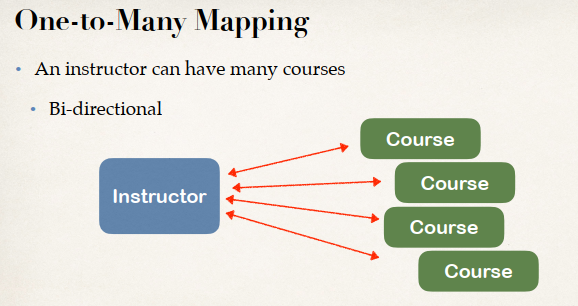
**24.1. OneToMany - Bi-Directional Overview**

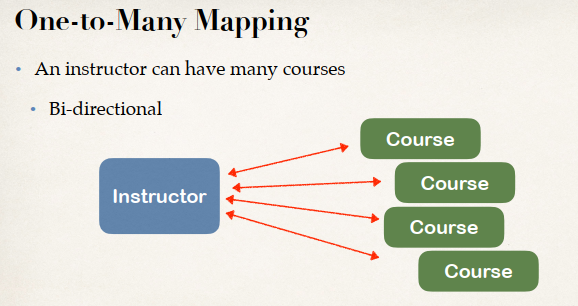
**One-to-Many Mapping**:

* An instructor can have many courses
* Bi-directional



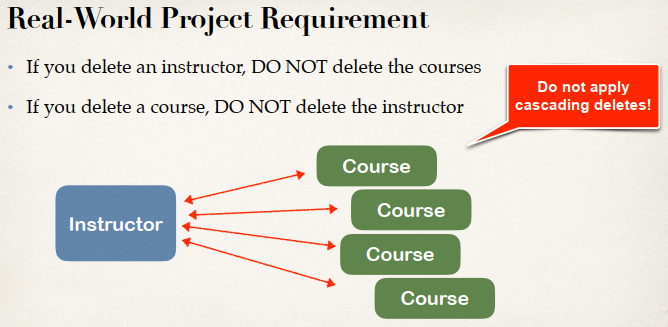
**Many-to-One Mapping**:

* Many courses can have one instructor
* Inverse / opposite of One-to-Many



**Real-World Project Requirement**:

* If we delete an instructor, DO NOT delete the courses
* If we delete a course, DO NOT delete the instructor



This means don’t apply the “cascade delete”

**Development Process: One-to-Many (Step-by-Step)**:

1. Prep Work - Define database tables
2. Create Course class
3. Update Instructor class
4. Create Main App

**1) Prep Work - Define database tables**:

Run the following SQL script from “hibernate-mapping-database-scripts/hb-03-one-to-many”

**Filename: create-db.sql**:

DROP SCHEMA IF EXISTS `hb-03-one-to-many`;

CREATE SCHEMA `hb-03-one-to-many`;

use `hb-03-one-to-many`;

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;

DROP TABLE IF EXISTS `course`;

CREATE TABLE `course` (

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

`title` varchar(128) DEFAULT NULL,

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

PRIMARY KEY (`id`),

UNIQUE KEY `TITLE\_UNIQUE` (`title`),

KEY `FK\_INSTRUCTOR\_idx` (`instructor\_id`),

CONSTRAINT `FK\_INSTRUCTOR`

FOREIGN KEY (`instructor\_id`)

REFERENCES `instructor` (`id`)

ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB AUTO\_INCREMENT=10 DEFAULT CHARSET=latin1;

SET FOREIGN\_KEY\_CHECKS = 1;

Here we are creating three tables.

