**Date: 19/10/2020**

**Spring Boot 7AM**

**Mr. RAGHU**

**-----------------------------------------------**

**one-to-one | many-to-many | joins**

**One-To-One**

\*) To implement one-to-one concept, we use many-to-one add one join-column with Unqiue

condition.

\*) Unqiue condition no duplicates are allowed into FK column.

\*\*\*) Unique condition columns allows null values even multiple times.

Task: Try to use @OneToOne and @JoinColumn(name="") for given example and find-out

FK column name in tables?

---code-----------

1. Model class

package in.nareshit.raghu.model;

import javax.persistence.Entity;

import javax.persistence.Id;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

@Entity

public class Profile {

@Id

private Integer profId;

private String pcode;

private String idType;

private String idNumer;

private String addr;

}

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

package in.nareshit.raghu.model;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

@Entity

public class Person {

@Id

private Integer id;

private String fname;

private String lname;

private String contact;

private String gender;

//@OneToOne

//@JoinColumn(name="pr\_id\_fk")

@ManyToOne

@JoinColumn(name="pr\_id\_fk",unique = true)

private Profile prob; //HAS-A

}

-----

2. Repository Interface

package in.nareshit.raghu.repo;

import org.springframework.data.jpa.repository.JpaRepository;

import in.nareshit.raghu.model.Profile;

public interface ProfileRepository

extends JpaRepository<Profile, Integer> {

}

package in.nareshit.raghu.repo;

import org.springframework.data.jpa.repository.JpaRepository;

import in.nareshit.raghu.model.Person;

public interface PersonRepository

extends JpaRepository<Person, Integer> {

}

3. Runner class

package in.nareshit.raghu.runner;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

import in.nareshit.raghu.model.Person;

import in.nareshit.raghu.model.Profile;

import in.nareshit.raghu.repo.PersonRepository;

import in.nareshit.raghu.repo.ProfileRepository;

@Component

public class DataInsertRunner implements CommandLineRunner {

@Autowired

private ProfileRepository prRepo;

@Autowired

private PersonRepository peRepo;

@Override

public void run(String... args) throws Exception {

Profile p1 = new Profile(50, "P1", "PANCARD", "AABB55001", "HYD");

Profile p2 = new Profile(51, "P2", "DL", "LOB85651", "CHN");

Profile p3 = new Profile(52, "P3", "AADHAR", "PIBB2419", "BAN");

prRepo.save(p1);

prRepo.save(p2);

prRepo.save(p3);

Person pe1 = new Person(1501, "S", "A", "550-65847", "MALE", p1);

Person pe2 = new Person(1502, "H", "B", "96-0065847", "MALE", p2);

Person pe3 = new Person(1503, "K", "C", "770-5665847", "FEMALE", p3);

peRepo.save(pe1);

peRepo.save(pe2);

peRepo.save(pe3);

}

}

4. application.properties

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

spring.datasource.url=jdbc:mysql://localhost:3306/boot7am

spring.datasource.username=root

spring.datasource.password=root

spring.jpa.show-sql=true

spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect

spring.jpa.hibernate.ddl-auto=create

========================================================================

**Many-To-Many**

In this case one extra table | 3rd table is created, which is called as JoinTable with 2 FK columns. They are join-column and inverse join-column.

a) Join-column : FK column connected to Parent table PK

b) Inverse Join Column : FK Column connected to child table PK.

---code---

1. Model

package in.nareshit.raghu.model;

import javax.persistence.Entity;

import javax.persistence.Id;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

@Entity

public class Author {

@Id

private Integer aid;

private String aname;

private String addr;

}

-------

package in.nareshit.raghu.model;

import java.util.List;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.JoinTable;

import javax.persistence.ManyToMany;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

@Entity

public class Book {

@Id

private Integer bid;

private String bname;

private String btype;

@ManyToMany

@JoinTable(

name="book\_auth\_tab",

joinColumns = @JoinColumn(name="bid\_fk"),

inverseJoinColumns = @JoinColumn(name="aid\_fk")

)

private List<Author> aobs;

}

========================================================================

2. Repository Interface

package in.nareshit.raghu.repo;

import org.springframework.data.jpa.repository.JpaRepository;

import in.nareshit.raghu.model.Author;

public interface AuthorRepository

extends JpaRepository<Author, Integer> {

}

package in.nareshit.raghu.repo;

import org.springframework.data.jpa.repository.JpaRepository;

import in.nareshit.raghu.model.Book;

public interface BookRepository

extends JpaRepository<Book, Integer> {

}

3. Runner class for Data Insert

package in.nareshit.raghu.runner;

import java.util.Arrays;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

import in.nareshit.raghu.model.Author;

import in.nareshit.raghu.model.Book;

import in.nareshit.raghu.repo.AuthorRepository;

import in.nareshit.raghu.repo.BookRepository;

@Component

public class DataInsertRunner implements CommandLineRunner {

@Autowired

private AuthorRepository arepo;

@Autowired

private BookRepository brepo;

@Override

public void run(String... args) throws Exception {

Author a1 = new Author(60, "SAM", "HYD");

Author a2 = new Author(61, "SYED", "CHN");

Author a3 = new Author(62, "RAM", "HYD");

Author a4 = new Author(63, "KHAN", "DHL");

arepo.save(a1);

arepo.save(a2);

arepo.save(a3);

arepo.save(a4);

Book b1 = new Book(98001, "CORE JAVA", "PRGM", Arrays.asList(a2,a3));

Book b2 = new Book(98002, "DP", "THERY", Arrays.asList(a1,a2));

Book b3 = new Book(98003, "CODE TYPE", "MATHS", Arrays.asList(a3,a4));

brepo.save(b1);

brepo.save(b2);

brepo.save(b3);

}

}

4. application.properties

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

spring.datasource.url=jdbc:mysql://localhost:3306/boot7am

spring.datasource.username=root

spring.datasource.password=root

spring.jpa.show-sql=true

spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect

spring.jpa.hibernate.ddl-auto=create

=============================================================================

Task:

Q) What is FetchType in JPA ? Provide example?

Q) What is CascadeType in JPA ? Provide example?

Q) What is Bi-directional mapping in JPA ? Provide example?

Q) Write Syntax to SQL based Joins ? with one example?

Q) Write Syntax for JPA/Hibernate based Joins ? with one example?

https://github.com/javabyraghu?tab=repositories&q=hibernate&type=&language=