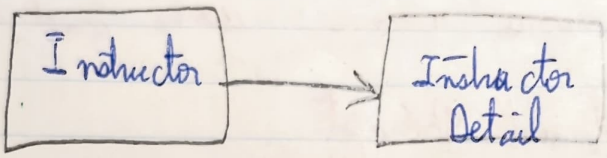
JPA / Hibernate
Advanced Mappings

One-to-One Mapping



FK field in a table. refers to PK of another table

Cascade: - cascade operations
- apply same operation to related entities

Fetch Types:
- Eager: retrieve everything
- Lazy: retrieve on request

```
@Entity
@Table(name="instructor_detail")
public class InstructorDetail { }
```

```
@Entity
@Table(name="instructor")
public class Instructor {
```

nouveau
```
    @OneToOne(cascade=CascadeType.ALL)
    @JoinColumn(name="instructor_detail_id")
    private InstructorDetail instructorDetail;
```

↳ au lieu de "instructor_detail_id", tu mets la classe correspondante

Entity lifecycle

- detach : entity not associated with Hibernate session
- merge : if instance is detached from session, merge will reattach to session
- persist : transitions new instance to managed state. Next flush / commit will save
- remove : transitions managed entity to be removed. Next flush / commit delete db
- refresh : reload / synch object with data from db. Prevents stale data

cascade = CascadeType.ALL
by default, no operation.

* tu peux spécifier liste cascade operation que tu veux.

One-to-One Mapping - Bi-directional

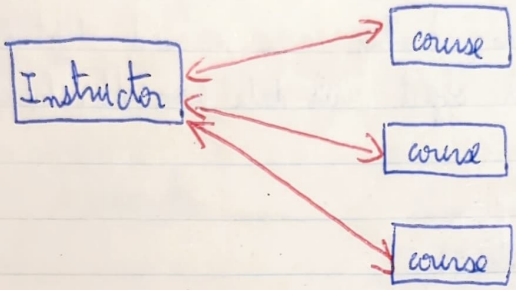
Use case : - Load InstructorDetail then associate with Instructor

- No change on db → update Java Code :
  1. Add field Instructor in InstructorDetail
  2. Getter / Setter for Instructor
  3. Add @OneToOne Annotation on Instructor field

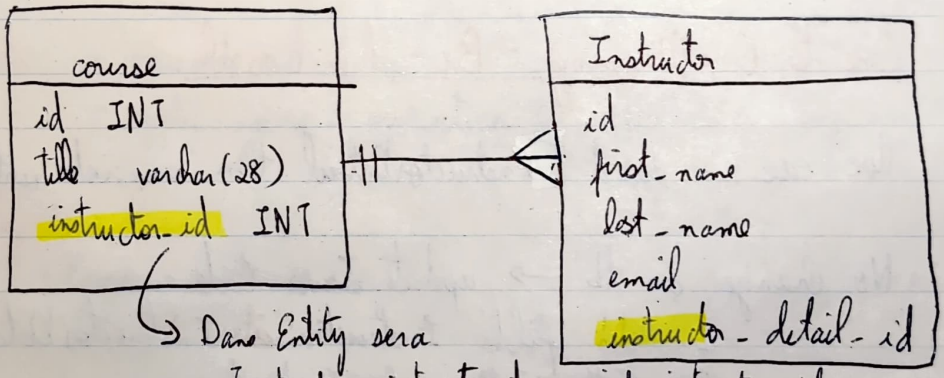
- Mapped by tells Hibernate :
  - Look at InstructorDetail in Instructor
  - Use info from Instructor class @JoinColumn
  - To help find Instructor associated with

One-to-Many

- An instructor can have many courses.



- When delete an instructor, do not delete courses.
- If you delete a course, do not delete instructor



Dans Entity sera
Instructor instructor et non int instructor_id

Dans Course entity :

@ Many To One
@ JoinColumn (name = "instructor_id")
private Instructor instructor

Special → Dans Instructor : @ One To Many (mappedBy = "instructor")
private List<Course> courses ;

non de la variable dans l'autre classe

mappedBy : - Look at Instructor in Course
- Use info from Course class @ JoinColumn
- To help find associated course for instructor.

One-to-Many - Fetch Types : Eager vs Lazy

- Eager : pull everything. Ex : load Instructor with Details
- Lazy : retreive on request
  - load main entity first, load dependent entities on demand

- Fetch Type

```
@OneToMany(fetch = FetchType.Lazy, mappedBy = "instructor")
private List<Course> courses;
```

- Default Fetch Type
  - @OneToOne : FetchType.EAGER
  - @OneToMany : FetchType.LAZY
  - @ManyToOne : FetchType.EAGER
  - @ManyToMany : FetchType.LAZY

⇒ Override - Default Fetch Type

⚠ If Hibernate session is closed and you attempt to retreive data, Hibernate will throw an exception.

Solution : - use FetchType = Eager (easy fix)
- ou bien : • FetchType.Lazy
  • TypedQuery on Course
  • instructor.setCourses (↑)

@OneToMany : Join Fetch Courses
Better Solution than previous : use Join Fetch
We want : - to get Instructor and Courses in a single query
- keep LAZY option available

```
TypedQuery<Instructor> query = entityManager.createQuery( ... )
```

JOIN FETCH is similar to EAGER loading

JOIN FETCH :

- ChatGPT
- load an Author and their books in single query

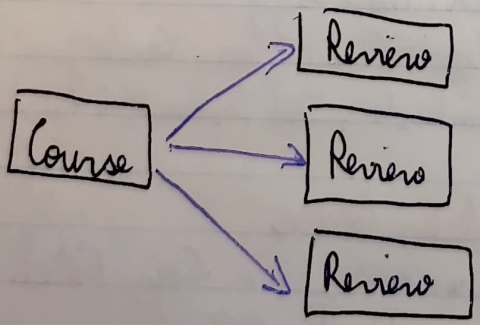
```
SELECT a
FROM Author a
JOIN FETCH a.books
WHERE a.id = :authorId
```

It's equivalent to :

```
SELECT a.*, b.*
FROM Author a
JOIN Book b ON a.id = b.author_id
WHERE a.id = ?
```

One To Many : Update Instructor
: Update Course
: Delete Instructor
: Delete Course

One To Many : Uni-directional



- Delete a course, delete a review
- Casca de delete

Many - to - Many Mapping

- Course can have many students.
- A Student can have many courses.

JOIN TABLE : A table that provides mapping between two tables

Dans class Student :

- @ Many To Many
  @ Join Table (
    name = "course - student"
    joinColumns = @JoinColumn(name = "student_id"),
    inverseJoinColumns = @JoinColumn(name = "course_id")
)
  private List<Course> courses;

define on Course and Students

- Do not cascade delete