**Demo Lazy Fetching**

**Demo Lazy Fetching (Step-by-Step)**:

1. Modify the Instructor class
2. Debug Main App for test

**1) Modify the Instructor class**:

We know that Instructor have a list of courses. Set up “FaceType” here for the courses to be Lazy.

@OneToMany(fetch=FetchType.***LAZY***, mappedBy = "instructor",

cascade = { CascadeType.***PERSIST***, CascadeType.***MERGE***,

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

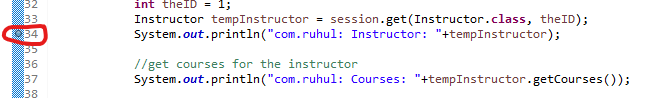
**private** List<Course> courses;

Here the courses will only be loaded on demand.

**2) Debug Main App for test**:

Now for test our main app we set a breakpoint in “**LazyDemo.java**” class.

Right Click on the line -> Toggle Breakpoint



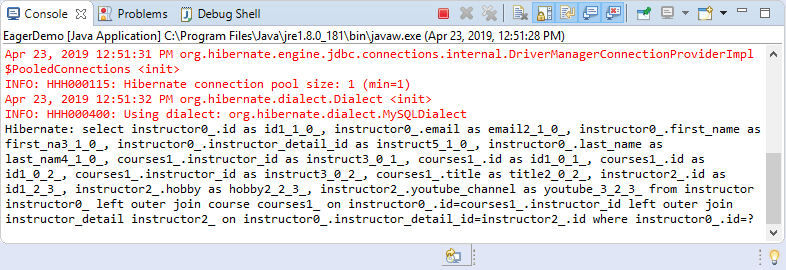
Now we run the application as Debug Mode

Run -> Debug As -> Java Application

Now Eclipse open “Debug Perspective”



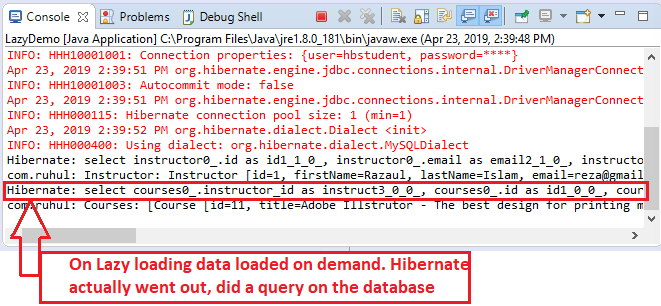
In the console we will see our App load the Instructor and InstructorDetail.



There are not loading the courses yet, because the courses are only loaded on demand. For show more information hit the “Step Over” button.



In Lazy loading App loaded data on demand. Hibernate actually went out, did a query on the database.



In the above picture we will show our application hit the database two times.

**File: EagerDemo.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.eager.Course;

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

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

**public** **class** EagerDemo {

**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("com.ruhul: Instructor: "+tempInstructor);

//get courses for the instructor

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

// commit the transaction

session.getTransaction().commit();

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

} **finally** {

// add clean up code

session.close();

factory.close();

}

}

}

Demo Lazy Fetching