**26. Hibernate-OneToMany-Unidirectional**

**One-to-Many Mapping**:

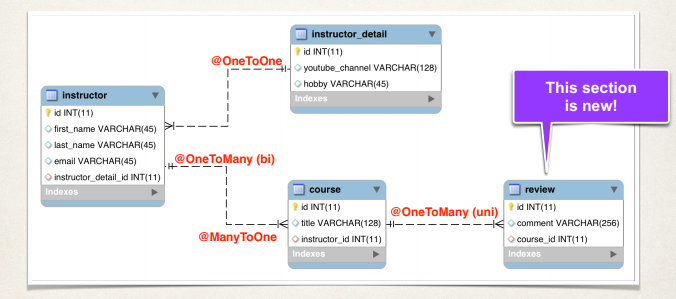
1. A course can have many reviews
2. This is an Uni-directional relationship



**Real-World Project Requirement**:

1. If you delete a course, also delete the reviews
2. Reviews without a course … have no meaning

**Table structure**:



**Development Process**:

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

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

DROP SCHEMA IF EXISTS `hb-04-one-to-many-uni`;

CREATE SCHEMA `hb-04-one-to-many-uni`;

use `hb-04-one-to-many-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;

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;

DROP TABLE IF EXISTS `review`;

CREATE TABLE `review` (

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

`comment` varchar(256) DEFAULT NULL,

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

PRIMARY KEY (`id`),

KEY `FK\_COURSE\_ID\_idx` (`course\_id`),

CONSTRAINT `FK\_COURSE`

FOREIGN KEY (`course\_id`)

REFERENCES `course` (`id`)

ON DELETE NO ACTION ON UPDATE NO ACTION

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

SET FOREIGN\_KEY\_CHECKS = 1;

**2) Create Review class**:

**File: Review.class**:

@Entity

@Table(name = "review")

**public** **class** Review {

// define fields

@Id

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

@Column(name = "id")

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

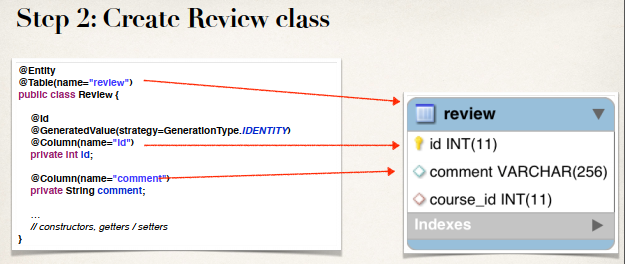
@Column(name = "comment")

**private** String comment;

// define constructors

// define getter/setter methods

}



**3) Update Course class**:

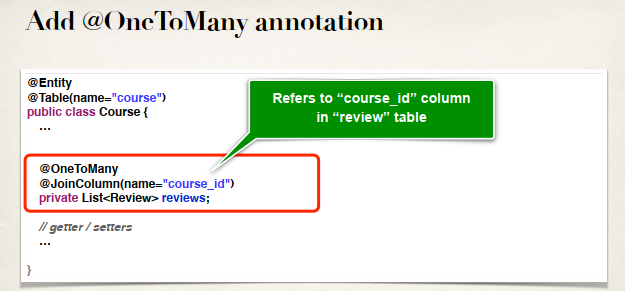
Now we have to update our **Course** class to references the Reviews. The course has a collection of reviews.

@OneToMany(fetch=FetchType.***LAZY***, cascade=CascadeType.***ALL***)

@JoinColumn(name="course\_id")

**private** List<Review> reviews;

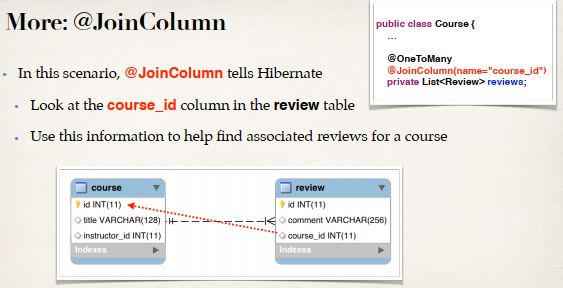
// define getter and setter for reviews



**More: @JoinColumn:**

In this scenario, @JoinColumn tells Hibernate

1. Look at the “**course\_id**” column in the **review** table
2. Use this information to help find associated reviews for a course



**Add support for Cascading and Lazy Loading**:

If we delete the **Course**, we also delete the Reviews. Without Course the Review has no meaning. Here we add “” because we want to cascade all operation.

Now in our Course class we add support for lazy loading, because we want to load Reviews on demand.

@OneToMany(fetch=FetchType.***LAZY***, cascade=CascadeType.***ALL***)

@JoinColumn(name="course\_id")

**private** List<Review> reviews;

**Add convenience method for adding review**:

//add a convenience method

**public** **void** addReview(Review theReview) {

**if**(reviews == **null**) {

reviews = **new** ArrayList<>();

}

reviews.add(theReview);

}

**4) 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.Course;

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

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

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

**public** **class** CreateCoursesAndReviewsDemo {

**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**)

.addAnnotatedClass(Review.**class**)

.buildSessionFactory();

// create session

Session session = factory.getCurrentSession();

**try** {

// start transaction

session.beginTransaction();

//create a course

Course tempCourse = **new**

Course("Oracle Certified Java Programmer - JAVA-8");

//add some reviews

tempCourse.addReview(**new** Review(

"Covere total 23 core concept related to Core-Jave"));

tempCourse.addReview(**new** Review(

"Provide Details Note and Source code for each topics"));

tempCourse.addReview(**new** Review(

"Details explanation with real world example"));

//save the course ... and leverage the cascade all

System.***out***.println("Saving the courses...");

System.***out***.println(tempCourse);

System.***out***.println(tempCourse.getReview());

session.save(tempCourse);

// commit the transaction

session.getTransaction().commit();

System.***out***.println("com.ruhul: Done!!!");

} **finally** {

// add clean up code

session.close();

factory.close();

}

}

}

26. Hibernate-OneToMany-Unidirectional