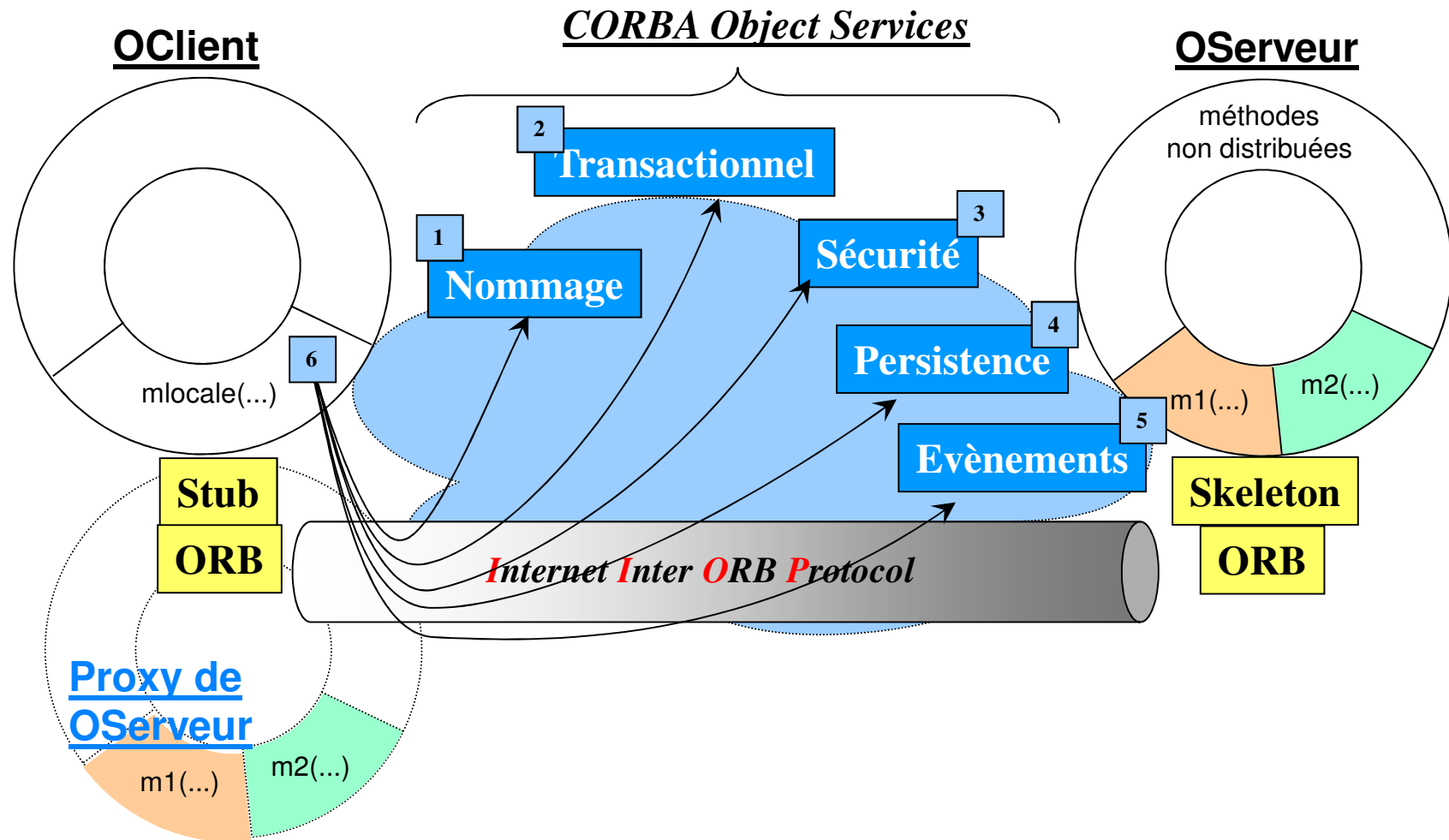




Les Enterprise Java Beans

Les Services CORBA

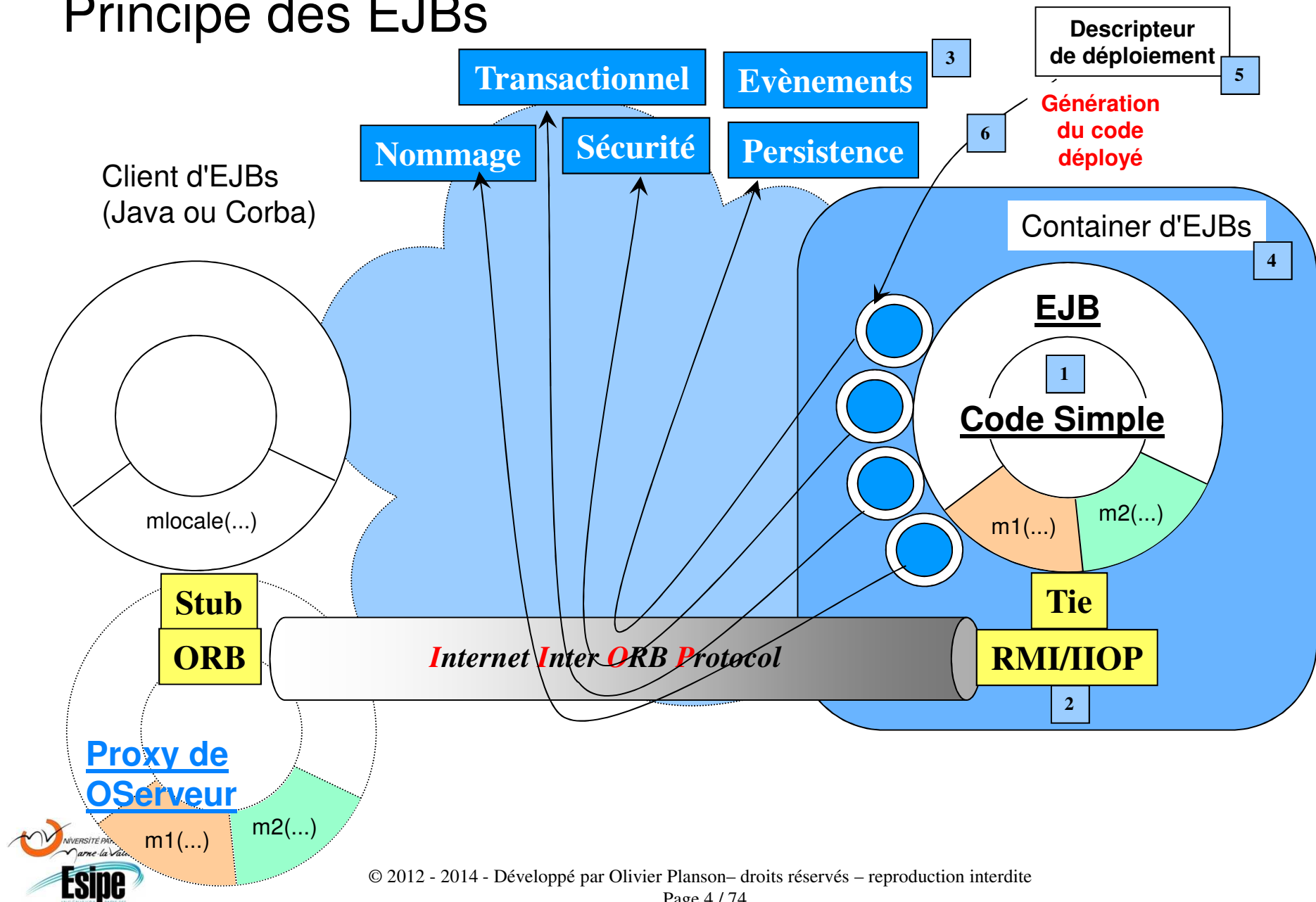




Rôles des EJBs

- ❑ Les EJBs sont des objets « métier » pouvant travailler en mode distribué ou en mode local.
- ❑ Ils pourront utiliser les différents services (Nommage, Sécurité, Transactionnel etc...) en ayant le moins possible à impacter leur code.
- ❑ Avantages:
 - Développement plus simple.
 - Grande potentialité de réutilisation.

