Hibernate Advanced Mapping

In DB

1 Multiple Tables

2 Relationship btw tables

Need to model this with Hibernate

Types

1 one to one

Ex : An instructor can have instructor details , Similar instrctur profile

Instructor(T)-> Instructor Details(T)

2 one to many , many to one

Ex: One to many

instructor can have many courses inverse of this is many to one relation many courses can have single Instrctor

3 many to many

Ex : A course an have many student and student can have many courses

Database concepts

1 primary key and Foreign key

-primary key : identify unique row in a table

-Foreign key : Link table together -> a field in one table refer to primary key in another table

2 cascade

Apply same operation to related entites

If I save Instrctor it will cascade and save instructor\_details also

If we save Instructor it performs same operation to Instructor\_details

If we delete Instructor should delete their instructor details also bcz they no longer have record known as “**CASCADE DELETE**”

We have to be carefull with cascade delete in terms Many to many we should not delete check use case

Fetch Types: Eager VS Lazy Loading => “should we retrieve everything”

Eager will retrieve everything

Lazy will retrieve on request

Uni – Directional Relationship

One way relation using Instrctor get instructor details

Instructor-> Instrctor\_details

Bi –directional

Both ways

Instructor<--> Instrctor\_details

ONE TO ONE

1st uni – directional example

Deveolpment process one to one

1 Define database tables

2 create instructor\_details class

3 create instructor class

4 create main App

Table : instructor\_details

create table instructor\_detail(id serial PRIMARY KEY , youtube\_channel VARCHAR(200) DEFAULT NULL,hobby VARCHAR(100) DEFAULT NULL);

Table : instructor

CREATE TABLE instructor (id serial PRIMARY KEY,first\_name varchar(45) DEFAULT NULL,last\_name varchar(45) DEFAULT NULL,email varchar(45) DEFAULT NULL,

instructor\_detail\_id int DEFAULT null, CONSTRAINT fk\_detail FOREIGN KEY (instructor\_detail\_id) REFERENCES instructor\_detail(id)

)

Need to set relation to the table instructor\_detail\_id

Link the tbale using FK

Definf FK

CONSTRAINT fk\_detail FOREIGN KEY (instructor\_detail\_id) REFERENCES instructor\_detail(id)

Forieng key

Preserve relatiosnship btween tbles

Referential Integrity

Prevents the operations that would destroy relatiosnhship

Ensuer only valid data is inserted into the foreign key column

Can contain only valid referenc to the primary key anoher table

create instructor\_details class

create instructor class

Need to link this class using @OnetoOne it allows to map instrutor\_details to Instructor class together

@OneToOne

@JoinColumn(name="instructor\_detail\_id")

**private** InstructorDetail instructorDetail;

For instructor\_detail we have join column, instructor\_detail\_id which is in instrutor table, hibernate use the foreign key to find instructor details record and laod data

Entity Life Cycle

States hibernate can go thorugh in your apps

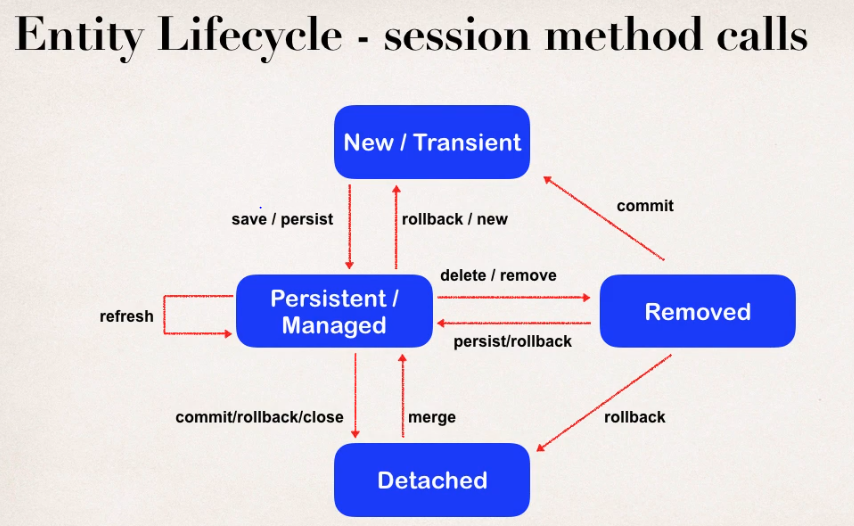
Detach: if entity is detached it is not associated with hbiernate session

