql-select-nhu-the-nao-hay-chon-phuong-an-dung-trong-cac-phuon>
* <https://codelearn.io/learning/working-with-sql>

Hy vọng những thông tin này hữu ích cho bạn!

Lamda

for (**SinhVien** s:sinhVienList)  
{  
 **System**.*out*.println(s);  
}  
sinhVienList.forEach(  
 sinhVien -> {  
 **System**.*out*.println(sinhVien);  
 }  
);

# REST API - Representational State Transfer

<https://bykowski.pl/rest-api-efektywna-droga-do-zrozumienia/>

là một kiến trúc phần mềm được sử dụng để thiết kế các giao diện lập trình ứng dụng (API). REST API thường sử dụng giao thức HTTP để truyền tải dữ liệu giữa client và server.

Là một kiểu thiết kế và các quy ước cách để mà các máy client và server giao tiếp với nhau qua phương thức HTTP, các API này được xây dựng dựa trên nguyên tắc của REST (Representational State Transfer)

Là một kiến trúc phần mềm được sử dụng trong lập trình web để thiết kế các dịch vụ web dựa trên giao thức HTTP

## API - Application Programming Interface

https://www.leadmine.net/glossary/application-programming-interface/

API (Application Programming Interface) là một tập hợp các quy tắc, giao thức và công cụ cho phép hai phần mềm tương tác và trao đổi dữ liệu với nhau.

Nó định nghĩa các phương thức và định dạng dữ liệu mà các ứng dụng khác nhau có thể sử dụng để

giao tiếp và tương tác với nhau.

URL( uniform resource locator)

HTTP: Hyper Text Transfer protocol

Tài nguyên (Resources): Tài nguyên là các đối tượng có thể được truy cập hoặc được thao tác trong dịch vụ web. Mỗi tài nguyên được đại diện bằng một URI (Uniform Resource

Identifier) duy nhất.

• Ví dụ, trong một ứng dụng quản lý sách, tài nguyên có thể là

"sách" và URI có thể là "/books"

Giao thức HTTP: REST sử dụng các phương thức HTTP như GET, POST, PUT và

DELETE để thực hiện các hoạt động tương ứng với các tài nguyên (resources)

trên máy chủ.

• Ví dụ, để lấy thông tin về một cuốn sách từ dịch vụ web, ta sử dụng

phương thức GET trên URI "/books/{id}"

Định dạng dữ liệu (Data Formats): Dữ liệu truyền tải giữa máy khách và máy

chủ thông qua RESTful webservice thường được sử dụng các định dạng dữ liệu

như JSON (JavaScript Object Notation) hoặc XML (eXtensible Markup

Language).

Không trạng thái (Statelessness): RESTful webservice tuân thủ nguyên tắc

của giao tiếp không trạng thái, nghĩa là mỗi yêu cầu từ phía máy khách

phải chứa đủ thông tin để máy chủ hiểu và xử lý yêu cầu, không cần

lưu trạng thái trước đó.

## Có phụ thuộc vào ngôn ngữ lập trình không? KHÔNG

Data Format

• Có thể sử dụng bất kỳ định dạng nào

• Phổ biến: XML và JSON

• JSON (JavaScript Object Notation) được sử dụng nhiều nhất

Ví dụ: lấy thông tin IP

<https://ipinfo.io/161.185.160.93/geo>

Ví dụ: lấy thông tin thời tiết

<https://www.weatherbit.io/api/weather-current>

Ví dụ: lấy thông tin giá Bitcoin

<https://api.coindesk.com/v1/bpi/currentprice.json>

REST API

REST WEB SERVICES

REST SERVICES

RESTful API

RESTful WEB SERVICES

RESTful SERVICES

A group of arrows pointing to different directions

Description automatically generated

<https://www.edureka.co/blog/what-is-rest-api/>

## Mã lỗi web

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# JSON (JavaScript Object Notation)

là một định dạng dữ liệu dựa trên văn bản (text-based) dễ đọc và dễ viết cho việc truyền tải dữ liệu giữa các máy tính.

• Nó đã trở thành một tiêu chuẩn cho truyền tải dữ liệu giữa các ứng dụng web.

key-value

JSON được sử dụng để đại diện dữ liệu dưới dạng các cặp key-value.

Một đối tượng có thể gồm nhiều cặp giá trị.

Dữ liệu trong JSON được biểu diễn bằng cú pháp đơn giản và nhìn gần giống cú pháp

của JavaScript, do đó nó phù hợp với việc sử dụng trong ứng dụng web.

A screenshot of a computer code

Description automatically generated

Đối tượng JSON lồng nhau:

A computer code on a black background

Description automatically generated

Mảng JSON

A screenshot of a computer

Description automatically generated A computer code with text

Description automatically generated with medium confidence

A black background with white text

Description automatically generated A computer screen shot of text

Description automatically generated

# Spring REST Controller

@RestController là một annotation trong Spring Framework được sử dụng để đánh dấu một lớp (class) trong ứng dụng là một RESTful Controller.

● Nó kết hợp hai annotation khác là @Controller và @ResponseBody.

● Khi một lớp được đánh dấu bằng @RestController, nó cho phép Spring hiểu rằng lớp này chịu trách nhiệm *xử lý các yêu cầu RESTful từ phía client và trả về dữ liệu dưới dạng JSON hoặc XML* (thông qua @ResponseBody).

@RequestMapping

● Là một annotation trong Spring Framework được sử dụng để ánh xạ các yêu cầu HTTP tới các phương thức trong một Controller.

● Nó xác định đường dẫn URL và phương thức HTTP mà một phương thức cụ thể trong Controller sẽ xử lý.

● Annotation @RequestMapping có thể được đặt cả trên cấp lớp (class-level) và phương thức (method-level) trong Controller.

● Khi đặt @RequestMapping trên class, nó xác định tiền tố của đường dẫn URL cho tất cả các phương thức trong Controller.

A black and white text on a black background

Description automatically generated

A screenshot of a computer

Description automatically generated

# FINISH