Hibernate FetchType là gì? Phân biệt FetchType Lazy với Eager

Posted on *Tháng Mười Một 2, 2017*

Hibernate FetchType là gì? Phân biệt FetchType Lazy với Eager.

(Xem thêm: Hướng dẫn tự học Hibernate)

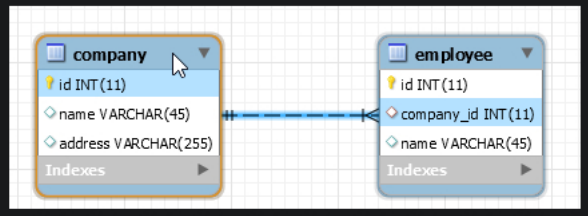
(Xem thêm: Code ví dụ Hibernate FetchType = EAGER (Eager loading))

(Xem thêm: Code ví dụ Hibernate FetchType = LAZY (Lazy loading))

Hibernate FetchType là gì?

Trong Hibernate, FetchType là một thuộc tính trong các annotation @OneToOne, @OneToMany, @ManyToOne, @ManyToMany, được dùng để định nghĩa phương thức lấy các đối tượng liên quan.

Ví dụ mình có quan hệ sau:



Quan hệ giữa company và employee là một-nhiều.

Khi mapping sang class với Hibernate sẽ như sau:

```
@Entity
@Table(name = "company")
public class Company {

  //...

  @OneToMany(fetch = FetchType.LAZY, mappedBy = "company")
  private List<Employee> listEmployee = new ArrayList<>();

}
```


HIBERNATE

Phân biệt save, persist, update,

Annotation `@OneToMany` định nghĩa quan hệ giữa company và employee 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.

FetchType mặc định

- Với annotation `@ManyToOne` và `@OneToOne` thì fetchType mặc định là `EAGER`
- Với annotation `@ManyToMany` và `@OneToMany` thì fetchType mặc định là `LAZY`

Có sự khác nhau như trên là vì với annotation `@ManyToOne`, và `@OneToOne` thì khi select với `fetchType = EAGER` nó chỉ lấy ra nhiều nhất 1 đối tượng liên quan nên không ảnh hưởng gì tới performance. Còn nếu sử dụng `fetchType = EAGER` với `@ManyToMany`, `@OneToMany` thì có thể nó lấy ra rất nhiều đối tượng liên quan dẫn tới làm giảm hiệu năng, tốn bộ nhớ.

Ưu nhược điểm của mỗi loại FetchType

Với FetchType = LAZY(Lazy Loading):

- Ưu điểm: tiết kiệm thời gian và bộ nhớ khi select
- Nhược điểm: gây ra lỗi `LazyInitializationException`, khi muốn lấy các đối tượng liên quan phải mở transaction 1 lần nữa để query

Với FetchType = EAGER(Eager Loading):

- Ưu điểm: có thể lấy luôn các đối tượng liên quan, xử lý đơn giản, tiện lợi
- Nhược điểm: tốn nhiều thời gian và bộ nhớ khi select, dữ liệu lấy ra bị thừa, không cần thiết.

Okay, Done!

References:

https://docs.jboss.org/hibernate/jpa/2.1/api/javax/persistence/FetchType.html

This entry was posted in *Hibernate*. Bookmark the *permalink*.

merge, saveOrUpdate trong hibernate

Sự khác nhau giữa merge với saveOrUpdate trong Hibernate

Hibernate Batch Processing là gì? Batch Processing trong Hibernate

Sự khác nhau giữa load() và get() trong Hibernate

Code ví dụ hibernate annotation @CreationTimestamp, @UpdateTimestamp (thời gian tạo/sửa)

Code ví dụ Hibernate annotation @Version (Hibernate Locking Version)

Locking trong Hibernate, so sánh Optimistic lock với Pessimistic lock

ORM là gì? Tổng quan về ORM Framework

Code ví dụ Hibernate ID tự tăng (@GeneratedValue, @GenericGenerator)