Merge: if instance is detached from session then merge will reatttach to session

Persist: Transitions new instance to managed state . Next flush / commit will save to db

Remove : Transitionsmanaged entity to be removed Next flush / commit will delete from db

Refresh : Reload / sync object with data from DB prevents stale data



@onetoone

Cascade Types

PERSIST : if entity is persisted/saved entity will also persisted

REMOVE: if entity is removed /deleted related entity will also be deleted

REFRESH: if entity is refreshed related entity is also refreshed

DETACH : if entity is detached(not associated with hbiernate session) then related entity is also detached

MERGE: if Entity is merged then related entity is aslo be merged

ALL : all above cascade types

Config Cascade

@OneToOne(cascade=cascadeType.ALL) means all operation we apply to our entity instructor it also cascade to related entity instructor details

so that applies for save delete operations

By default no operations are cascaded, if we dnt specify cascade then none of th operation will cascade we have explicitly reference given cascade type

Config mulptile CAscade type

In case I dnt want give all but 2 or more cascade types

@OneToOne(cascade={cascadeType.DETACH,cascadeType.MERGE,cascadeType.PERSIST})

4 Create Main App

1 create 2 objects withintialized data

2 assosiate the objects by using setterMethod (now these objects are connected in memory and they are associated based on setter method)

setup relationship for mapping between instrutor and instructor details

Followed begin transcton, save , commit both will be saved

Create Tables using scripts

4 Creating Main APP Save

**public** **class** CreateDemoOnetoOne {

**public** **static** **void** main(String[] args) {

//create Session Factory

SessionFactory factory=**new** Configuration().configure("hibernate.cfg.xml")

.addAnnotatedClass(Instructor.**class**)

.addAnnotatedClass(InstructorDetail.**class**)

.buildSessionFactory();

//create session

Session session=factory.getCurrentSession();

**try**

{

// create object

Instructor instructor=**new** Instructor("sachin", "gowda", "sachin.hs@bcits.in");

InstructorDetail instructorDetail=**new** InstructorDetail("www.youtube.com", "riding");

//assocaite objects now these two enetity are assoicated in memory still changes reuired

instructor.setInstructorDetail(instructorDetail);

// begin Transaction

session.beginTransaction();

// save transition

System.***err***.println("Saving Instructor: "+instructor);

session.save(instructor); // Note : it will also saves the detail oject bcz of CascadeType.ALL

// commit transaction

session.getTransaction().commit();

System.***err***.println("Done");

}**catch** (Exception e) {

e.printStackTrace();

System.***err***.println(e.getMessage());

}**finally** {

factory.close();

}

}

4 Creating Main APP Delete

When we delete Instructor it cascade all opertaion and delete its related instructor details also

Currently compplete uni directionl

Instrutor---> instructordetails

OneToOne Bi-directional

New use case

If we load InstructorDetails then wed like get assicated Instructor

Not possible in Uni Directional

We can slove it by using Bi directional

Instrutor<---> instructordetails

-No change in db setup

- update java code

Development process OneToOne( Bi-Directional)

1 Make update to InstructionDetails Class

A Add new field to reference Instructor

B add Getter/setter method for Instructor

C Add OneToOne annotation

2 Create Main App

1 @OneToOne(mappedBy="instructorDetail",cascade=CascadeType.***ALL***) // instructorDetail refers to "instructorDetail" property in Instrutor Class