1. instructor
2. instructor\_detail
3. course

**Table: instructor**:

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;

**Table: instructor\_detail**:

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;

**Table: course**:

CREATE TABLE `course` (

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

`title` varchar(128) DEFAULT NULL,

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

PRIMARY KEY (`id`),

UNIQUE KEY `TITLE\_UNIQUE` (`title`),

KEY `FK\_INSTRUCTOR\_idx` (`instructor\_id`),

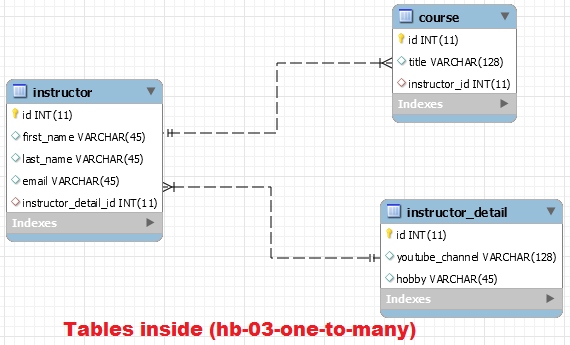
CONSTRAINT `FK\_INSTRUCTOR`

FOREIGN KEY (`instructor\_id`)

REFERENCES `instructor` (`id`)

ON DELETE NO ACTION ON UPDATE NO ACTION

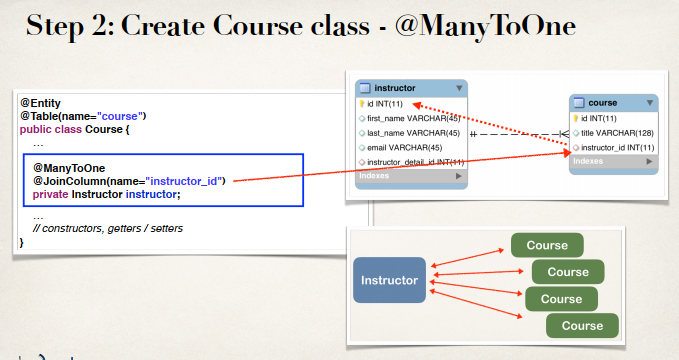
) ENGINE=InnoDB AUTO\_INCREMENT=10 DEFAULT CHARSET=latin1;



**2) Create Course class**:

Instructor can have many Course (Many-to-One)

Join column Instructor(id) -> Course(instructor\_id)



**File: Course.class**:

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

**import** javax.persistence.CascadeType;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.JoinColumn;

**import** javax.persistence.ManyToOne;

**import** javax.persistence.Table;

@Entity

@Table(name = "course")

**public** **class** Course {

// define our fields

@Id

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

@Column(name = "id")

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

@Column(name = "title")

**private** String title;

@ManyToOne(cascade = {

CascadeType.***PERSIST***, CascadeType.***MERGE***,

CascadeType.***DETACH***, CascadeType.***REFRESH*** })

@JoinColumn(name = "instructor\_id")

**private** Instructor instructor;

// define constructor

**public** Course() {

}

**public** Course(String title) {

**this**.title = title;

}

// define getter and setter

// generate toString() method

}

**3) Update Instructor class**:

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

@Entity

@Table(name = "instructor")

**public** **class** Instructor {

// update Instructor for Course

// Refers to “instructor” property in "Course" class

@OneToMany(mappedBy = "instructor", cascade = {

CascadeType.***PERSIST***,CascadeType.***MERGE***,

CascadeType.***DETACH***,CascadeType.***REFRESH*** })

**private** List<Course> courses;

// generate getter and setter methods

**public** List<Course> getCourses() {

**return** courses;

}

**public** **void** setCourses(List<Course> courses) {

**this**.courses = courses;

}

// add convenience methods for bi-directional relationship

**public** **void** add(Course tempCourse) {

**if** (courses == **null**) {

courses = **new** ArrayList<Course>();

}

courses.add(tempCourse);

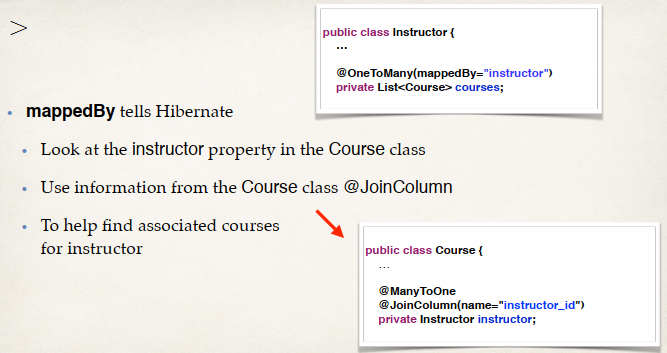
tempCourse.setInstructor(**this**);

