**THAO TÁC VỚI DATABASE**

**Các annotation** **làm việc với db**

* **@Entity**: Đánh dấu một lớp Java là một entity, tức là một đại diện cho một bảng trong cơ sở dữ liệu
* **@Table**: Đánh dấu mapping giữa một entity và một bảng cụ thể trong cơ sở dữ liệu. Bằng cách sử dụng thuộc tính *name*, chúng ta có thể chỉ định tên của bảng. Nếu không có **@Table**, tên của bảng sẽ được coi là tên của lớp Java tương ứng
* **@Column:** Đánh dấu mapping giữa một trường trong entity và một cột trong bảng cơ sở dữ liệu. Bằng cách sử dụng thuộc tính name, chúng ta có thể chỉ định tên của cột. Nếu không có **@Column**, tên của cột sẽ được coi là tên của trường
  + Thuộc tính *columnDefinition* trong **@Column** được sử dụng để chỉ định định nghĩa cột cụ thể cho một trường trong cơ sở dữ liệu



* **@Id**: Đánh dấu một trường là khóa chính của entity, tức là trường đại diện cho cột PRIMARY KEY trong cơ sở dữ liệu
* **@GeneratedValue**: Đánh dấu một trường khóa chính sẽ được tự động tăng giá trị
* **@Formula và @Transient**: Chỉ định rằng một trường của đối tượng không được lưu trữ vào cơ sở dữ liệu

Diagram

Description automatically generated

* + **@Formula:** Đánh dấu một trường của đối tượng là một trường được tính toán dựa trên một biểu thức SQL. Trường này không được lưu trữ vào cơ sở dữ liệu, mà chỉ được sinh động ngay khi truy vấn dữ liệu từ cơ sở dữ liệu. Kết quả được tính toán sẽ có sẵn trong đối tượng sau khi truy vấn hoàn thành

A picture containing text, font, screenshot, line

Description automatically generated

Graphical user interface, text

Description automatically generated

* + **@Transient**
    - Đánh dấu một trường của đối tượng là một trường tạm thời (transient field) không được lưu trữ vào cơ sở dữ liệu. Trường này chỉ tồn tại trong quá trình thực thi ứng dụng Java (runtime), không ảnh hưởng đến cơ sở dữ liệu. Điều này có nghĩa là trường được bỏ qua trong quá trình mapping và không tham gia vào các hoạt động CRUD
    - Với **@Transient**, trường sẽ sinh động khi getter của trường được gọi trong quá trình thực thi ứng dụng Java. Khi gọi getter, chúng ta có thể thực hiện bất kỳ xử lý nào để trả về giá trị của trường dựa trên các trạng thái hiện tại của đối tượng. Tuy nhiên, giá trị của trường này không được lưu trữ và không ảnh hưởng đến cơ sở dữ liệu

Graphical user interface, text, application

Description automatically generated

* **@Temporal**: Đánh dấu kiểu dữ liệu ngày tháng trong các trường của đối tượng được ánh xạ vào cơ sở dữ liệu
  + **@Temporal(TemporalType.DATE)**: Đánh dấu kiểu dữ liệu ngày tháng dạng DATE, chỉ lưu trữ ngày (không bao gồm giờ, phút, giây)
  + **@Temporal(TemporalType.TIME)**: Đánh dấu kiểu dữ liệu ngày tháng dạng TIME, chỉ lưu trữ giờ, phút, giây (không bao gồm ngày)
  + **@Temporal(TemporalType.TIMESTAMP)**: Đánh dấu kiểu dữ liệu ngày tháng dạng TIMESTAMP, lưu trữ cả ngày và giờ, phút, giây

A picture containing text, font, white

Description automatically generated

* **@Enumerated**:Ánh xạ giá trị của enum vào một cột trong cơ sở dữ liệu. Có hai giá trị EnumType mà chúng ta có thể sử dụng:
  + EnumType.ORDINAL: Giá trị của enum được chuyển đổi thành một số nguyên, tức là vị trí của phần tử trong enum
  + EnumType.STRING: Giá trị của enum được chuyển đổi thành một chuỗi, tức là tên của phần tử enum

A black and white text

Description automatically generated

* **@JsonIgnore**: Là một annotation trong thư viện Jackson dùng để đánh dấu một trường (field) trong lớp Java để bỏ qua quá trình serialize và deserialize thành JSON. Khi một trường được đánh dấu bằng **@JsonIgnore**, nó sẽ không được bao gồm trong JSON được tạo ra từ đối tượng và sẽ bị bỏ qua khi deserialize JSON thành đối tượng Java

A picture containing text, font, screenshot, algebra

Description automatically generated

**Tạo column unique trong database**

* **@Column(unique = true)**: Chúng ta có thể sử dụng annotation **@Column(unique = true)** để đánh dấu trường là duy nhất trong cơ sở dữ liệu



* **@UniqueConstraint**: Chúng ta có thể sử dụng annotation này để chỉ định một nhóm hoặc nhiều nhóm các thuộc tính là duy nhất cùng nhau
  + Một nhóm:

A picture containing text, screenshot, font, line

Description automatically generated

* + Nhiều nhóm:

A picture containing text, screenshot, font, line

Description automatically generated

**Cấu hình DataSource**

* Trong dự án Spring Boot, chúng ta có thể sử dụng các thuộc tính trong file application.properties hoặc application.yml để cấu hình DataSource một cách dễ dàng mà không cần phải viết mã cấu hình chi tiết trong code. Spring Boot đã tích hợp sẵn các cấu hình mặc định cho nhiều loại cơ sở dữ liệu phổ biến
* Việc sử dụng các thuộc tính trong file application.properties cho phép chúng ta tuỳ chỉnh cấu hình DataSource một cách linh hoạt và dễ dàng. Các thuộc tính như spring.datasource.url, spring.datasource.username, spring.datasource.password, spring.datasource.driver-class-name, ... cho phép chúng ta chỉ định thông tin kết nối cơ sở dữ liệu
* Khi chúng ta cấu hình tường minh một DataSource trong Spring Boot bằng cách sử dụng Java Config, cấu hình tự động của Spring Boot sẽ bị ghi đè (overridden). Spring Boot sẽ tạo ra một DataSource từ Bean và sử dụng nó để kết nối với cơ sở dữ liệu theo cấu hình tường minh chúng ta đã cung cấp

**A screenshot of a computer code

Description automatically generated with low confidence**

* Ngoài cấu hình tường minh DataSource trong Spring Boot, chúng ta cũng có thể cấu hình transaction để quản lý giao dịch trong ứng dụng của mình

**A screenshot of a computer code

Description automatically generated with medium confidence**

A picture containing text, font, screenshot, document

Description automatically generated

* Trong Spring Framework và Spring Boot, ba đối tượng quan trọng để làm việc với cơ sở dữ liệu là DataSource, EntityManagerFactory và TransactionManager. Dưới đây là ý nghĩa và mối quan hệ giữa ba đối tượng này:
  + DataSource: Đây là một đối tượng giúp thiết lập và quản lý kết nối đến cơ sở dữ liệu. DataSource chịu trách nhiệm cung cấp các kết nối đến cơ sở dữ liệu cho EntityManagerFactory và các hoạt động truy vấn dữ liệu. DataSource định nghĩa các thông tin như URL, tên người dùng, mật khẩu và các thuộc tính khác cần thiết để kết nối đến cơ sở dữ liệu
  + EntityManagerFactory: Đây là một đối tượng quan trọng trong JPA (Java Persistence API). EntityManagerFactory là một factory object để tạo ra các EntityManager. Nó là trung tâm để quản lý các entity (đối tượng) và thực hiện các hoạt động với cơ sở dữ liệu. EntityManagerFactory được cấu hình để tương tác với cơ sở dữ liệu thông qua DataSource và định nghĩa các thuộc tính như packagesToScan để quét các entity, JpaVendorAdapter để xác định nhà cung cấp JPA sử dụng (ví dụ như Hibernate) và các thuộc tính JPA/Hibernate khác
  + TransactionManager: Đây là một đối tượng quản lý giao dịch. Nó giúp quản lý quá trình giao dịch trong ứng dụng, đảm bảo tính nhất quán và an toàn của các thao tác cơ sở dữ liệu. TransactionManager quản lý các transaction (giao dịch) và cung cấp các phương thức để bắt đầu, commit hoặc rollback các giao dịch. Nó làm việc cùng với EntityManagerFactory để đảm bảo rằng các thay đổi dữ liệu được thực hiện trong một transaction được xác nhận hoặc hủy bỏ đúng cách
* Mối quan hệ giữa ba đối tượng này như sau:
  + DataSource được sử dụng bởi EntityManagerFactory để thiết lập kết nối đến cơ sở dữ liệu và thực hiện các hoạt động truy vấn
  + EntityManagerFactory sử dụng thông tin từ DataSource để tạo ra các EntityManager, quản lý các entity và thực hiện các hoạt động với cơ sở dữ liệu
  + TransactionManager sử dụng EntityManagerFactory để quản lý các giao dịch (transaction) và đảm bảo tính nhất quán và an toàn của các thao tác cơ sở dữ liệu
* Khi chúng ta sử dụng **@SpringBootApplication** trong ứng dụng chính của mình, **@EnableAutoConfiguration** giúp Spring Boot quét classpath và tự động cấu hình repository JPA dựa trên các dependency có sẵn, mà không yêu cầu chúng ta phải cung cấp **@EnableJpaRepositories** một cách tường minh

A black text on a white background

Description automatically generated with low confidence

Tương tự, chúng ta củng không cần phải sử dụng **@EnableTransactionManagement** một cách tường minh. Spring Boot sẽ tự động kích hoạt transaction management nếu chúng ta sử dụng **@EnableAutoConfiguration** trong ứng dụng

**Cấu hình multiple datasources**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Cấu hình college-db (cấu hình student-db tương tự)

A picture containing text, screenshot, font

Description automatically generated

A picture containing text, screenshot, font

Description automatically generated

A picture containing text, font, line, screenshot

Description automatically generated

**@ConfigurationProperties**:là một annotation được sử dụng để liên kết các giá trị cấu hình từ tệp cấu hình (ví dụ: application.properties, application.yml) với các thuộc tính của một bean