

Les associations entre entités avec JPA

Contrôles des relations entre entités avec JPA.

Emmanuel BRUNO

Université de Toulon, LIS UMR CNRS 7020

emmanuel.bruno@univ-tln.fr

2026-10-01

Table des matières

1 One To One (1-1)	1
2 One to Many (1-N)	3
2.1 Avec deux entités	4
3 Many to Many (N-M)	5

JPA offre une prise en charge complète pour la gestion des relations entre entités de différents types, que ce soit en mode un-vers-un (OneToOne), un-vers-plusieurs (OneToMany), ou plusieurs-vers-plusieurs (ManyToMany). Ces relations peuvent être configurées de manière unidirectionnelle ou bidirectionnelle.

1 One To One (1-1)

Dans certaines situations, lors de la modélisation orientée objet, il est nécessaire d'associer deux classes dans une relation un-à-un, mais en gardant une seule relation.

```
java.io.IOException: Cannot run
program "/opt/local/bin/dot": Exec
failed, error: 2 (No such file or
directory)
    at java.base/java.lang.ProcessBuilder.start(ProcessBuilder.java:1048)
    at java.base/java.lang.ProcessBuilder.start(ProcessBuilder.java:1048)
    at net.sourceforge.plantuml.dot.ProcessBuilder.start(ProcessBuilder.java:1048)
    at net.sourceforge.plantuml.dot.ProcessBuilder.start(ProcessBuilder.java:1048)
    at net.sourceforge.plantuml.dot.GraphvizVersionFinder.finder(java:126)
```

```
at net.sourceforge.plantuml.dot.GraphvizVersionFinder.finder(java:126)
at net.sourceforge.plantuml.dot.GraphvizRuntimeEnvironment.finder(java:126)
at net.sourceforge.plantuml.svek.DotStringFactory.generateDiagram(DotStringFactory.java:126)
at net.sourceforge.plantuml.svek.DotStringFactory.generateDiagram(DotStringFactory.java:126)
at net.sourceforge.plantuml.svek.GraphvizImageBuilder.generateImage(GraphvizImageBuilder.java:126)
at net.sourceforge.plantuml.svek.GraphvizImageBuilder.generateImage(GraphvizImageBuilder.java:126)
at net.sourceforge.plantuml.svek.GraphvizImageBuilder.generateImage(GraphvizImageBuilder.java:126)
at net.sourceforge.plantuml.svek.CucaDiagramFileMaker.generateDiagram(CucaDiagramFileMaker.java:126)
at net.sourceforge.plantuml.svek.CucaDiagramFileMaker.generateDiagram(CucaDiagramFileMaker.java:126)
at net.atmp.CucaDiagram.exportDiagramInternal(CucaDiagram.java:126)
at net.sourceforge.plantuml.classdiagram.ClassDiagram.exportDiagram(ClassDiagram.java:126)
at net.sourceforge.plantuml.UmlDiagram.exportDiagram(UmlDiagram.java:126)
at net.sourceforge.plantuml.AbstractPSystem.exportDiagram(AbstractPSystem.java:126)
at net.sourceforge.plantuml.SourceStringReader.outputDiagram(SourceStringReader.java:126)
at net.sourceforge.plantuml.SourceStringReader.outputDiagram(SourceStringReader.java:126)
at io.github.spencerpark.ijava.magics.JavaPlantUMLMagics.generateDiagram(JavaPlantUMLMagics.java:126)
at java.base/jdk.internal.reflect.DirectMethodHandleAccessor.invoke(DirectMethodHandleAccessor.java:126)
at java.base/java.lang.reflect.Method.invoke(Method.java:126)
at io.github.spencerpark.jupyter.kernel.magic.registerMagicFunction(registerMagicFunction.java:126)
at io.github.spencerpark.jupyter.kernel.magic.registerMagicFunction(registerMagicFunction.java:126)
at io.github.spencerpark.jupyter.kernel.magic.registerMagicFunction(registerMagicFunction.java:126)
at REPL.$JShell$61.do_it($JShell$61.java:65)
at java.base/jdk.internal.reflect.DirectMethodHandleAccessor.invoke(DirectMethodHandleAccessor.java:126)
at java.base/java.lang.reflect.Method.invoke(Method.java:126)
at io.github.spencerpark.ijava.execution.IJavaExecution.execute(IJavaExecution.java:126)
at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:126)
at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:126)
at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:126)
at java.base/java.lang.Thread.run(Thread.java:1474)
```

```
Caused by: java.io.IOException: Exec
failed, error: 2 (No such file or
directory)
    at java.base/java.lang.ProcessImpl.forkAndExec(Native
Method)
    at java.base/java.lang.ProcessImpl.<init>(...)
    at java.base/java.lang.ProcessImpl.start(...)
    at java.base/java.lang.ProcessBuilder.start(...)
    ... 34 more
```