**private** Instructor instructor; // Adding New Field to make Bi- directionl also add getter and Setter

//instructor field is mapped by "instructorDetail" property in Instrutor Class

mappedBy : tells hibernate look at the instructionDetail proprty in the Instruction class

Hibernate use the information form Instructor class @JoinColumn

To help assoicated instructor

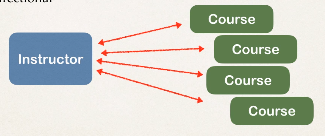
Add support For cascading

Refactoring and Exception Handling

OneToMany / ManyToOne mapping

-an isntructor can hvae many course

-Bi-directional



-many course can have one instructor its inverse/oppsite of oneToMany

Real -world project

- if you delete instructor dnt delete the course

- if you delete course don’t delete instructor

So NO CASCADE DELETE

Deveoplment Process

1 Define Tables

2 Course class creation

3 Update Instruction class

4 create mainAPP

1 define Table

create table course(id serial PRIMARY KEY,title VARCHAR(200) ,instructor\_id int , CONSTRAINT FK\_INSTRUCTOR\_DETIALS FOREIGN KEY (instructor\_id) REFERENCES instructor(id));

Add Unique key to title to aovid avoid dulicate

2 create class

//define field

//define Constructor

// define getter / setter

// define toString

// annotate fields

@Id

@GeneratedValue(strategy=GenerationType.***IDENTITY***)

@Column(name="id")

**private** **int** id;

@Column(name="title")

**private** String title;

//ManyToOne many course can have one instructor

// join column instructor\_id of course table maps to Instructor Table id

// add cascade NO CASCADE DELETE

@ManyToOne(cascade= {CascadeType.***DETACH***,CascadeType.***MERGE***,CascadeType.***PERSIST***,CascadeType.***REFRESH***})

@JoinColumn(name="instructor\_id")

**private** Instructor instructor;

3 Update Instructor Reference class

Mapped By

In Instructor entity look into instructor property in Course class use cource column from @joinColumn

Help to find assoicated courses for instructor

Add supprting CASCADE except REMOVE so no CASCADE DELETE in both classes

Add convinecen methhod for bi-driectional

4 Create MainAPP

Fetch Types

When we load data should we retrive everything

1 Eager will retrive everything

- it will load all dependententites ex: load Instructor and all of their cources at once

- one shot to db bring all data and its assoicated data

-its k for small no of instructor and Coursce if its big impact performance of applciation

Easily could turn into performnace nightmare

All of the studnets for the cource will slow applciation and its performance

-in our app if we are searching for a cource by keyword

- only want a list of matching cources

-Eager wuld still load all the studnets for each cource which is not good

2 lazy will retrive on request

- it will load main entity first

- load dependent entites on demand(lazy)

Load Cource 1st then when u need student then u load it on demand

Lazy lodaing is prefered

Best practice : only load data when absolutly needed prefer lazy Loading insted of Eager loading

Real Time

Instructor details if want see that instructor go to its details load its assoaicted class

Real Time use case

- in mastee view, use lazy loading

- in detail view, retrive the entity and nessary dependent entites

Real Time use case -MASTER VIEW

- in master view use lazy loading for search case

- only load the instructor not their Cources

Real Time use case -Detail VIEW

- in detail view , retrive entity and its nesscry dependent entity

-Load instructor and thier Cources

Fetch Type

When we define mapping relationship we can specify fetch type Eager / lazy

In Intstructor Class

Where we call course there we can aply with mapping relation



Override Default Fetch type

Specifying the fetch type , overrides the defaults

More on Lazy Loading

- when you lazy load, data is only retrived on demand

- require open hibernate session

-need an connection to retrive data from db

- if hbiernate session session I sclosed and trying to retrive lazy data hibernate will throw exception

Retrtive lazt data using

Option 1 : session.get call appricate getter method

Option 2: HQL

Test Lazy Loading

1 lets break it on purpose close session