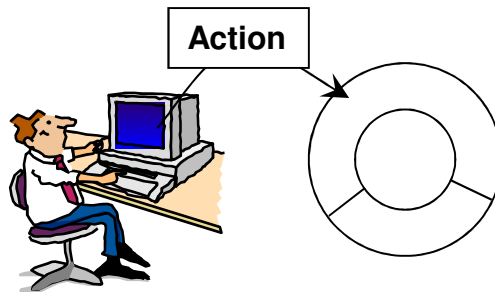
Principe des EJBs





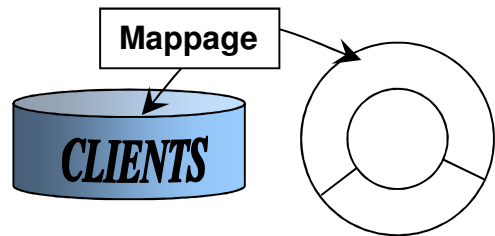
Les Différents Types d'EJB(s)

Les EJBs Session:



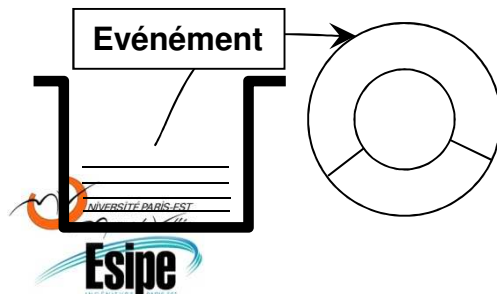
- Ils sont liés à un client (Client lourd, Servlet, JavaBean etc...). Ils sont impliqués dans l'aspect Transactionnel et de Sécurité par rapport à ce client

Les EJBs Entité:



- Ils sont pas liés à un client particulier, mais représentent les données. Ils seront vus comme des objets persistants

Les MDB (Message Driven Beans):



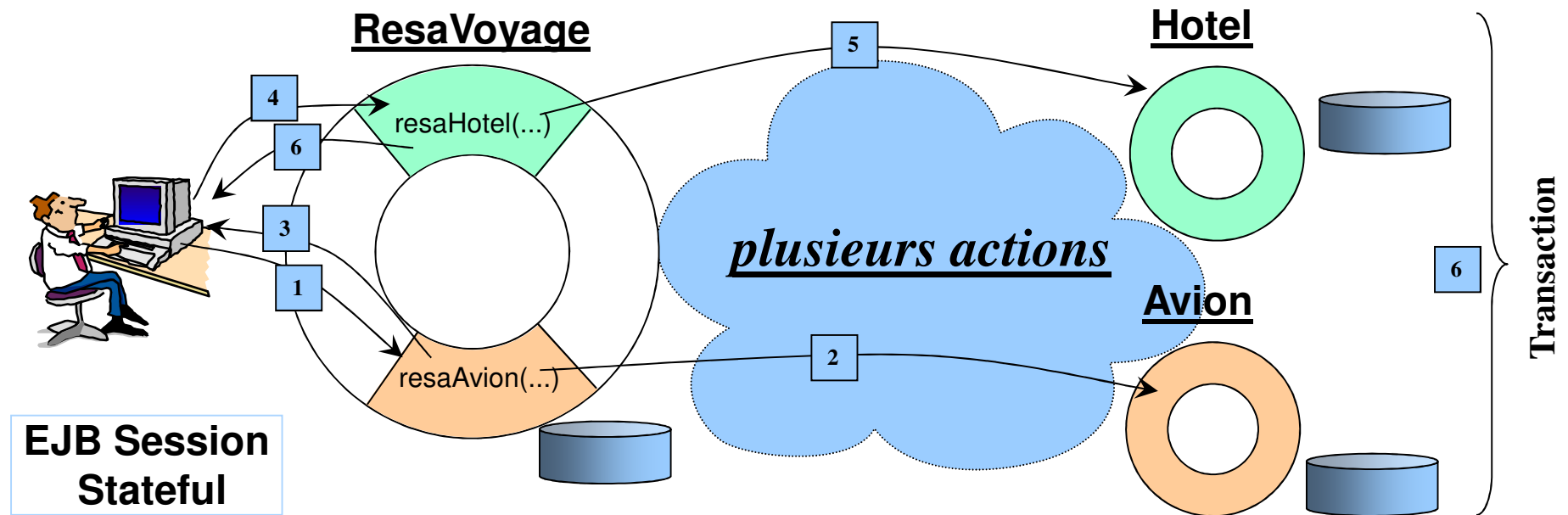
- Ils sont à l'écoute des événements intervenant sur les files d'attente asynchrones (ex: JMS).