}

}

**mappedBy tells Hibernate**:

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



**4) Create Main App**:

1. CreateInstructorDemo.class
2. CreateCoursesDemo.class
3. GetInstructorCoursesDemo.class
4. DeleteCourseDemo.class

**File: CreateInstructorDemo.class**:

**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.Course;

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

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

**public** **class** CreateInstructorDemo {

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

// create session factory

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

.addAnnotatedClass(InstructorDetail.**class**)

.addAnnotatedClass(Course.**class**)

.buildSessionFactory();

// create session

Session session = factory.getCurrentSession();

**try** {

// create the object

Instructor tempInstructor =

**new** Instructor("Ruhul", "Amin", "ruhul@gmail.com");

InstructorDetail tempInstructorDetail = **new** InstructorDetail(

"https://www.codestepbystep.com", "Coding");

// associate the objects

tempInstructor.setInstructorDetail(tempInstructorDetail);

// start transaction

session.beginTransaction();

// save the instructor

// Note: this will ALSO save the tempInstructorDetail object

because of CascadeType.ALL

System.***out***.println("Saving instrucror: "+tempInstructor);

session.save(tempInstructor);

// commit the transaction

session.getTransaction().commit();

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

}

**finally** {

//add clean up code

session.close();

factory.close();

}

}

}

**File CreateCoursesDemo.class**:

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

// create session factory

SessionFactory factory = **new** Configuration()

.configure("hibernate.cfg.xml")

.addAnnotatedClass(Instructor.**class**)

.addAnnotatedClass(InstructorDetail.**class**)

.addAnnotatedClass(Course.**class**)

.buildSessionFactory();

// create session

Session session = factory.getCurrentSession();

**try** {

// start transaction

session.beginTransaction();

// get the instructor from DB

**int** theID = 1;

Instructor tempInstructor = session.get(Instructor.**class**, theID);

// create some courses

Course tempCourse1 = **new** Course(

"Adobe Photoshop - The best design for electrical device ever");

Course tempCourse2 = **new** Course(

"Adobe Illstrutor - The best design for printing media ever");

// add courses to instructor

tempInstructor.add(tempCourse1);

tempInstructor.add(tempCourse2);

// save the courses

session.save(tempCourse1);

session.save(tempCourse2);

// commit the transaction

session.getTransaction().commit();

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

} **finally** {

// add clean up code

session.close();

factory.close();

}

}

**File: GetInstructorCoursesDemo.java**:

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

// create session factory

SessionFactory factory = **new** Configuration()

.configure("hibernate.cfg.xml")

.addAnnotatedClass(Instructor.**class**)

.addAnnotatedClass(InstructorDetail.**class**)

.addAnnotatedClass(Course.**class**)

.buildSessionFactory();

// create session

Session session = factory.getCurrentSession();

**try** {

// start transaction

session.beginTransaction();

// get the instructor from DB

**int** theID = 1;

Instructor tempInstructor = session.get(Instructor.**class**, theID);

System.***out***.println("Instructor: "+tempInstructor);

//get courses for the instructor

System.***out***.println("Courses: "+tempInstructor.getCourses());

// commit the transaction

session.getTransaction().commit();

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

} **finally** {

// add clean up code

session.close();

factory.close();

}

}

}

**File: DeleteCourseDemo.java**:

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

// create session factory

SessionFactory factory = **new** Configuration()

.configure("hibernate.cfg.xml")

.addAnnotatedClass(Instructor.**class**)

.addAnnotatedClass(InstructorDetail.**class**)

.addAnnotatedClass(Course.**class**)

.buildSessionFactory();

// create session

Session session = factory.getCurrentSession();

**try** {

// start transaction

session.beginTransaction();

//get a course

**int** theId = 10;

Course tempCourse = session.get(Course.**class**, theId);

//delete course

System.***out***.println("Deleting course: "+tempCourse);

session.delete(tempCourse);

// commit the transaction

session.getTransaction().commit();

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

} **finally** {

// add clean up code

session.close();

factory.close();

}

}

24.1. OneToMany - Bi-Directional Overview