Dot Executable: /opt/local/bin/dot
Dot executable does not exist
Cannot find Graphviz. You should try

```
@startuml
testdot
@enduml
```

or

```
java -jar plantuml.jar -testdot
```

Figure 1. – Un Customer a une Biography

```
No JDBC connection available.
Set system properties
'jdbc.url' (and optionally
'jdbc.user'/'jdbc.password'), or
provide a Connection in the kernel
environment.
```

Figure 2

Pour cela JPA propose des annotations spécifiques :

@Embeddable: Cette annotation indique que les membres d'une classe Java peuvent être persistés en tant qu'attributs de la classe qui les utilise. En d'autres termes, elle spécifie qu'une classe peut être utilisée comme composant réutilisable dont les propriétés peuvent être incorporées dans une autre entité.

@Embedded: Utilisée au niveau d'un membre de classe annoté avec **@Embeddable**, cette an-

notation indique que le membre annoté doit être intégré dans la relation. Cela signifie que les propriétés de la classe annotée avec **@Embeddable** seront incluses dans la table de la classe qui les utilise.

```
@Builder
@NoArgsConstructor(access =
AccessLevel.PROTECTED)
@AllArgsConstructor

@Embeddable
public class Biography {
@Column(name = "BIO_BRIEF")
private String brief;

@Lob
@Column(name = "BIO_EXTENDED")
private String extended;
}
```

Figure 3. – Embeddable.java

```
@Getter
@Setter
@ToString
@RequiredArgsConstructor(staticName =
"of")
@NoArgsConstructor(access =
AccessLevel.PROTECTED)

@Entity
@Table(name = "CUSTOMER", schema =
"ex_biography")
public class Customer {
@Id
@GeneratedValue
private long id;

@Column(length = 50, nullable
= false)
@NotNull
private String name;
```

```

    @Embedded
    @Column()
    private Biography biography;
}

```

Customer.java

On peut donc rendre persistente une instance de Customer associé à une instance de Biography.

```

import fr.univtlb.bruno.demos.jpa.hello.samples.ex_biography.Customer;
try (EntityManager entityManager =
    emf.createEntityManager()) {
    entityManager.getTransaction().begin();

        Customer    customer    =
Customer.of("Jim");
    customer.setBiography(Biography.builder
        .brief("bla")
        .extended("bla bla")
        .build());
    entityManager.persist(customer);

    entityManager.getTransaction().commit()
}

```

dans une seule relation

```

No JDBC connection available.
Set system properties
'jdbc.url' (and optionally
'jdbc.user'/'jdbc.password'), or
provide a Connection in the kernel
environment.

```

Figure 4

2 One to Many (1-N)

PlantUML 1.2026.0

This version of PlantUML is 264 days old, so you should consider upgrading from <https://plantuml.com/download>

[From string (line 6)]

```

@startuml
class Order {
    - id: long
    - date: LocalDateTime
}
class Line {
Syntax Error? (Assumed diagram type: class)

```

Figure 5. – Une Order a plusieurs Line

```

@Getter
@Setter
@ToString
@AllArgsConstructor(staticName =
    "of")
@NoArgsConstructor(access =
    AccessLevel.PROTECTED)

@Embeddable
@Table(name = "LINE", schema =
    "EX_ONE_TO_MANY_B")
public class Line {
    @NotNull
    private String product;

    private double price;
}

```

```

@Getter
@Setter
@ToString
@NoArgsConstructor
@Builder
@AllArgsConstructor

@Entity
@Table(name = "EORDER", schema =
    "EX_ONE_TO_MANY_B")
public class Order {
    @Id
    @GeneratedValue(strategy =

```

```

GenerationType.SEQUENCE)
    private long id;

    @Builder.Default
    @Column(nullable = false)
    private LocalDateTime date =
        LocalDateTime.now();

    @ElementCollection
    @CollectionTable(name = "LINE",
        schema = "EX_ONE_TO_MANY_B")
    @Singular
    private Set<Line> lines = new
        HashSet<>();
}

```

No JDBC connection available.
Set system properties
'jdbc.url' (and optionally
'jdbc.user'/'jdbc.password'), or
provide a Connection in the kernel
environment.

Figure 6

2.1 Avec deux entités

Dans le cas général, l'attribut de type `Collection<T>` est annoté avec `@OneToMany`.