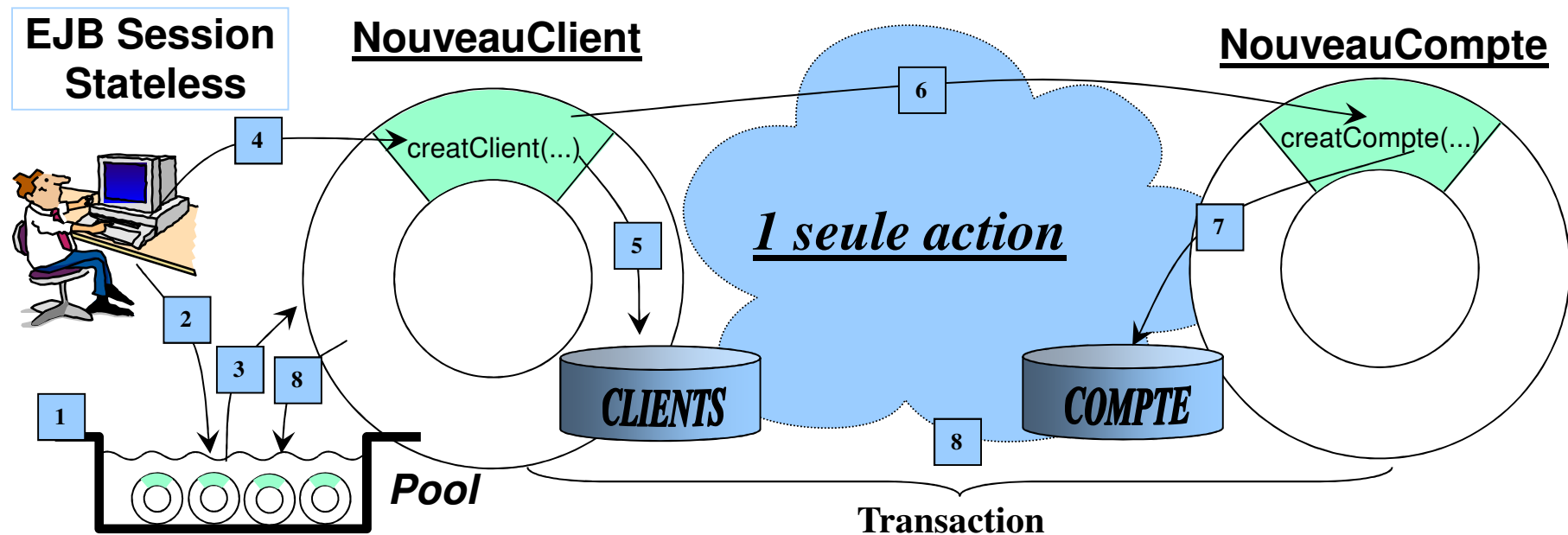
Les EJB's Session

V3

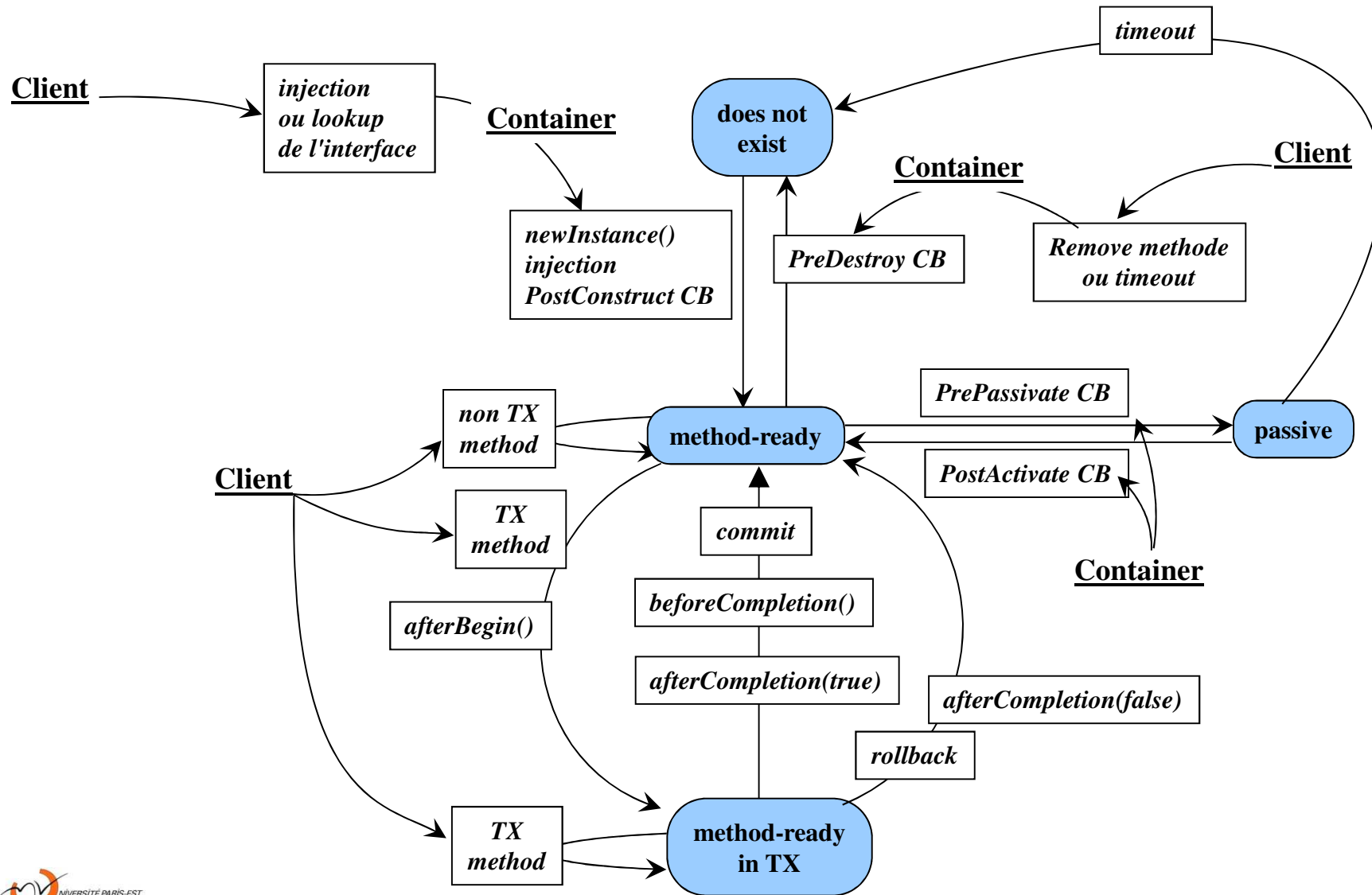
EJB Session Stateful



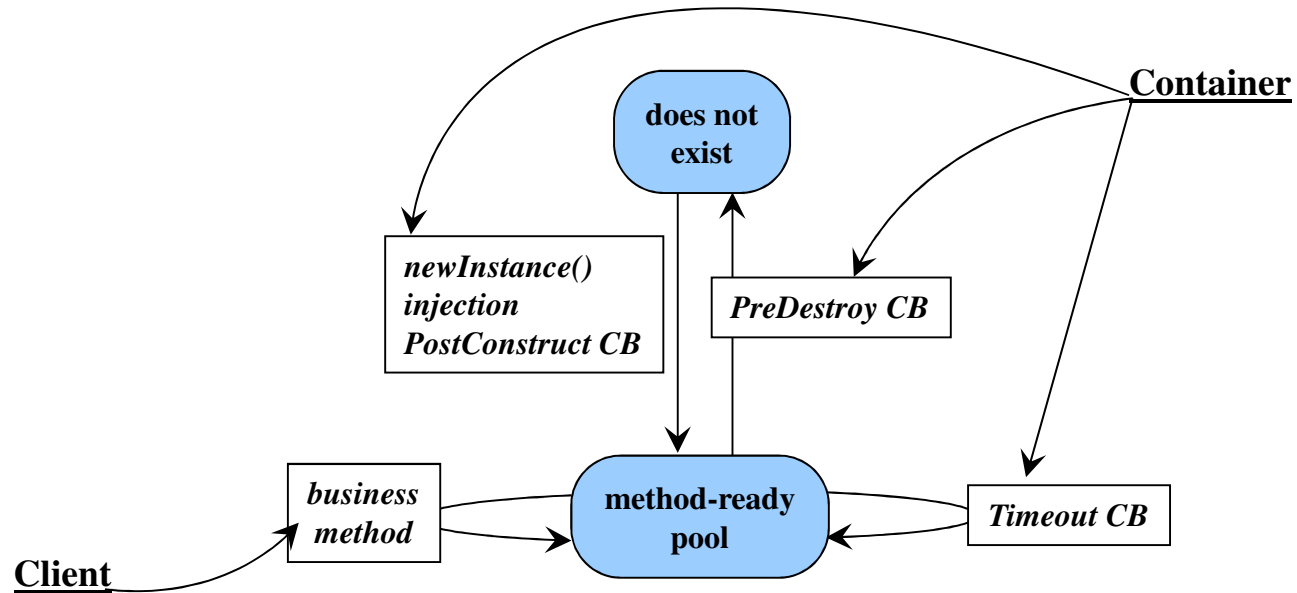
EJB Session Stateless



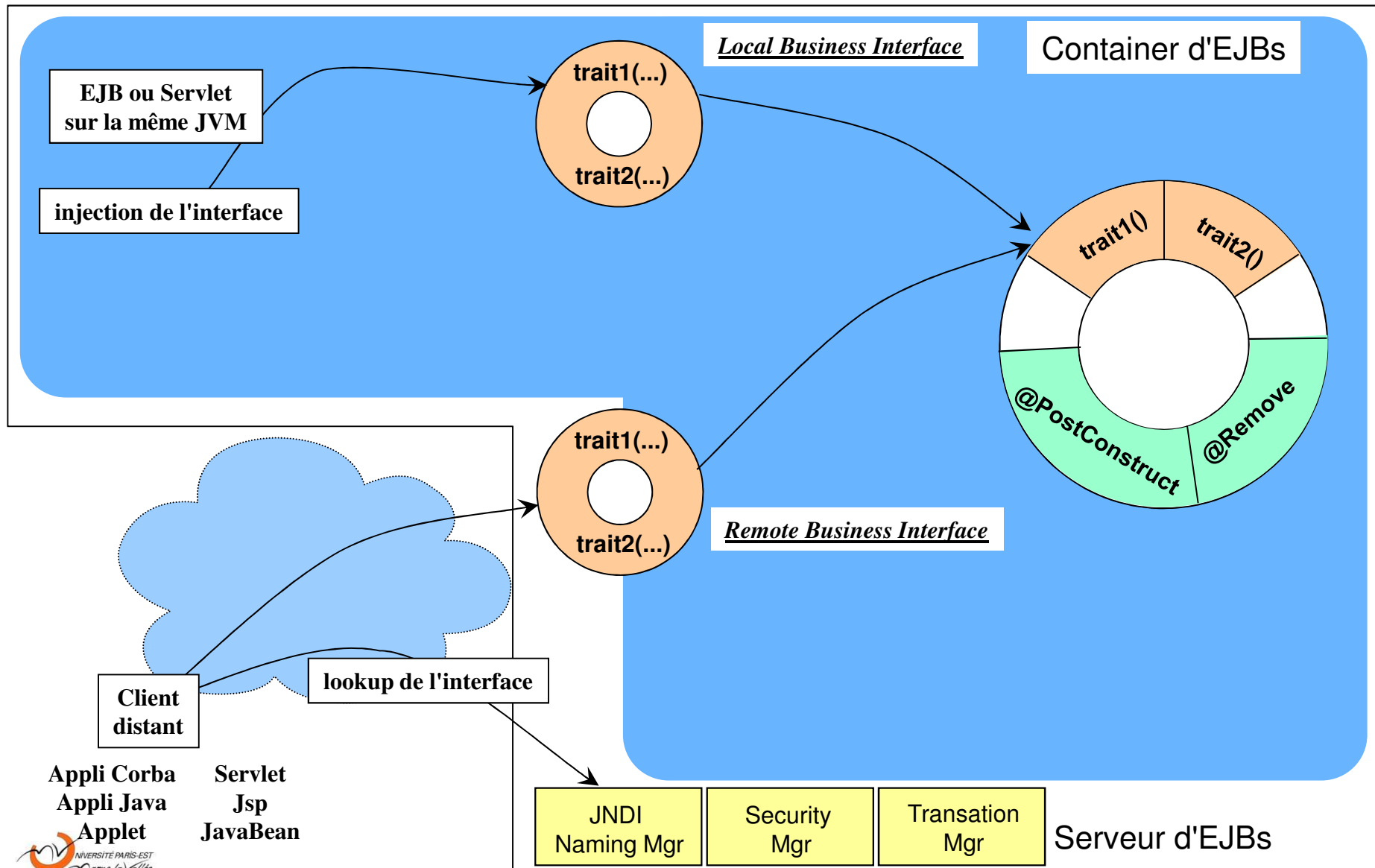
Cycle de vie d'un EJB Session Stateful



Cycle de vie d'un EJB Session Stateless



L'Accès aux EJBs Sessions





Structure de base d'un EJB Session

Interfaces

```
public interface MySession {  
    public void initialise(String param);  
    public int traite(String valeur);  
    public void termine();  
}
```

```
public interface RemoteMySession {  
    public void initialise(int i, String param);  
    public int traite(String valeur);  
    public void termine();  
}
```

