

# 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

2026-10-14

### Table des matières

|                             |   |
|-----------------------------|---|
| 1 One To One (1-1) .....    | 1 |
| 2 One to Many (1-N) .....   | 4 |
| 2.1 Avec deux entités ..... | 4 |

```
[postgres] starting...
[postgres] OK
[maven] starting...
[maven] OK
[diag] 1/5 classpath du kernel
exposée
[diag] 2/5 imports slf4j/logback OK
[diag] 3/5 contexte logback OK
[diag] 4/5 %jars OK
[diag] Postgres driver registered OK
[diag] 5/5 imports JPA/
DatabaseManager OK
17:20:43.587 [IJava-executor-0] INFO
f.u.b.d.jpa.hello.DatabaseManager
-- Database settings:
URL=jdbc:postgresql://dind/notebook-
db, User=dba
17:20:43.595 [IJava-executor-0] INFO
f.u.b.d.jpa.hello.DatabaseManager
-- Starting JPA with URL:
jdbc:postgresql://dind/notebook-db
and User: dba
17:20:44.794 [IJava-executor-0] INFO
com.zaxxer.hikari.HikariDataSource
```

```
-- HikariPool-1 - Starting...
17:20:45.039 [IJava-executor-0] INFO
com.zaxxer.hikari.pool.HikariPool --
HikariPool-1 - Added connection org.postgresql.jdbc.PgCo
17:20:45.041 [IJava-executor-0] INFO
com.zaxxer.hikari.HikariDataSource
-- HikariPool-1 - Start completed.
--> Connexion JPA réussie (Tentative
1) !
[diag] cell init terminée ✓
```

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.

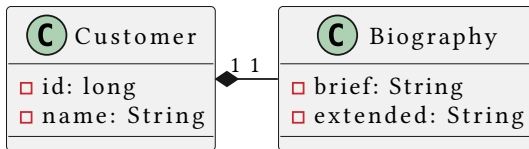


Figure 1. – Un Customer a une Biography

```

```dbgaof execute: false || output: true ||
echo: false || label: fig-rel-onetoone || fig-
cap: L'unique relation CUSTOMER || vscode:
{languageId: java} // PRINT SCHEMA String
script=« % »;
  
```

```

IJava.getKernelInstance().getMagics().apply-
CellMagic(« rdbmsSchema »,
List.of(« ex_biography »),script); return null;
  
```

```

:::
:::

:::      {.content-visible      when-
profile="notes"}
Pour cela JPA propose des annotations
spécifiques :

[\@Embeddable](https://jakarta.ee/
specifications/persistence/3.1/
apidocs/jakarta.persistence/jakarta/
persistence/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](https://jakarta.ee/
specifications/persistence/3.1/
apidocs/jakarta.persistence/jakarta/
persistence/embedded): Utilisée au
  
```

niveau d'un membre de classe annoté avec `\@Embeddable`, cette annotation 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.

```

:::

:::      {.content-visible      when-
profile="slides"}
- Pour cela, JPA propose :
  - [\@Embeddable](https://jakarta.
ee/specifications/persistence/3.1/
apidocs/jakarta.persistence/jakarta/
persistence/embeddable): Indique que
les membres d'une classe peuvent être
persistés en tant qu'attributs de la
classe qui les utilise.
  - [\@Embedded](https://jakarta.
ee/specifications/persistence/3.1/
apidocs/jakarta.persistence/jakarta/
persistence/embedded): Utilisé avec
`@Embeddable` pour intégrer ses
propriétés dans une autre entité
:::

:::      {.content-visible      when-
profile="slides"}
## Un client a une et une seule
biographie {.smaller}
:::

:::: {columns}

::: {column width="50%"}

::: {cell filename='null' quarto-
private-1={"key":"vscode","value":
{"languageId":"java"}}}
execution_count=4}
  
```

```

::: {#fig-class-embeddable .cell-
output .cell-output-display .cell-
output-markdown}
```Java
@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;
}

```

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_biog
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 2

