>

>

>

>

>

Improvized Many To Many Association Mapping Example App

=======

=======

BootJpaProj14-Association Mapping-ManyToMany [boot]

src/main/java

#com.nt

> BootJpaProj14AssociationMapping ManyToManyApp.java

com.nt.entity

> D Faculty.java

Student.java

#com.nt.repository

> IFacultyRepository.java

>

IStudentRepository.java

com.nt.runners

> ManyToManyAssociation TestRunner.java

>

com.nt.service

CollegeMgmtServiceImpl.java

> ICollegeMgmtService.java

src/main/resources

application.properties

src/test/java

JRE System Library [JavaSE-21] Maven Dependencies

target/generated-sources/annotations target/generated-test-sources/test-annotations

src

target

UCID md

//parent class

//Student.java

package com.nt.entity;

import java.util.HashSet;

```
import java.util.Set;
import jakarta.persistence.CascadeType; import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.FetchType; import jakarta.persistence.GeneratedValue; import
jakarta.persistence.GenerationType; import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn; import jakarta.persistence.JoinTable; import
jakarta.persistence.ManyToMany; import jakarta.persistence.SequenceGenerator;
import jakarta.persistence.Table;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;
import lombok.Setter;
@Entity
@Table(name="MTM_STUDENT_TAB")
@Setter
@Getter
@AllArgsConstructor
@RequiredArgsConstructor
public class Student {
@Id
@SequenceGenerator(name="gen1",sequenceName = "SID_SEQ",initialValue = 100,allocationSize = 1)
@GeneratedValue(generator = "gen1",strategy = GenerationType.SEQUENCE)
private Integer sid;
@Column(length = 30)
@NonNull
private String sname;
@Column(length = 30)
@NonNull
private String saddrs;
@ManyToMany(targetEntity = Faculty.class,cascade = CascadeType.ALL,fetch = FetchType.EAGER)
@JoinTable(name="MTM_STUDS_FACULTIES_TAB",
joinColumns = @JoinColumn(name="STUD_ID",referencedColumnName = "SID"),
inverseJoinColumns=@JoinColumn(name="FACULTY_ID",referencedColumnName = "FID"))
private Set<Faculty> faculties=new HashSet<Faculty>();
public Student() {
System.out.println("Student:: 0-param constructor");
}
//toString() alt+shift+s,s
@Override
public String toString() {
return "Student [sid=" + sid + ", sname=" + sname + saddrs=" + saddrs + "]";
}
}
//Faculty.java (Child class)
package com.nt.entity;
import java.util.HashSet;
import java.util.Set;
import jakarta.persistence.CascadeType;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.FetchType;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.ManyToMany;
import jakarta.persistence.SequenceGenerator;
import jakarta.persistence.Table;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NonNull;
import lombok. RequiredArgsConstructor;
import lombok.Setter;
@Entity
@Table(name="MTM_FACULTY_TAB")
@Setter
@Getter
@AllArgsConstructor
@RequiredArgsConstructor
public class Faculty {
@Id
@SequenceGenerator(name="gen1",sequenceName = "FID_SEQ",initialValue = 1000, allocationSize = 1)
@GeneratedValue(generator = "gen1",strategy = GenerationType.SEQUENCE)
private Integer fid;
```

```
@Column(length = 30)
@NonNull
}
private String fname;
@Column(length = 30)
@NonNull
private String subject;
@Column(length = 30)
@NonNull
private String qlfy;
@ManyToMany(targetEntity = Student.class, cascade = CascadeType.ALL, fetch =
FetchType.EAGER,mappedBy = "faculties") private Set<Student> students=new HashSet<Student>();
// O-param constructor
public Faculty() {
System.out.println("Faculty:: O-param constructor");
}
//toString() (alt+shift+s,s)
@Override
public String toString() {
return "Faculty [fid=" + fid + ", fname=" + fname + ", subject=" + subject + ", qlfy=" + qlfy + "]";
}
//IStudentRepository.java
package com.nt.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import com.nt.entity.Student;
public interface IStudentRepository extends JpaRepository<Student, Integer> {
}
//IFacultyRepository.java
package com.nt.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import com.nt.entity.Faculty;
public interface IFaculty Repository extends JpaRepository<Faculty, Integer> {
}
//ServiceInterface
package com.nt.service;
import java.util.List;