Classe de l'EJB

```
@Stateless ou @Stateful  
@Local((RemoteMySession.class))  
@Remote((RemoteMySession.class))  
public class MySessionBean implements MySession, RemoteMySession, Serializable {  
    @PostConstruct  
    public void initialise(String param) { ..... }  
    @TransactionAttribute(TransactionAttributeType.REQUIRED)  
    public int traite(String valeur) { ..... }  
    @Remove  
    public void termine() { ..... }  
}
```

Client

```
private RemoteMySession mysess = null;  
.....  
try {  
    InitialContext ctx = new InitialContext();  
    mysess = (RemoteMySession) ctx.lookup(RemoteMySession.class.getName());  
} catch (Exception e) { e.printStackTrace (); }  
int res = mysess.traite("AAA");  
.....
```



Les "call-backs" (1)

Annotations destinées à solliciter certaines méthodes métier de l'EJB lors d'évènements spécifiques du cycle de vie d'un EJB Session ou MDB.

- **@PostConstruct:**

- Se définit pour une méthode qui sera appelée après l'instanciation de l'EJB

- **@Init:**

- Même rôle que *PostConstruct*, mais il peut y avoir plusieurs méthodes marquées *@Init*
 - Elle est présente pour permettre la migration EJB2.1 >> EJB3.0. Elle représente la méthode *create()* du *Home Interface*.

- **@PrePassivate:**

- Se définit pour une méthode qui sera appelée avant qu'un EJB Session Stateful soit endormi.

- **@PostActivate:**

- Se définit pour une méthode qui sera appelée avant qu'un EJB Session Stateful soit réactivé.

Removal method:

- **@Remove:**

- Se définit pour une méthode qui sera appelée avant qu'un EJB Session Stateful soit détruit.



Les "call-backs" (2)

Les 'callbacks' dans l'EJB

```
@Stateful
public class ExempleBean implements Exemple, Serializable {

    // ... ..
    @PostConstruct
    public void initialize () {
        // ... ..
    }
}
```

```
@Stateful
@CallbackListener(ExempleCallbackListener.class)
public class ExempleBean implements Exemple, Serializable {

    // ... ..
}
```

Les 'callbacks' dans un CallbackListener

```
public class ExempleCallbackListener {

    // ... ..
    @PostConstruct
    public void initialize (ExempleBean ex) {
        // ... ..
    }
}
```



Les "Interceptors" (1)

Annotations destinées à certaines méthodes qui devront être exécutées avant l'appel des méthodes métier de l'EJB.

```
@Stateful
public class ExempleBean implements Exemple, Serializable {

    // ... ..
```

@AroundInvoke

```
public Object methodeInterception (InvocationContext ctx)
    throws Exception {
```

```
    // traitement .....
    return ctx.proceed();
}
}
```

```
public interface InvocationContext {
    public Object getTarget();
    public Method getMethod();
    public Object[] getParameters();
    public void setParameters(Object[] params);
    public java.util.Map<String, Object> getContextData();
    public Object proceed() throws Exception;
}
```



Les "Interceptors" (2)

@Stateful

@**Interceptor(s)** (Intercepteur.class)

public class ExempleBean implements Exemple, Serializable {

//
}

public class Intercepteur {

//

@**AroundInvoke**

public Object methodeInterception (**InvocationContext** ctx)
throws Exception {

// traitement
return ctx.**proceed**();
}
}



Le Transactionnel

Annotations destinées à positionner le comportement transactionnel d'un EJB.

```
@Stateful
@TransactionManagement(TransactionManagementType.BEAN)
ou @TransactionManagement(TransactionManagementType.CONTAINER)
public class ExempleBean implements Exemple, Serializable {

    // ... ...
}
```

```
@Stateful
public class ExempleBean implements Exemple, Serializable {
    @TransactionAttribute(TransactionAttributeType.REQUIRED)
    public void methodeTransactionnelle () {
        // traitement .....
    }
}
```

```
public enum TransactionAttributeType {
    MANDATORY,
    REQUIRED,
    REQUIRES_NEW,
    SUPPORTS,
    NOT_SUPPORTED,
    NEVER
}
```



L'Injection

Permet de se substituer au lookup JNDI pour accéder à un EJB ou une ressource déclarée dans JNDI.
Ne peut, pour l'instant, s'effectuer qu'en local.

```
@Stateful
public class ExempleBean implements Exemple, Serializable {

    @EJB MonEJB1 ejb1;
    @EJB MonEJB2 ejb2;

    // utilisation des variables ejb1 et ejb2
    // ... ..

}
```

```
@Stateful
public class ExempleBean implements Exemple, Serializable {

    @Resource (name="MonDS")
    DataSource monDs;
    monDs.getConnection();
    // ... ..

}
```



Les Interfaces de l'EJB

Fichier *TraiteOperLocal* (Local Interface):

```
package operations;
@Local
public interface TraiteOperLocal {
    public void consulte(String pNo) throws TraitementException
    public void executeOp(String pOper, String pValeur) throws
                                TraitementException {
}
}
```

Fichier *TraiteOperRemote* (Remote Interface):

```
package operations;
@Remote
public interface TraiteOperRemote {
    public void consulte(String pNo) throws TraitementException
    public void executeOp(String pOper, String pValeur) throws
                                TraitementException {
}
}
```

