Entity

|  |  |
| --- | --- |
| @Entity  @Table(name="Orders")  public class Order implements Serializable { | Khai báo Entity của bảng Orders |
| @Id | Khai báo cột khóa chính của bảng |
| @GeneratedValue(strategy = GenerationType.IDENTITY) | Khai báo cột với giá trị tự sinh ra  Tong database có từ khóa IDENTITY |
| @Column(name="Password")  private String password; | Khai báo biến match với cột trong database.  Nếu tên cột trong database và tên biến tuân theo luật Javabean (viết thường chữ hoa đầu của tên) thì không cần khai báo @Column |
| @JsonIgnore | Dùng cho các field không có Jackson.  Một class sẽ tự độngn tạo các Jackson( JSON) thông qua các field hiện có của nó và gửi xuống database.  Nếu thông tin của một field được lấy thông qua Entity khác, nó không cần JSON |
| @JsonIgnore  @OneToMany(mappedBy="customer", fetch=FetchType.EAGER)  private List<Order> orders; | Truyền từ lớp con lên lớp cha (lớp có khóa ngoài)  @JsonIgnore: Bỏ qua Json  @OneToMany định nghĩa quan hệ giữa customer và orders là 1-n. Trong đó có thuộc tính fetch:  fetch = FetchType.LAZY tức là khi bạn find, select đối tượng Company từ database thì nó sẽ không lấy các đối tượng Employee liên quan  fetch = FetchType.EAGER tức là khi bạn find, select đối tượng Company từ database thì tất cả các đối tượng Employee liên quan sẽ được lấy ra và lưu vào listEmployee  \* Lưu ý:  fetch = FetchType.LAZY tức là mặc định không lấy ra các đối tượng liên quan nhưng bên trong transaction, bạn gọi method company.getListEmployee() thì nó vẫn có dữ liệu nhé, bởi vì khi bạn gọi method nó sẽ query các đối tượng Employee liên quan và lưu vào listEmployee, và khi kết thúc transaction listEmployee sẽ chứa các employee liên quan. Tuy nhiên nếu bạn không gọi method đó thì listEmployee không có dữ liệu và khi kết thúc transaction listEmployee sẽ không có đối tượng employee nào  fetch = FetchType.EAGER thì khi lấy đối tượng Company là nó mặc định query luôn các đối tượng Employee liên quan và lưu vào listEmployee, do đó khi kết thúc transaction, listEmployee sẽ có chứa các đối tượng Employee của Company đó. |
| @ManyToOne()  @JoinColumn(name="CustomerId")  private Customer customer; | Truyền từ lớp cha (lớp có khóa ngoài) về lớp con  @JoinColumn(name="CustomerId") |
| @DateTimeFormat(pattern="MM/dd/yyyy")  @Temporal(TemporalType.TIMESTAMP)  private Date orderDate = new Date(); | @DateTimeFormat : Format cho dữ liệu dạng Date  Khai báo cho dữ liệu dạng Date  TemporalType.TIMESTAMP: Từ năm đến nano giây  TemporalType.DATE: Ngày tháng năm |
|  |  |