## 2 One to Many (1-N)

PlantUML 1.2026.0

This version of PlantUML is 277 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 3. – 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 {
    @NonNull
    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<>();
}
```

```
dbgaof execute: false ||| output:
true ||| echo: false ||| label: fig-rel-
onetomanyb ||| fig-cap: Les relations
LINE et EORDER ||| tags: [remove-
input] ||| vscode: {languageId: java} ||
PRINT SCHEMA String script="%"; IJava.getKernelInstance().
List.of("ex_one_to_many_b"),script);
return null;
```

### 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;
    @NotNull
    private String product;

    @NotNull // 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);
        }
    }
}

```

```

17:20:50.373 [IJava-executor-0]
INFO notebook -- Order(id=1,
date=2026-10-14T17:20:50.212876)
[Line(id=1, product=Pen, price=1.0),
Line(id=3, product=Scissor,
price=8.0), Line(id=2,
product=Paper, price=5.0)]

```

```

```dbgaof execute: false || output: true ||
echo: false || label: fig-rel-onetomanya || fig-
cap: Les relations LINE et EORDER || tags:

```

```
[remove-input] //| vscode: {languageId: java} //
PRINT SCHEMA String script=« % »;
```

```
IJava.getKernelInstance().getMagics().apply-
CellMagic(« rdbmsSchema »,
List.of(« ex_one_to_many_a »),script); return
null;
```

```
::: {#fig-rel-onetomanya-query .cell
quarto-
private-1='{ "key": "vscode", "value":
{"languageId": "java"} }'
execution_count=16}

::: { .cell-output .cell-output-
stdout }
```

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

```
:::
:::

## Many to Many (N-M)

::: { .content-visible when-
profile="notes" }
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.
:::

::: { .content-visible when-
profile="slides" }
- \@ManyToMany en Java Persistence
API (JPA) :
- Utilisée pour modéliser
```

une relation many-to-many entre deux entités.

- La table d'association est générée et utilisée implicitement.

```
:::
::: { .cell quarto-
private-1='{ "key": "vscode", "value":
{"languageId": "java"} }'
execution_count=17}
```

```
::: { .cell-output .cell-output-
display }
![Un Customer a plusieurs
adresses (qui peuvent être
associées à d'autres)]
(02_java_jpa_L_associations_files/
figure-typst/fig-uml-manytomany-
output-1.svg){#fig-uml-manytomany}
:::
:::
```

```
```{java} execute: false
//| output: true
//| echo: false
//| label: fig-rel-manytomany
//| fig-cap: Les relation CUSTOMER,
ADDRESS et la relation de
d'association
//| tags: [remove-input]
//| vscode: {languageId: java}
// PRINT SCHEMA
String script="%";
```

```
IJava.getKernelInstance().getMagics().applyCellMagic("r
List.of("ex_many_to_many"),script);
return null;
```

```
@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;

    @NonNull
    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;

    @NonNull
    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);
    }
}

```

```

17:20:51.797 [IJava-executor-0] WARN
o.a.h.c.p.ResponseProcessCookies --
Invalid cookie header: "set-cookie: __cf_bm=cHvAo3WS6bzF
_xkoYchWz1V_fELE2Pe3ihqL2c40PsEDtsBa2TrsuV_2AoqIvY0ZxZy
HttpOnly; SameSite=None; Secure;
Path=/;
Domain=repo.maven.apache.org;
Expires=Wed, 14 Oct 2026 17:50:51