Le code de l'EJB

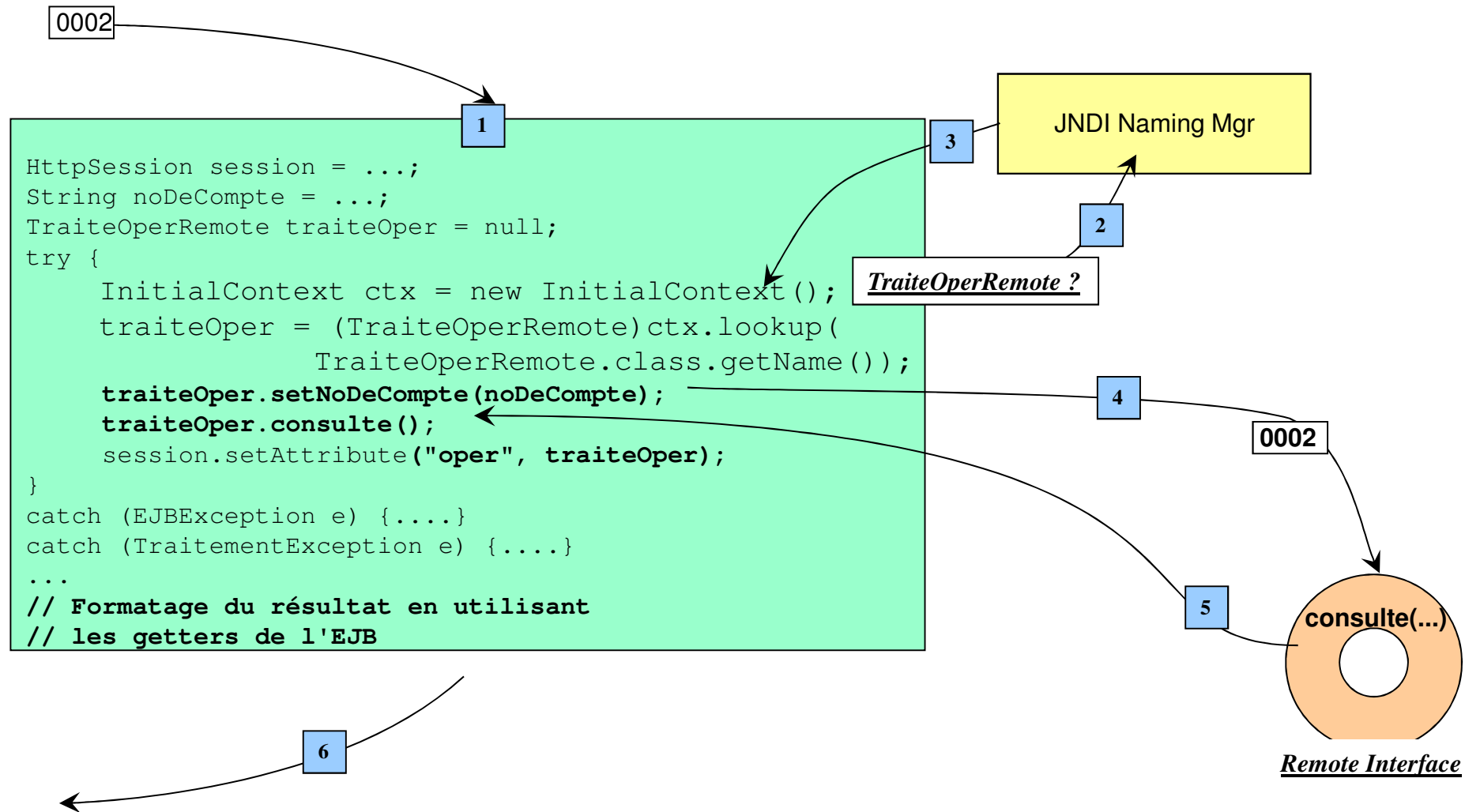
Fichier *TraiteOperBean*:

```
package operations;
@Stateful
@TransactionManagement(TransactionManagementType.BEAN)
public class TraiteOperBean implements TraiteOperLocal, TraiteOperRemote, Serializable {
    String no, nom, prenom, solde, soldePrec, soldeNouv;
    UserTransaction tx;
    @Resource SessionContext mySessionCtx;
    @Resource (name=" ClientsD ")
        DataSource ds;
    tx = mySessionCtx.getUserTransaction;
    public void consulte(String aNo) throws TraitementException {
        tx.begin();
        this.no = aNo;

        try {
            Connection connect = ds.getConnection ("USERID", "PASSWORD");
            .....
        }

        @Remove
    public void executeOp(String pOper, String pValeur) throws TraitementException {
        BigDecimal ancSolde;
        BigDecimal nouvSolde;
        .....
        if(nouvSolde.signum()== -1) {
            tx.rollback();
            throw new TraitementException("....");
        } else {
            ....
            tx.commit();
        }
    }
}
```

Le Code du Client (1er appel)



Le Code du Client (2ème appel)

0002, +, 50.10

7

```
...  
HttpSession session = ...;  
String op = ...;  
String valeur = ...;  
TraiteOperRemote traiteOper = null;  
try {  
    traiteOper = (TraiteOperRemote) session.getAttribute("oper");  
    traiteOperRI.executeOp(op, valeur );  
}  
catch (EJBException e) {....}  
catch (TraitementException e) {....}  
...  
// Formatage du résultat en utilisant  
// les getters de l
```

10

8

0002, +, 50.10

9

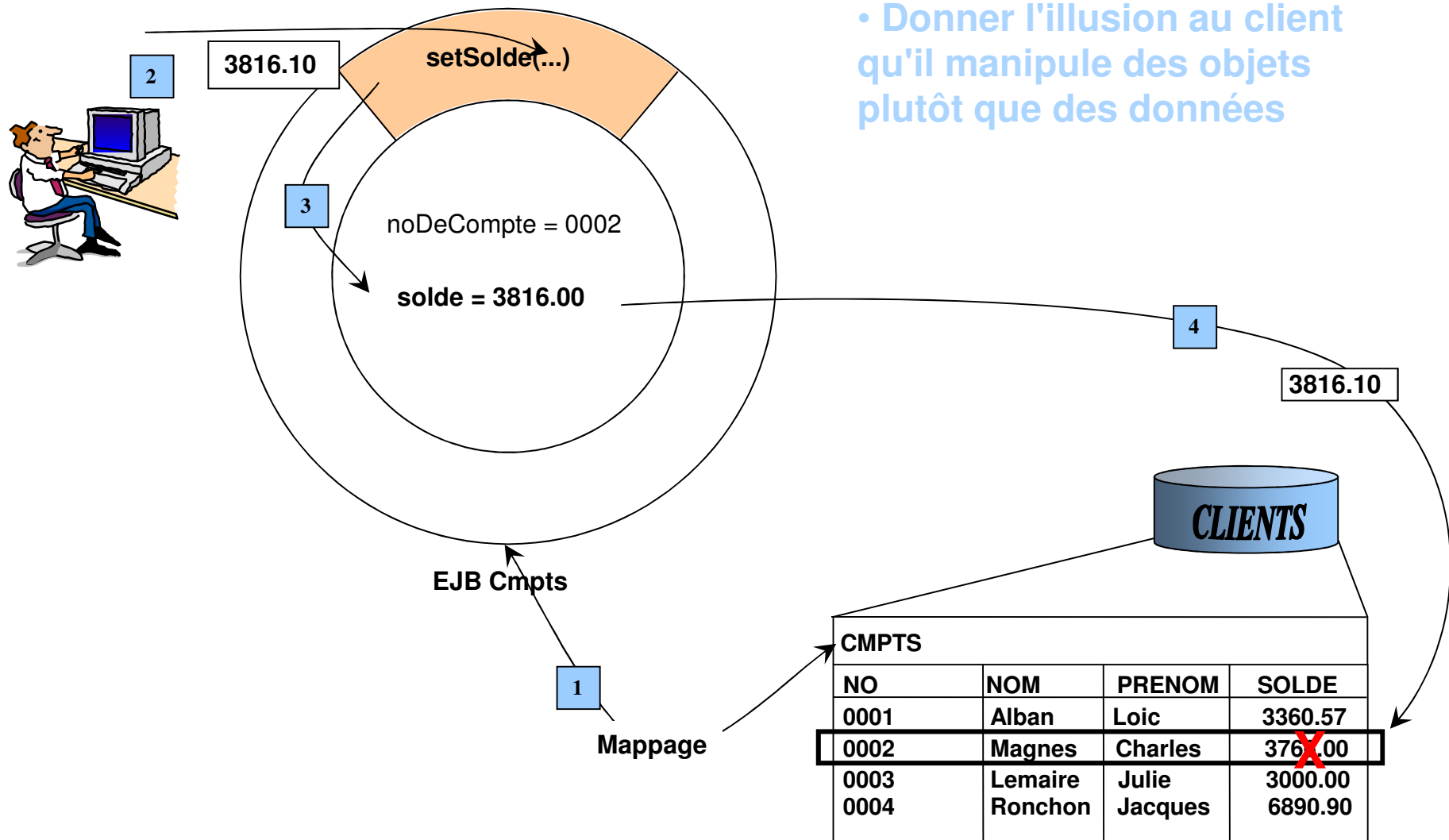
executeOp(...)