L'attribut optionnel `mappedBy` indique le nom de l'attribut correspondant dans le type `T` si l'on souhaite une relation bi-directionnelle. L'autre entité (de Type `T`) est alors annotée avec `@ManyToOne`

```

@Getter
@Setter
@ToString
@NoArgsConstructor

@Entity
@Table(name = "EORDER", schema =

```

```

"EX_ONE_TO_MANY_A")
public class Order {
    @Id
    @GeneratedValue(strategy =
        GenerationType.SEQUENCE)
    private long id;

    private LocalDateTime date =
        LocalDateTime.now();

    @OneToMany(mappedBy = "order",
        cascade =
        {CascadeType.PERSIST,
            CascadeType.REMOVE},
        orphanRemoval = true)
    @ToString.Exclude
    private Set<Line> lines = new
        HashSet<>();

    public Order addLine(Line line) {
        line.setOrder(this);
        lines.add(line);
        return this;
    }

    public Order removeLine(Line line)
    {
        line.setOrder(null);
        lines.remove(line);
        return this;
    }
}

```

```

@Getter
@Setter
@ToString
@RequiredArgsConstructor(staticName =
    "of")
@NoArgsConstructor(access =
    AccessLevel.PROTECTED)

@Entity

```

```

@Table(name = "LINE", schema =
"EX_ONE_TO_MANY_A")
public class Line {
    @Id
    @GeneratedValue(strategy =
 GenerationType.SEQUENCE)
    private long id;
    @NonNull
    private String product;

    @NonNull // just for
requiredargsconstructor
    private double price;

    @ManyToOne
    @JoinColumn(nullable = false)
    @ToString.Exclude
    private Order order;

    public void setOrder(Order order)
    {
        if (this.order != null) {
this.order.getLines().remove(this);
        }
        this.order = order;
        if (order != null) {
            order.getLines().add(this);
        }
    }
}

```

```

12:25:16.807      [IJava-executor-0]
INFO      notebook  --  Order(id=1,
date=2026-10-01T12:25:16.666150)
[Line(id=1,          product=Paper,
price=5.0),          Line(id=3,
product=Scissor,     price=8.0),
Line(id=2, product=Pen, price=1.0)]

```

```

No JDBC connection available.
Set system properties
'jdbc.url' (and optionally
'jdbc.user'/'jdbc.password'), or
provide a Connection in the kernel
environment.

```

Figure 7

```

No JDBC connection available.
Set system properties
'jdbc.url' (and optionally
'jdbc.user'/'jdbc.password'), or
provide a Connection in the kernel
environment.

```

Figure 8

3 Many to Many (N-M)

L'annotation `@ManyToMany` en Java Persistence API (JPA) est utilisée pour modéliser une relation many-to-many entre deux entités. La table d'association sera alors générée et utilisée implicitement.

```

Dot Executable: /opt/local/bin/dot
Dot executable does not exist
Cannot find Graphviz. You should try

```

```

@startuml
testdot
@enduml

```

or

```

java -jar plantuml.jar -testdot

```

Figure 9. – Un Customer a plusieurs adresses (qui peuvent être associées à d'autres)

No JDBC connection available.
Set system properties
'jdbc.url' (and optionally
'jdbc.user'/'jdbc.password'), or
provide a Connection in the kernel
environment.

Figure 10

```
@Getter
@Setter
@ToString
@RequiredArgsConstructor(staticName
= "of")
@NoArgsConstructor(access
= AccessLevel.PROTECTED)

@Entity
@Table(name = "CUSTOMER", schema =
"EX_MANY_TO_MANY")
public class Customer {
    @Id
    @GeneratedValue
    private long id;

    @NotNull
    private String name;

    @ManyToMany
    @ToString.Exclude
    @JoinTable(schema
= "EX_MANY_TO_MANY")
    private Set<Address> places =
new HashSet<>();

    public void addPlace(Address
address) {
        this.places.add(address);
        address.getOccupants().add(this);
    }

    public void removePlace(Address
address) {
```

```
        this.places.remove(address);
        address.getOccupants().remove(this);
    }
}
```

```
@Getter
@Setter
@ToString
@RequiredArgsConstructor(staticName
= "of")
@NoArgsConstructor(access
= AccessLevel.PROTECTED)

@Entity
@Table(name = "ADDRESS", schema =
"EX_MANY_TO_MANY")
public class Address {
    @Id
    @GeneratedValue
    private long id;

    @NotNull
    private String addressDetail;

    @ManyToMany(mappedBy = "places")
    @ToString.Exclude
    private Set<Customer> occupants=
new HashSet<>();

    public void addOccupant(Customer
customer) {
        this.occupants.add(customer);
        customer.getPlaces().add(this);
    }

    public void
removeOccupant(Customer customer) {
        this.occupants.remove(customer);
        customer.getPlaces().remove(this);
    }
}
```

```
}
```

No JDBC connection available.
Set system properties
'jdbc.url' (and optionally
'jdbc.user'/'jdbc.password'), or
provide a Connection in the kernel
environment.

Figure 11

No JDBC connection available.
Set system properties
'jdbc.url' (and optionally
'jdbc.user'/'jdbc.password'), or
provide a Connection in the kernel
environment.

Figure 12

No JDBC connection available.
Set system properties
'jdbc.url' (and optionally
'jdbc.user'/'jdbc.password'), or
provide a Connection in the kernel
environment.

Figure 13