import com.nt.entity.Faculty;
import com.nt.entity.Student;

public interface ICollegeMgmtService {

public String registerStudent(List<Student> list); public String
registerFacultiesAndTheirStudents(List<Faculty> list); public List<Student>
showAllStudentsAndTheirFaculties();

public List<Faculty> showAllFacultiesAndThierStudents(); public String removeStudent From Faculty(int
fid,int sid); public String removeFacultyFromStudent(int sid,int fid);

}

//SErvice Impl class

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

//ServiceImpl class

package com.nt.service;

import java.util.List;

import java.util.Optional;

import java.util.Set;

import java.util.stream.Collectors;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Propagation; import
org.springframework.transaction.annotation.Transactional;

import com.nt.entity.Faculty;

import com.nt.entity.Student;

import com.nt.repository.IFaculty Repository;

import com.nt.repository.IStudentRepository;

@Service

public class CollegeMgmtServiceImpl implements ICollegeMgmtService {

@Autowired

private IStudentRepository studRepo;

@Autowired

private IFaculty Repository facultyRepo;

@Override

}

public String registerStudent(List<Student> list) {

//save the batch

List<Student> saved List=stud Repo.saveAll(list);

//get only id values of the saved List

List<Integer> ids=saved List.stream().map(Student::getSid).collect(Collectors.toList());

return "Students and associated Faculties are saved with id values ::"+ids;
}
@Override
public String registerFacultiesAndTheir Students(List<Faculty> list) {
//save the batch
List<Faculty> saved List=faculty Repo.saveAll(list);
//get only id values of the saved List
List<Integer> ids=saved List.stream().map(Faculty::getFid).collect(Collectors.toList());
return "Faculties and associated Students are saved with id values ::"+ids;
}
@Override
public List<Student> showAllStudentsAndTheirFaculties() {
return studRepo.findAll();
}
@Override
public List<Faculty> showAllFacultiesAndThierStudents() {
return facultyRepo.findAll();
}
@Override
@Transactional(propagation = Propagation.REQUIRED)
public String removeStudent From Faculty(int fid, int sid) {
//Load faculty
Faculty faculty=faculty Repo.findByld (fid).orElse Throw(()-> new IllegalArgumentException("Invalid Id"));
//Load Student
Student st=studRepo.findByld(sid).orElse Throw(()->new IllegalArgumentException("Invalid Id"));
//get all the students of the faculty
Set<Student> childs=faculty.getStudents();
//remove student from the existing students
childs.remove(st);
//remove faculty from the list of faculty beloging to parent
Set<Faculty> parents=st.getFaculties();
parents.remove(faculty);
// update the Faculty object
facultyRepo.save(faculty);
return sid+" Student is removed from "+fid+" Faculty's list of students";
}
@Override
@Transactional
```

```
public String remove FacultyFromStudent(int sid, int fid) {
//Load faculty
Faculty faculty=facultyRepo.findById(fid).orElse Throw(()-> new IllegalArgumentException("Invalid Id"));
//Load Student
Student st=studRepo.findById (sid).orElse Throw(()->new IllegalArgumentException("Invalid Id"));
//get all the students of the faculty
Set<Student> childs=faculty.getStudents();
//remove student from the existing students
childs.remove(st);
//remove faculty from the list of faculty beloging to parent
Set<Faculty> parents-st.getFaculties();
parents.remove(faculty);
//update the Student obj
studRepo.save(st);
return fid+" faculty is removed for "+sid+" student";
//Runner class
==============
package com.nt.runners;
import java.util.List;
import java.util.Set;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;
import com.nt.entity.Faculty;
import com.nt.entity.Student;
import com.nt.service.ICollegeMgmtService;
@Component
public class ManyToManyAssociation TestRunner implements CommandLineRunner {
@Autowired
private ICollegeMgmtService collegeService;
@Override
public void run(String... args) throws Exception {
/*try {
//prepare parent objs
Student stud1=new Student("raja","hyd");
Student stud2=new Student("mahesh","vizag");
Student stud3=new Student("ramesh","delhi");