```

```

GMT". Invalid 'expires' attribute:
Wed, 14 Oct 2026 17:50:51 GMT
17:20:51.903 [IJava-executor-0] WARN
o.a.h.c.p.ResponseProcessCookies --
Invalid cookie header: "set-cookie: __cf_b
Ib2K3jVSxrEvmccLAjR9cMJBHs.Hl052m2DZLU0Zf
HttpOnly; SameSite=None; Secure;
Path=/;
Domain=repo.maven.apache.org;
Expires=Wed, 14 Oct 2026 17:50:51
GMT". Invalid 'expires' attribute:
Wed, 14 Oct 2026 17:50:51 GMT
17:20:51.950 [IJava-executor-0] WARN
o.a.h.c.p.ResponseProcessCookies --
Invalid cookie header: "set-cookie: __cf_b
YD.mQcJsML8RoUbA_QpqFegecR2jiMzt7UfG08d4i
HttpOnly; SameSite=None; Secure;
Path=/;
Domain=repo.maven.apache.org;
Expires=Wed, 14 Oct 2026 17:50:51
GMT". Invalid 'expires' attribute:
Wed, 14 Oct 2026 17:50:51 GMT
17:20:51.993 [IJava-executor-0] WARN
o.a.h.c.p.ResponseProcessCookies --
Invalid cookie header: "set-cookie: __cf_b
hFpiR60Q_Y8EnaNU8.vFCVYLilp.yirWg7QMrZ2fa
HttpOnly; SameSite=None; Secure;
Path=/;
Domain=repo.maven.apache.org;
Expires=Wed, 14 Oct 2026 17:50:51
GMT". Invalid 'expires' attribute:
Wed, 14 Oct 2026 17:50:51 GMT
17:20:52.039
[BasicRepositoryConnector-
repo.maven.apache.org-0-1] WARN
o.a.h.c.p.ResponseProcessCookies --
Invalid cookie header: "set-cookie: __cf_b
WwIAzs.LwqXFEEv9QU00mjn021_WJ0RAAEF7JbeMLiGBCumym4m/qKYlh4_I.MGowLRdRbH_CgtGLuqDtHlp9bnFa7rbroLL
HttpOnly; SameSite=None; Secure;
Path=/;
Domain=repo.maven.apache.org;
Expires=Wed, 14 Oct 2026 17:50:52
GMT". Invalid 'expires' attribute:

Wed, 14 Oct 2026 17:50:52 GMT
17:20:52.094
[BasicRepositoryConnector-
repo.maven.apache.org-0-0] WARN
o.a.h.c.p.ResponseProcessCookies -- <AZ1v0-1791998451
Invalid cookie header: "set-cookie: __cf_b
vb8ScoLqrP6.C0n5Nhn0esQqCmtavR7Wi_xq_WpSf7rd64gDXLVnJc
HttpOnly; SameSite=None; Secure;
Path=/;
Domain=repo.maven.apache.org;
Expires=Wed, 14 Oct 2026 17:50:52
GMT". Invalid 'expires' attribute:
Wed, 14 Oct 2026 17:50:52 GMT
17:20:52.095
[BasicRepositoryConnector-
repo.maven.apache.org-0-3] WARN CypatY-1791998451
o.a.h.c.p.ResponseProcessCookies -- 5xmIw3n6pmP3Ayh5Q
Invalid cookie header: "set-cookie: __cf_bm=03T8DwdrwC6h
ZBgwmR0yFYp06qqUyLoluQo1haGI8SycuKFU7xDX9k5ziurMTIGBU6K
HttpOnly; SameSite=None; Secure;
Path=/;
Domain=repo.maven.apache.org;
Expires=Wed, 14 Oct 2026 17:50:52
GMT". Invalid 'expires' attribute:
Wed, 14 Oct 2026 17:50:52 GMT 9ajC4-1791998451
17:20:52.100 ApxUr2vEgu4JE68k2
[BasicRepositoryConnector-
repo.maven.apache.org-0-2] WARN
o.a.h.c.p.ResponseProcessCookies --
Invalid cookie header: "set-cookie: __cf_bm=HwymWgGBBYbL
eRjtXKlZmPRZ76fRIDRjryy2yV.X7Fj_zoTznVWpKRPniFHVg9DTLp4
HttpOnly; SameSite=None; Secure;
Path=/;
Domain=repo.maven.apache.org;
Expires=Wed, 14 Oct 2026 17:50:52
GMT". Invalid 'expires' attribute:
Wed, 14 Oct 2026 17:50:52 GMT 0J9eE-1791998452

```

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

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

Figure 5

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