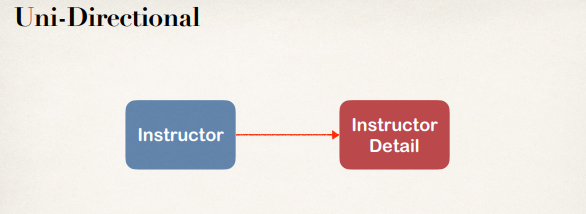
**23.5. One-to-One Bi-Directional**

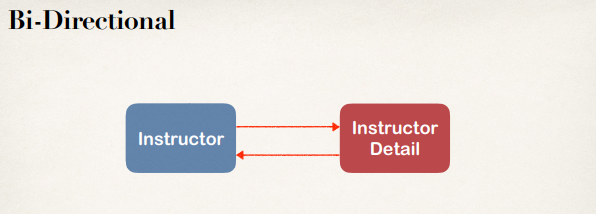
**One-to-One Bi-Directional**:

Here we load an InstructorDetail object and then we would like to get the associated Instructor for that detail object.

We cannot do this by using Uni-directional relationship. In Uni-directional relationship we can only start with Instructor and then move to InstructorDetail. We cannot go the other way. With Uni-directional its just a one-way street.



We can solve the problem by using a Bi-directional relationship. We can start with InstructorDetail and make it back to the Instructor. And since we already have the previous Uni-directional relationship in place, we electively have Bi-directional now. So now this is a two-way street (Uni-directional & Bi-directional). We can start on any side, Instructor -> InstructorDetail or InstructorDetail -> Instructor.



**Note**:

We are using our previous database schema. No change required to our database.

**Database Script**:

DROP SCHEMA IF EXISTS `hb-01-one-to-one-uni`;

CREATE SCHEMA `hb-01-one-to-one-uni`;

use `hb-01-one-to-one-uni`;

SET FOREIGN\_KEY\_CHECKS = 0;

DROP TABLE IF EXISTS `instructor\_detail`;

CREATE TABLE `instructor\_detail` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`youtube\_channel` varchar(128) DEFAULT NULL,

`hobby` varchar(45) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE = InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `instructor`;

CREATE TABLE `instructor` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`first\_name` varchar(45) DEFAULT NULL,

`last\_name` varchar(45) DEFAULT NULL,

`email` varchar(45) DEFAULT NULL,

`instructor\_detail\_id` int(11) DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `FK\_DETAIL\_idx` (`instructor\_detail\_id`),

CONSTRAINT `FK\_DETAIL` FOREIGN KEY (`instructor\_detail\_id`)

REFERENCES `instructor\_detail` (`id`)

ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=latin1;

SET FOREIGN\_KEY\_CHECKS = 1;

**Development Process**:

1. Make updates to InstructorDetail class:
   1. Add new field to reference Instructor
   2. Add getter/setter methods for Instructor
   3. Add @OneToOne annotation
2. Create Main App.

**1.a) Add new field to reference Instructor**:

@Entity@Table(name="instructor\_detail")public class InstructorDetail { ...

private Instructor instructor;

...

}

**1. b) Add getter/setter methods Instructor**:

@Entity@Table(name="instructor\_detail")public class InstructorDetail {

... private Instructor instructor;

public Instructor getInstructor() { return instructor; }

public void setInstructor(Instructor instructor) { this.instructor = instructor; }

...

}

**1.c) Add @OneToOne annotation**:

@Entity

@Table(name="instructor\_detail")

public class InstructorDetail {

...

// mappedBy -> Refers to “instructorDetail” property in “Instructor” class

// Cascade all operations to the associated Instructor @OneToOne(mappedBy="instructorDetail", cascade=CascadeType.ALL) private Instructor instructor;

public Instructor getInstructor() { return instructor; }

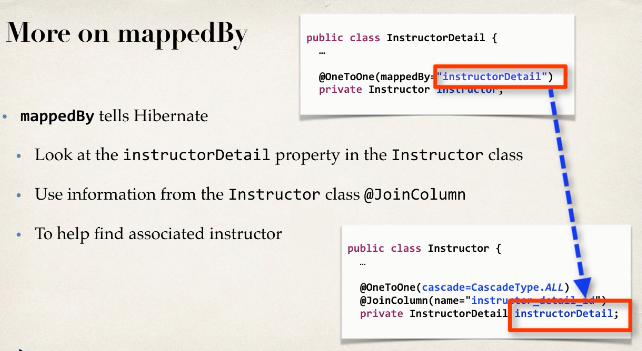
public void setInstructor(Instructor instructor) { this.instructor = instructor; }

**}**

**More on mappedBy**:

mappedBy tells Hibernate

1. Look at the instructorDetail property in the Instructor class
2. Use information from the Instructor class @JoinColumn
3. To help find associated instructor



**2) Create Main App**:

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

**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 = 5;

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();

}

}

}

23.5. One-to-One Bi-Directional