```

```
Faculty faculty1=new Faculty("karan", "CS", "M.Tech");
Faculty faculty2=new Faculty("mahesh","Chemistry", "M.sc");
// assign students faculties
faculty1.getStudents().add(stud1);
faculty1.getStudents().add(stud2);
faculty1.getStudents().add(stud3);
faculty2.getStudents().add(stud2);
faculty2.getStudents().add(stud3);
//assign faculties to Studnets
stud1.getFaculties().add(faculty1);
stud2.getFaculties().add(faculty1);
stud2.getFaculties().add(faculty2);
stud3.getFaculties().add(faculty1);
stud3.getFaculties().add(faculty2);
// call the service class method
List<Student> list-Arrays.asList(stud1,stud2,stud3);
String msg=collegeService.registerStudent(list);
System.out.println(msg);
}//try
catch(Exception e) {
e.printStackTrace();
}
*/
/*try {
//prepare parent objs
Student stud1=new Student("raja1","hyd"); Student stud2=new Student("mahesh1","vizag"); Student
stud3=new Student("ramesh1","delhi"); Faculty faculty1=new Faculty("karan1", "CS", "M.Tech");
Faculty faculty2=new Faculty("mahesh1","Chemistry","M.sc");
// assign students faculties
faculty1.getStudents().add(stud1);
faculty1.getStudents().add(stud2);
faculty1.getStudents().add(stud3);
faculty2.getStudents().add(stud2);
faculty2.getStudents().add(stud3);
//assign faculties to Studnets
stud1.getFaculties().add(faculty1);
stud2.getFaculties().add(faculty1);
```

```
stud2.getFaculties().add(faculty2);
stud3.getFaculties().add(faculty1);
stud3.getFaculties().add(faculty2);
// call the service class method
List<Faculty> list-Arrays.asList(faculty1, faculty2);
String msg=collegeService.registerFacultiesAndTheir Students(list);
System.out.println(msg);
}
catch (Exception e) {
e.printStackTrace();
}
*/
/* try {
List<Student> list-collegeService.showAllStudentsAndTheirFaculties();
list.forEach(st->{
System.out.println("Parent ::"+st);
Set<Faculty> childs=st.getFaculties();
childs.forEach(fc->{
System.out.println("Child::"+fc);
System.out.println("==================");
});
});
}
catch(Exception e) {
e.printStackTrace();
}
*/
/*try {
List<Faculty> list=collegeService.showAllFacultiesAndThierStudents();
list.forEach(fc->{
System.out.println("Child ::"+fc);
Set<Student> parents=fc.getStudents();
parents.forEach(st->{
System.out.println("parent::"+st);
System.out.println("=================");
});
});
```

```
}
catch(Exception e) {
e.printStackTrace();
}*/
/* try {
String msg=collegeService.removeStudentFromFaculty(62,161);
System.out.println(msg);
}
catch (Exception e) {
e.printStackTrace();
}*/
try {
String msg=collegeService.remove FacultyFromStudent(163, 63);
System.out.println(msg);
}
catch(Exception e) {
e.printStackTrace();
}
}//method
}//class
```

=> The process of combining related operations into single unit and executing them by applying

do every thing or nothing principle is called Transaction Managemement

=> if the service class b.method is performing one or more non-select operations then it is recommended to enable Transaction management on the top of the b.method .. For this we need to place @Tranactional on top of B,method in service class

=> Instead of placing @Transactional on the top of multiple b,methods, we can place directly on @Transactional

on top of service class then all the b.methods of that class execute having Tx mgmt support

=> Since Select persistence operations does not modify db table data.. So we no need to apply @Tranactional on the

B.method that deals with select operations

Example code

===

```
@Transactional
public String removeFacultyFromStudent(int sid, int fid) {
//Load faculty
Faculty faculty=facultyRepo.findById(fid).orElseThrow(()-> new IllegalArgumentException("Invalid Id"));
//Load Student
Student st=studRepo.findById (sid).orElseThrow(()->new IllegalArgumentException("Invalid Id"));
//get all the students of the faculty
Set<Student> childs=faculty.getStudents();
//remove student from the existing students
childs.remove(st);
//remove faculty from the list of faculty beloging to parent
Set<Faculty> parents=st.getFaculties();
parents.remove(faculty);
//update the Student obj
studRepo.save(st);
return fid+" faculty is removed for "+sid+" student";
}
```