Code ví dụ Hibernate @ElementCollection, lưu dữ liệu dạng list

Code ví dụ Hibernate @EmbeddedId, @Embeddable, Id gồm nhiều column

Code ví dụ Hibernate tự sinh ID dạng text, String

Code ví dụ Hibernate FetchType = LAZY (Lazy loading)

orphanRemoval là gì? Code ví dụ Hibernate orphanRemoval = true

Code ví dụ Hibernate cascade,

← Sự khác nhau giữa Cascade REMOVE/DELETE với orphanRemoval = true

Code ví dụ gửi gmail bằng Java – demo →

annotation @Cascade

Code ví dụ Hibernate FetchType = EAGER (Eager loading)

Code ví dụ Hibernate @Enumerated, lưu dữ liệu dạng Enum

So sánh sự khác nhau giữa @ElementCollection và @OneTomany

Code ví dụ Hibernate @OneToOne – Quan hệ một – một

Code ví dụ Hibernate @ManyToMany – Quan hệ nhiều nhiều

Code ví dụ Hibernate One To Many (@OneToMany, @ManyToOne)

hibernate.dialect là gì – Các loại SQL Dialects trong hibernate

Hibernate configuration – Các thông tin cấu hình hibernate

Code ví dụ Hibernate Pagination, phân trang trong hibernate

Code ví dụ với JPA Callbacks method @PrePersist, @PreUpdate, @PostRemove

So sánh Hibernate Criteria với HQL, HSQL /JPQL

Code ví dụ Hibernate Criteria (Hibernate Criteria Queries vs Restrictions)

Code ví dụ Hibernate Named Query (annotation @NamedQuery, @NameQueries)

Sự khác nhau giữa openSession() và getCurrentSession() trong

CHUYÊN MỤC

- Algorithm
- Apache
- Apache JMeter
- Apache Kafka
- AWS
- C/C++
- CDI
- Clean Code
- Demo
- Design Pattern
- Docker
- Eclipse
- Elasticsearch
- Excel
- FAQ
- Framework
- Freemaker
- FreeMarker
- Gradle
- Hibernate
- HttpClient
- HttpComponents
- Install
- Intellij IDEA
- Java
- Java Basic
- Java Core

Java8

JavaScript

jooq

JSF

JSP-Servlet

JUnit

Library

Linux

Maven

MongoDB

MySQL

Network Programming

Node.js

OOP

PostgreSQL

PrimeFaces

Principle

Python

quartz

Redis

SDKMan

Security

SocketCluster

Spring

Spring Boot

Spring Core

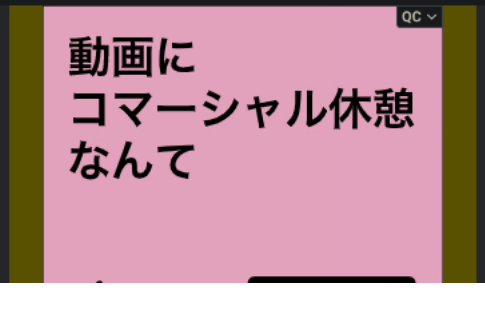
Spring Data

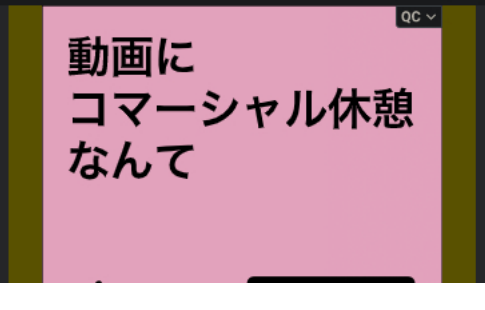
Spring Hibernate

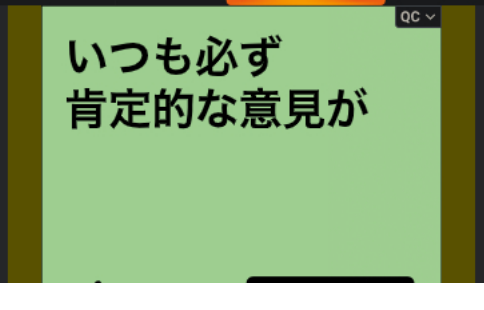
Spring JDBC

Spring MVC

Spring Security
Thymeleaf
Tomcat
Uncategorized
Web Service
WebSocket
Wordpress








vimeo
詳細を見る
vimeo
詳細を見る
vimeo
詳細を見る

stackjava.com