Remote Interface



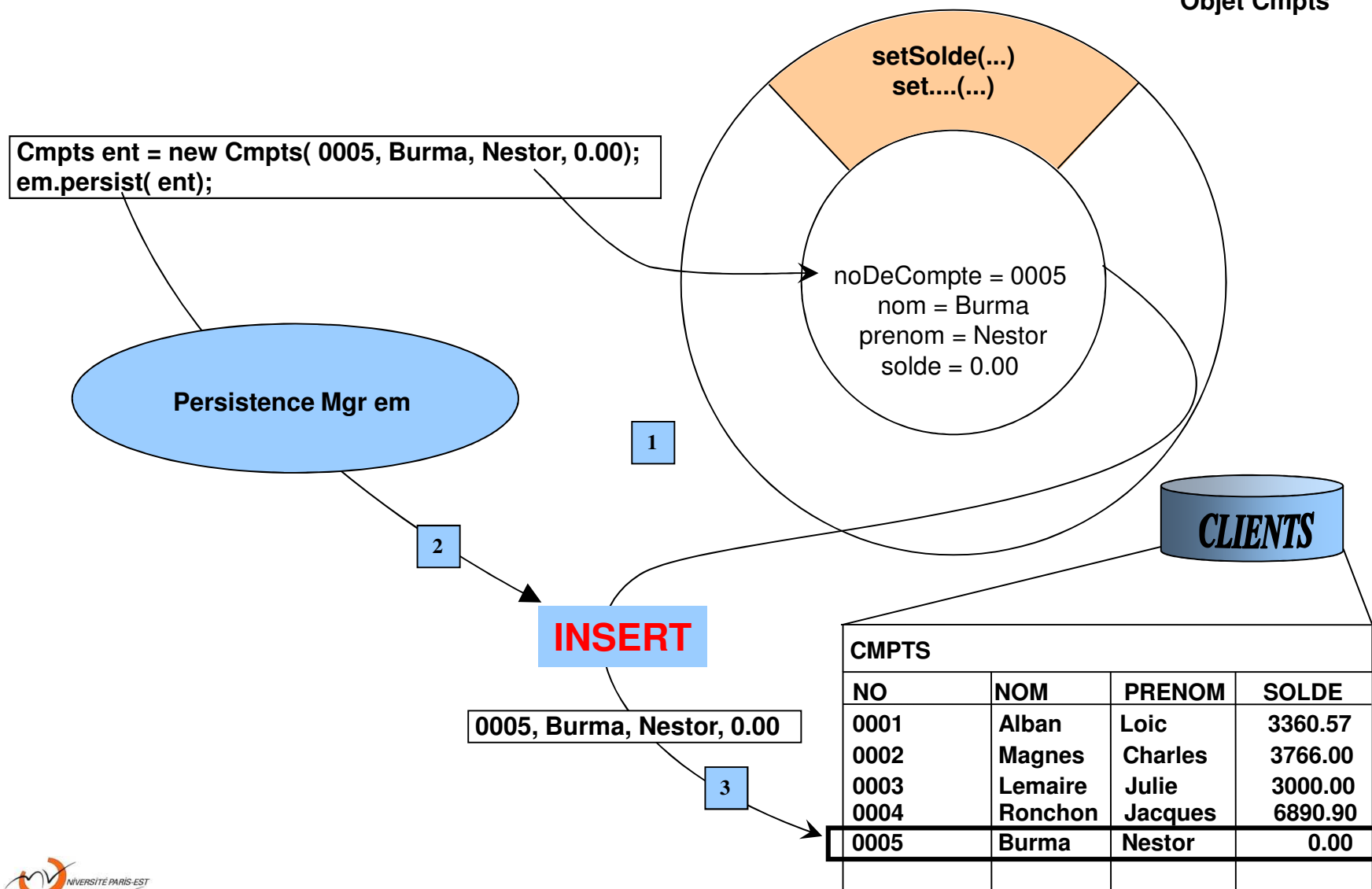
Les EJB's Entity V3

But des EJBs Entité

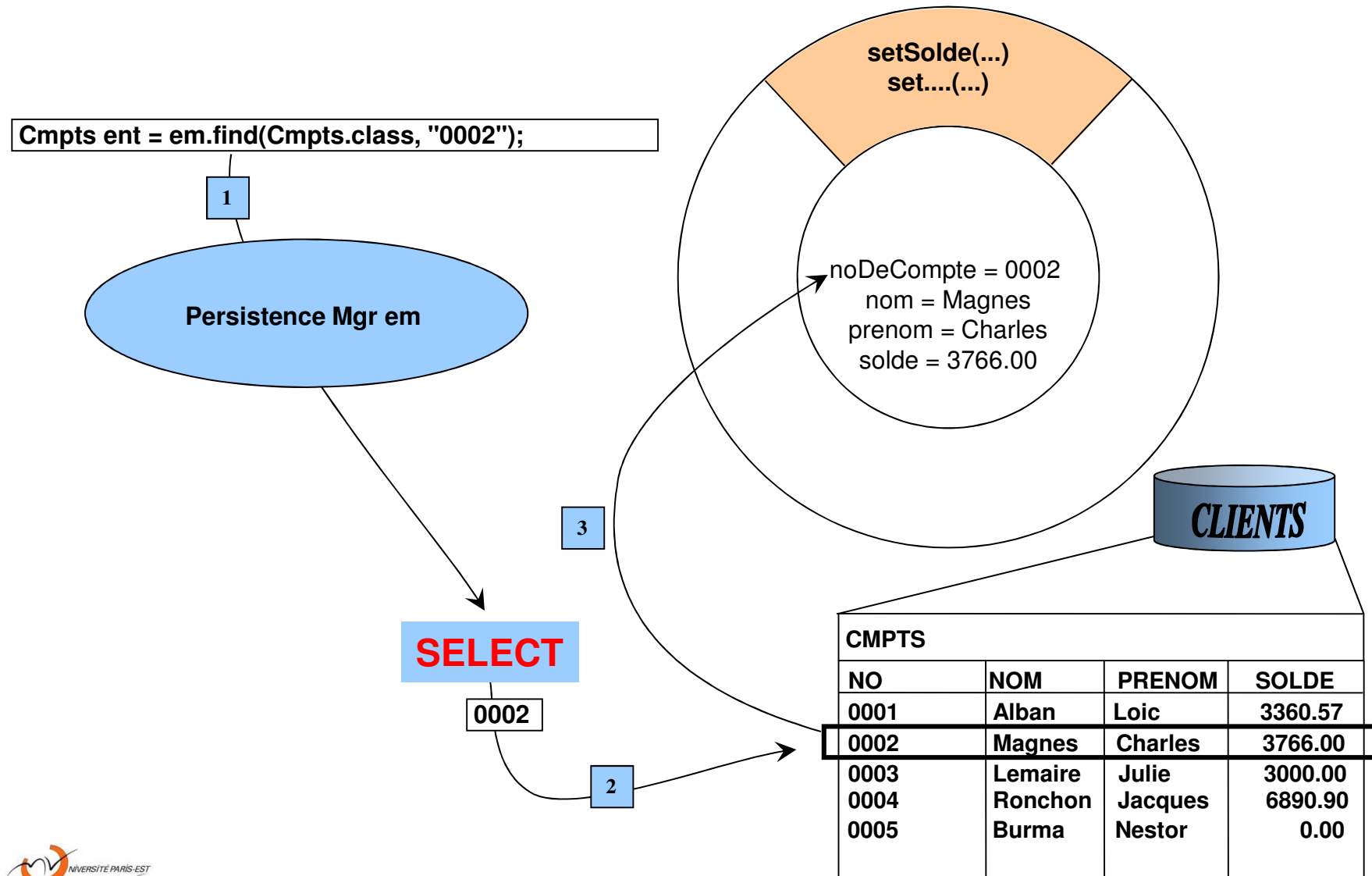


Opérations sur les EJBs Entité (1)

