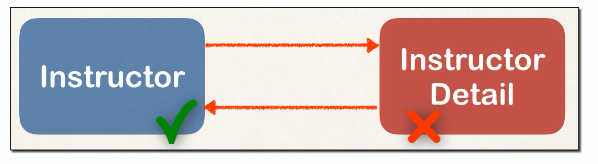
**23.6. OneToOne - Bi-Directional - Delete Only InstructorDetail**

**Delete Only InstructorDetail**:

Now we want to delete the InstructorDetail but keep the Instructor in the database.



1. Modify the cascadeType in the InstructorDetail entity.
2. Break bi-directional link in MainApp

**1) Modify the cascadeType in the InstructorDetail entity**:

Remove CascadeType.***ALL*** and add

CascadeType.***DETACH***,

CascadeType.***MERGE***,

CascadeType.***PERSIST***,

CascadeType.***REFRESH***

**File: InstructorDetail.java**:

**package** com.ruhul.odduu.hibernate.entity;

**import** javax.persistence.\*;

//annotate the class class as an entity and map to BD table

@Entity

@Table(name = "instructor\_detail")

**public** **class** InstructorDetail {

// define the fields

// annotate the field with BD column name

...

// add new field for instructor and add getter and setter method

// add OneToOne annotation

@OneToOne(mappedBy = "instructorDetail", cascade = {

CascadeType.***DETACH***, CascadeType.***MERGE***,

CascadeType.***PERSIST***, CascadeType.***REFRESH***})

**private** Instructor instructor;

// create constructors

// generate getter and setter methods

// generate toString() method

}

}

**2) Break bi-directional link in MainApp (DeleteInstructorDetailDemo.java)**:

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.cfg.Configuration;

**import** com.ruhul.odduu.hibernate.entity.Instructor;

**import** com.ruhul.odduu.hibernate.entity.InstructorDetail;

**public** **class** DeleteInstructorDetailDemo {

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

// create session factory

SessionFactory factory = **new** Configuration()

.configure("hibernate.cfg.xml")

.addAnnotatedClass(InstructorDetail.**class**)

.addAnnotatedClass(Instructor.**class**)

.buildSessionFactory();

// create session

Session session = factory.getCurrentSession();

**try** {

// start transaction

session.beginTransaction();

// get the InstructorDetail object

**int** theId = 7;

InstructorDetail tempInstructorDetail = session

.get(InstructorDetail.**class**, theId);

// print the InstructorDetail

System.***out***.println("tempInstructorDetail: "

+ tempInstructorDetail);

// print the associated instructor

System.***out***.println("The associated Instructor: "

+ tempInstructorDetail.getInstructor());

//now let delete the instructorDetail

System.***out***.println("Deleting tempInstructorDetail: "

+tempInstructorDetail);

session.delete(tempInstructorDetail);

// commit the transaction

session.getTransaction().commit();

System.***out***.println("Done!!!");

} **catch** (Exception exc) {

exc.printStackTrace();

} **finally** {

//handle connection leak issue

session.close();

factory.close();

}

}

}

Now if we run the above code we will get an error saying

javax.persistence.EntityNotFoundException: deleted object would be re-saved by cascade (remove deleted object from associations): [com.ruhul.odduu.hibernate.entity.InstructorDetail#7]

at org.hibernate.internal.ExceptionConverterImpl.convert(ExceptionConverterImpl.java:133)

at org.hibernate.internal.ExceptionConverterImpl.convert(ExceptionConverterImpl.java:181)

at org.hibernate.internal.ExceptionConverterImpl.convert(ExceptionConverterImpl.java:188)

at org.hibernate.internal.SessionImpl.doFlush(SessionImpl.java:1490)

at org.hibernate.internal.SessionImpl.managedFlush(SessionImpl.java:515)

at org.hibernate.internal.SessionImpl.flushBeforeTransactionCompletion(SessionImpl.java:3355)

at org.hibernate.internal.SessionImpl.beforeTransactionCompletion(SessionImpl.java:2526)

at org.hibernate.engine.jdbc.internal.JdbcCoordinatorImpl.beforeTransactionCompletion(JdbcCoordinatorImpl.java:473)

at org.hibernate.resource.transaction.backend.jdbc.internal.JdbcResourceLocalTransactionCoordinatorImpl.beforeCompletionCallback(JdbcResourceLocalTransactionCoordinatorImpl.java:178)

at org.hibernate.resource.transaction.backend.jdbc.internal.JdbcResourceLocalTransactionCoordinatorImpl.access$300(JdbcResourceLocalTransactionCoordinatorImpl.java:39)

at org.hibernate.resource.transaction.backend.jdbc.internal.JdbcResourceLocalTransactionCoordinatorImpl$TransactionDriverControlImpl.commit(JdbcResourceLocalTransactionCoordinatorImpl.java:271)

at org.hibernate.engine.transaction.internal.TransactionImpl.commit(TransactionImpl.java:104)

at com.ruhul.odduu.hibernate.demo.DeleteInstructorDetailOnlyDemo.main(DeleteInstructorDetailOnlyDemo.java:48)

Apr 16, 2019 12:11:23 AM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PoolState stop

INFO: HHH10001008: Cleaning up connection pool [jdbc:mysql://localhost:3306/hb-01-one-to-one-uni?useSSL=false]

For solve the problem we have to

1. Remove the associated object reference
2. Break bi-directional link

In the **DeleteInstructorDetailOnlyDemo.class** add the line before delete operation.

//remove the associated object reference

//break bi-directional link

tempInstructorDetail.getInstructor().setInstructorDetail(**null**);

session.delete(tempInstructorDetail);

23.6. OneToOne - Bi-Directional - Delete Only InstructorDetail