EntityManagerFactory emf =

Persistence.createEntityManagerFactory("n\_1\_problem");

EntityManager em = emf.createEntityManager();

em.getTransaction().begin();

// NORMAL SCRIPT

// List<Post> posts = em.createQuery("select p from Post p").getResultList();

// JOIN FETCH SCRIPT

List<Post> posts = em.createQuery("select p from Post p join fetch p.postComments pc").getResultList();

for (Post post : posts) {

System.out.println("Post Comments: " + post.getPostComments().size());

}

em.getTransaction().commit();

em.close();

emf.close();

**@Entity** annotation dùng để chỉ định một mapping class tương ứng với một bảng trong database.

* @Entity(name="student")
* @Table(name="STUDENT")

Các bạn cần phân biệt giữa @Entity(name = ‘EntityName’) và @Table(name = “TableName”) vì mình thấy rất nhiều bạn bị nhầm lẫn, @Entity(name = ‘EntityName’) dùng để chỉ tên của entity được Hibernate quản lý trong khi @Table(name = “TableName”) chỉ đích danh tên của table dưới database. Ví dụ khi chúng ta sử dụng HQL để truy vấn thì chúng ta cần chỉ định **EntityName** và Hibernate sẽ dựa vào đó để ánh xạ thành ***TableName*** tương ứng.

* **@ID** chỉ định các thuộc tính của khoá chính trong table Message.
* **@GeneratedValue** dùng để sinh giá trị tự động cho ID.

JPA hỗ trợ generate primary key tự động với @GeneratedValue annotation bao gồm 4 kiểu: AUTO, IDENTITY, SEQUENCE, TABLE. Nếu chúng ta không chỉ định 1 trong 4 tuỳ chọn này thì Hibernate mặc định là AUTO.

@GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "book\_generator")

@SequenceGenerator(name="book\_generator", sequenceName = "book\_seq", initialValue = 10)

<https://shareprogramming.net/generate-primary-key-trong-hibernate-jpa/>

private UUID id;

@GeneratedValue(strategy = GenerationType.AUTO)

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column annotation

@Column dùng để mô tả các thông tin của một cột trong database bao gồm name, length, nullable, unique.

@Column(name="STUDENT\_NAME", length=50, nullable=false, unique=false)

@Transient

@Transient dùng để đánh dấu một thuộc tính trong entity class không phải là một cột tương ứng trong database. Ví dụ chúng ta có thuộc tính age được tính bằng công thức năm hiện tại trừ đi năm sinh, do thuộc age không cần lưu trong database.

@Enumerated

@Enumerated(EnumType.STRING)

@ManyToOne

@JoinColumn ( dung ở trong entity chứa khóa ngoại – tên cột )

@OneToMany(mappedBy = "orderItem")

Mappedby lấy dữ liệu từ truongf nào của bảng liên kết

@JoinTable

@JoinTable(name = "store\_product",

joinColumns = { @JoinColumn(name = "fk\_store") },

inverseJoinColumns = { @JoinColumn(name = "fk\_product") })

@ManyToMany

@OnToOne

@OneToMany(mappedBy = "order", fetch = FetchType.LAZY)

FetchType.Lazy : chỉ tìm kiếm trong 1 bảng theo entity

FetchType.Eager : tìm kiếm toàn bộ dữ liệu các bảng liên kết với nhau

N+1 problem suwr dungj  JOIN FETCH

**@Fetch**

FetchMode.JOIN, FetchMode.SUBSELECT, FetchMode.SELECT.

@Fetch(FetchMode.SELECT)

**@BatchSize**

@BatchSize(size=10)

* [Access modifiers](https://shareprogramming.net/access-modifiers-trong-java-voi-vi-du-cu-the/)composite primary key class phải là **public**.
* Nó phải có một [no-arg constructor](https://shareprogramming.net/constructor-trong-java/).
* Phải định nghĩa[equals() và hashCode()](https://shareprogramming.net/moi-quan-he-giua-equals-va-hashcode-trong-java/) method.
* Cuối cùng là phải implement [Serializable](https://shareprogramming.net/marker-interface-serializable-trong-java/)

### @IdClass

@NoArgsConstructor

@AllArgsConstructor

@EqualsAndHashCode

public **class** AccountId **implements** Serializable {

private String accountNumber;

private String accountType;

}

-----------------------------------------

@Entity

@IdClass(AccountId.class)

@Getter

@Setter

@AllArgsConstructor

public **class** Account {

@Id

private String accountNumber;

@Id

private String accountType;

private String name;

}

@EmbeddedId

@Embeddable

@EqualsAndHashCode

@NoArgsConstructor

@AllArgsConstructor

public **class** BookId **implements** Serializable {

private String title;

private String language;

}

@Entity

@Getter

@Setter

@AllArgsConstructor

@NoArgsConstructor

public **class** Book {

@EmbeddedId

private BookId bookId;

private String author;

}

@Size

@Length

@Column (length=value)

@Enumerated

@Enumerated(EnumType.ORDINAL)

**@Enumerated(EnumType.STRING)**