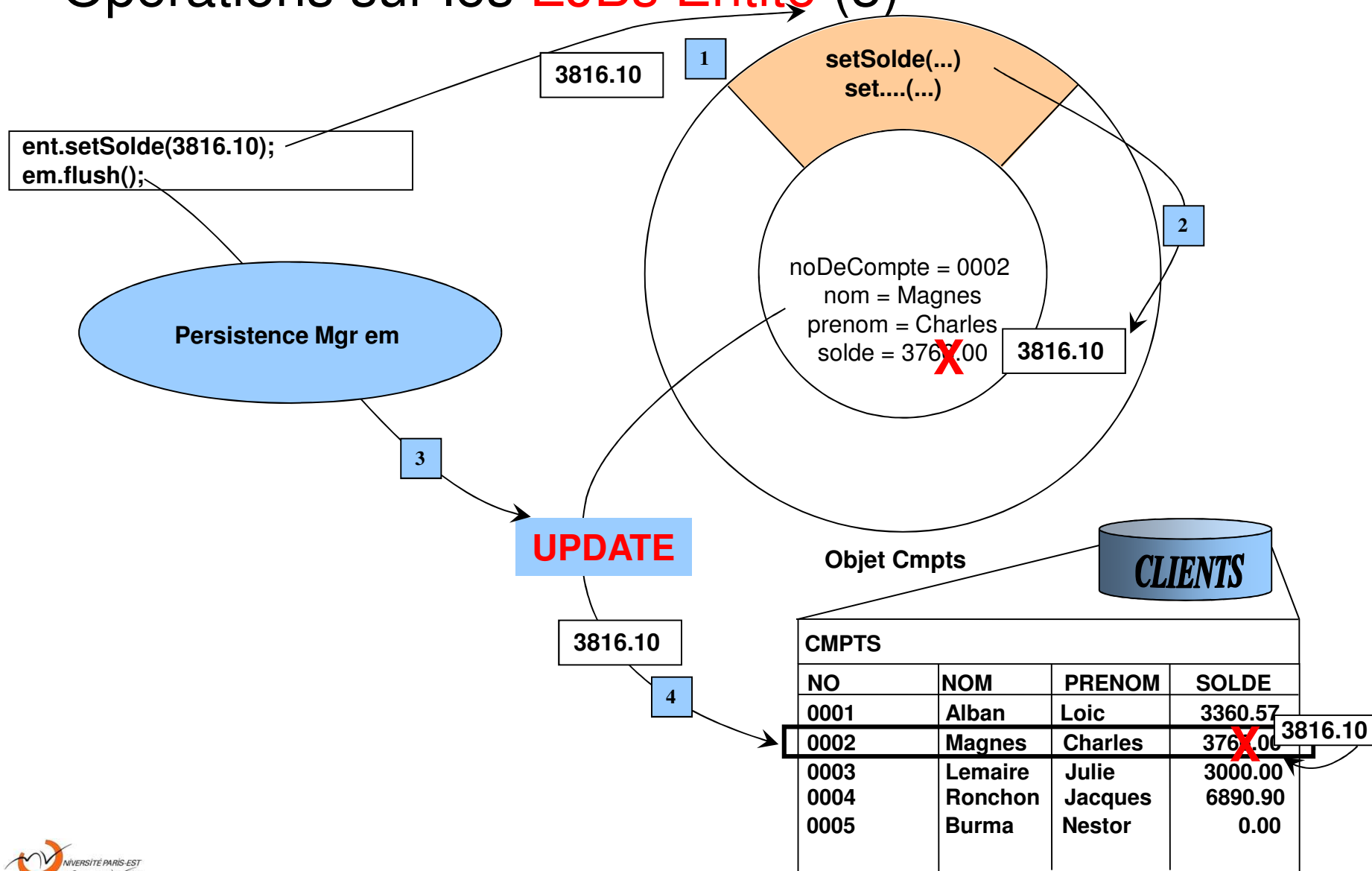
Objet Cmpts



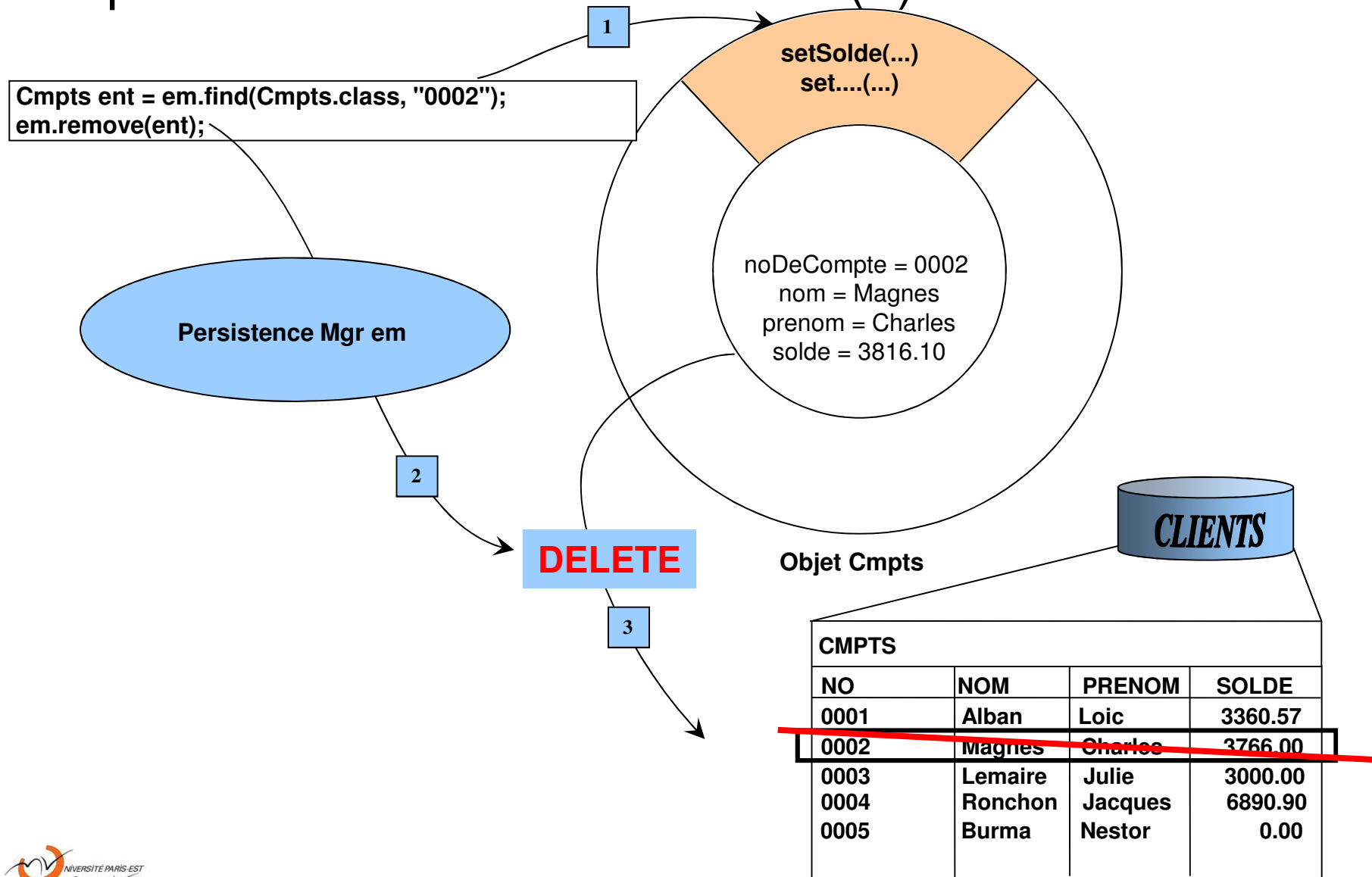
Opérations sur les EJBs Entité (2)



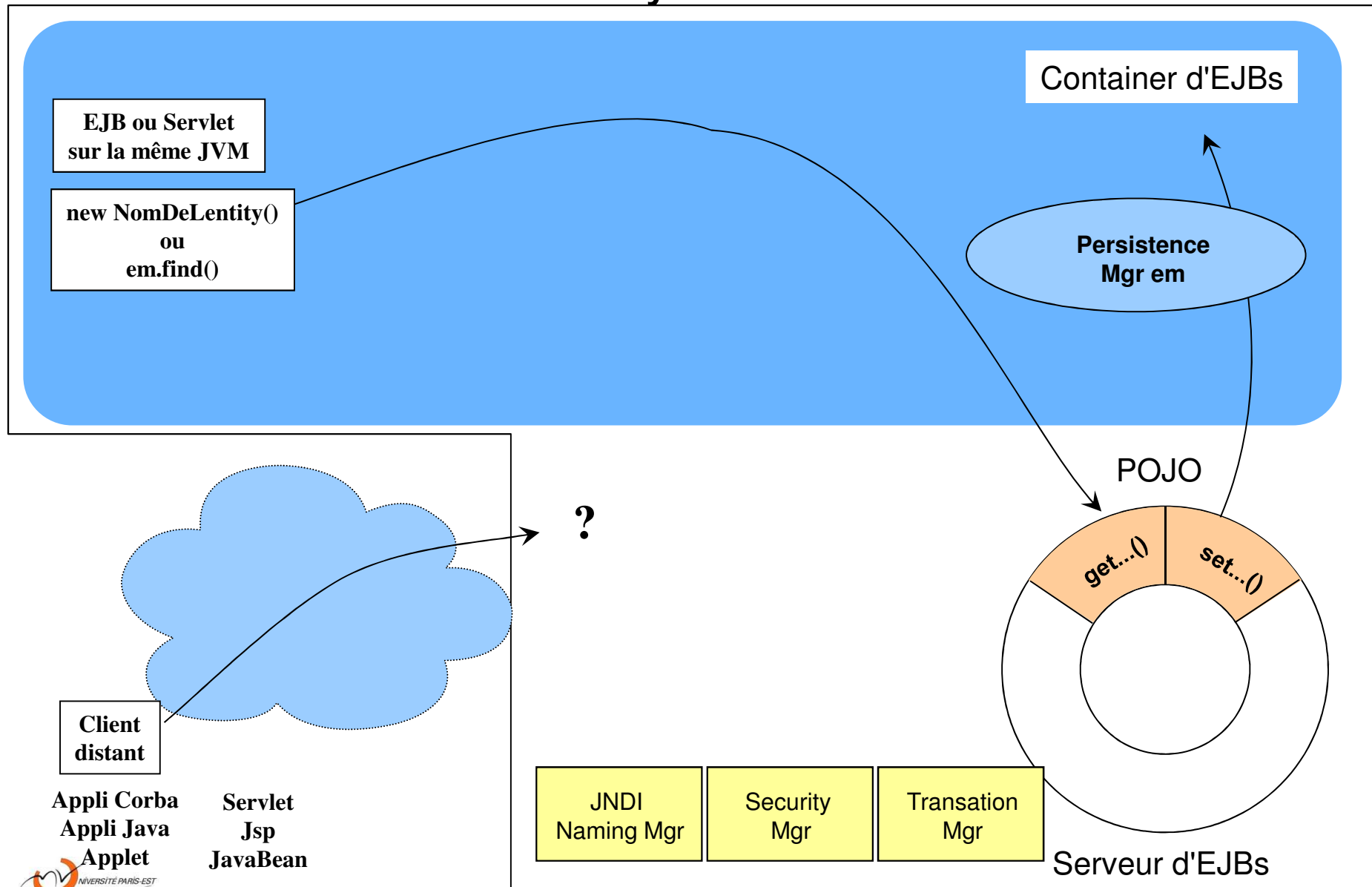
Opérations sur les EJBs Entité (3)



Opérations sur les EJBs Entité (4)



L'Accès aux EJBs Entity





Structure de base d'un EJB Entity V3

Classe
de l'EJB

```
@Entity
@Table(name="MYTABLE")
public class MyEntity Serializable{
    private int clePrimaire;
    private String var1;
    private String var2;
    public MyEntity(String var1, String var2) {.....}
    @Id(generate = GenerationType.Auto)
    public int getClePrimaire() {.....}
    public void setClePrimaire(int clePrimaire) {.....}
    @Column(name = "COL1", nullable =false, length = 4)
    public int getVar1() {.....}
    public void setVar1(String var1) {.....}
    @Column(name = "COL2", nullable =true, length = 10)
    public int getVar2() {.....}
    public void setVar2(String var2) {.....}

    .....
}
```



DataSource d'un EJB Entity V3

Fichier
myds-ds.xml
du répertoire
deploy

```
<?xml version="1.0" encoding="UTF-8"?>

<datasources>
  <local-tx-datasource>
    <jndi-name>MyDS</jndi-name>
    <connection-url>jdbc:mysql://localhost:3306/mabase</connection-url>
    <driver-class>com.mysql.jdbc.Driver</driver-class>
    <user-name>userId</user-name>
    <password>password</password>
    <exception-sorter-class-name>
      org.jboss.resource.adapter.jdbc.vendor.MySQLExceptionSorter
    </exception-sorter-class-name>
    <metadata>
      <type-mapping>mySQL</type-mapping>
    </metadata>
  </local-tx-datasource>
</datasources>
```

Appels d'un EJB Entity V3

Fichier
persistence.xml
du répertoire
META-INF

```
<?xml version="1.0" encoding="UTF-8"?>
<entity-manager>
  <name>myEntityMgr</name>
  <jta-data-source>java:/MyDS</jta-data-source>
  <properties>
    <property name="hibernate.hbm2ddl.auto" value="update">
    <property name="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect">
  </properties>
</entity-manager>
```

@PersistenceContext (unitName="myEntityMgr")

protected EntityManager **em**;

.....

```
MyEntity entity = new MyEntity("VAR1", "VAR2");
em.persist(entity);
```

.....

```
int clePrimaire = 1;
MyEntity retEntity = em.find(MyEntity.class, Integer.valueOf(clePrimaire));
retEntity.setVar1("VARX");
em.flush();
```

....

```
int bas = 10;
int haut = 20;
```

```
Collection <MyEntity> collect = return em.createQuery(
    "from MyEntity m where m.iD > :bas AND m.iD < :haut") .
    setParameter ("bas", new Integer (bas)) .
    setParameter ("haut", new Integer (haut)) .getResultList();
```

.....

```
em.remove(retEntity);.....
```

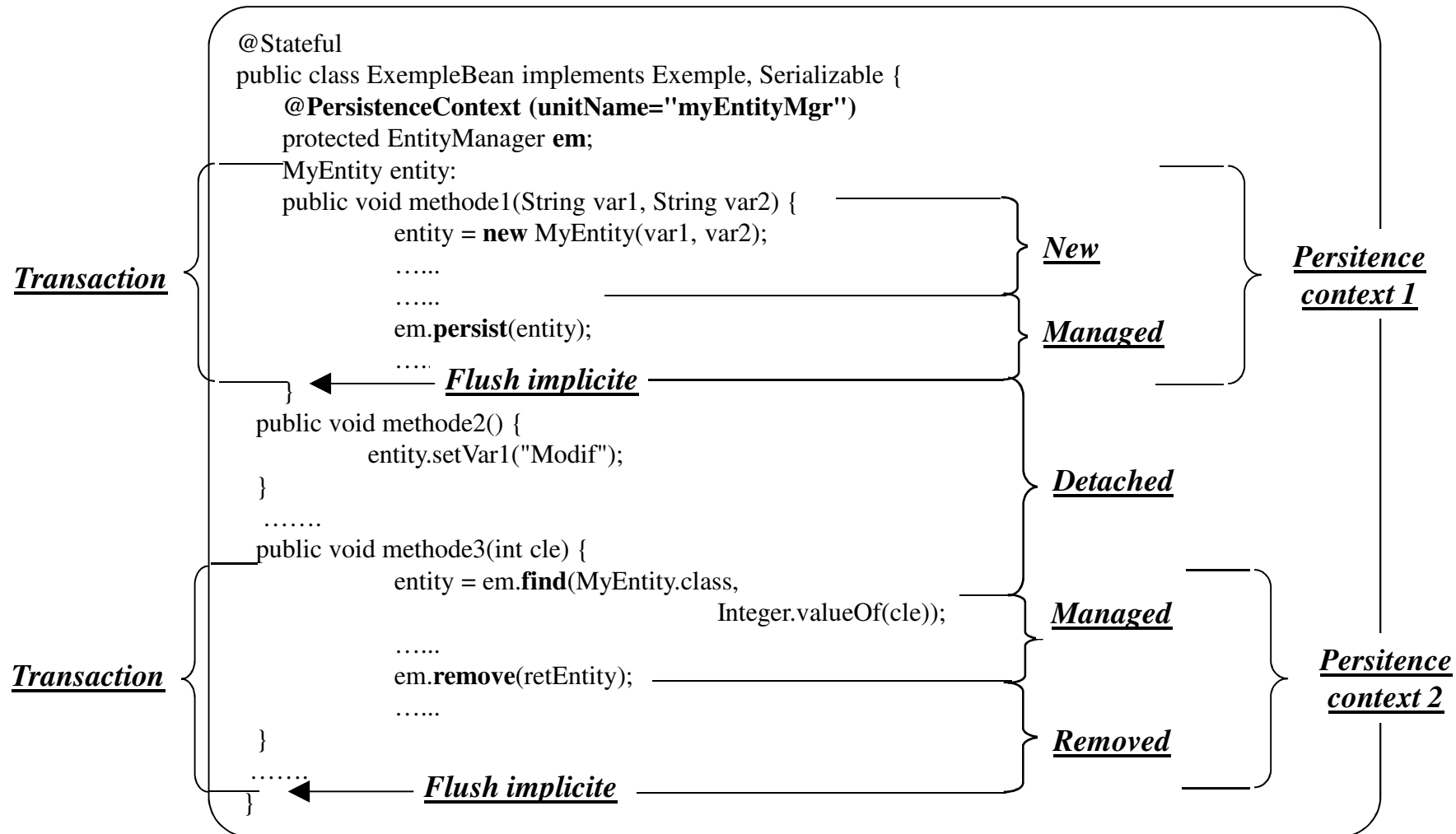
Client
(sur la même JVM que l'EJB)



Le Contexte de *Persitance*

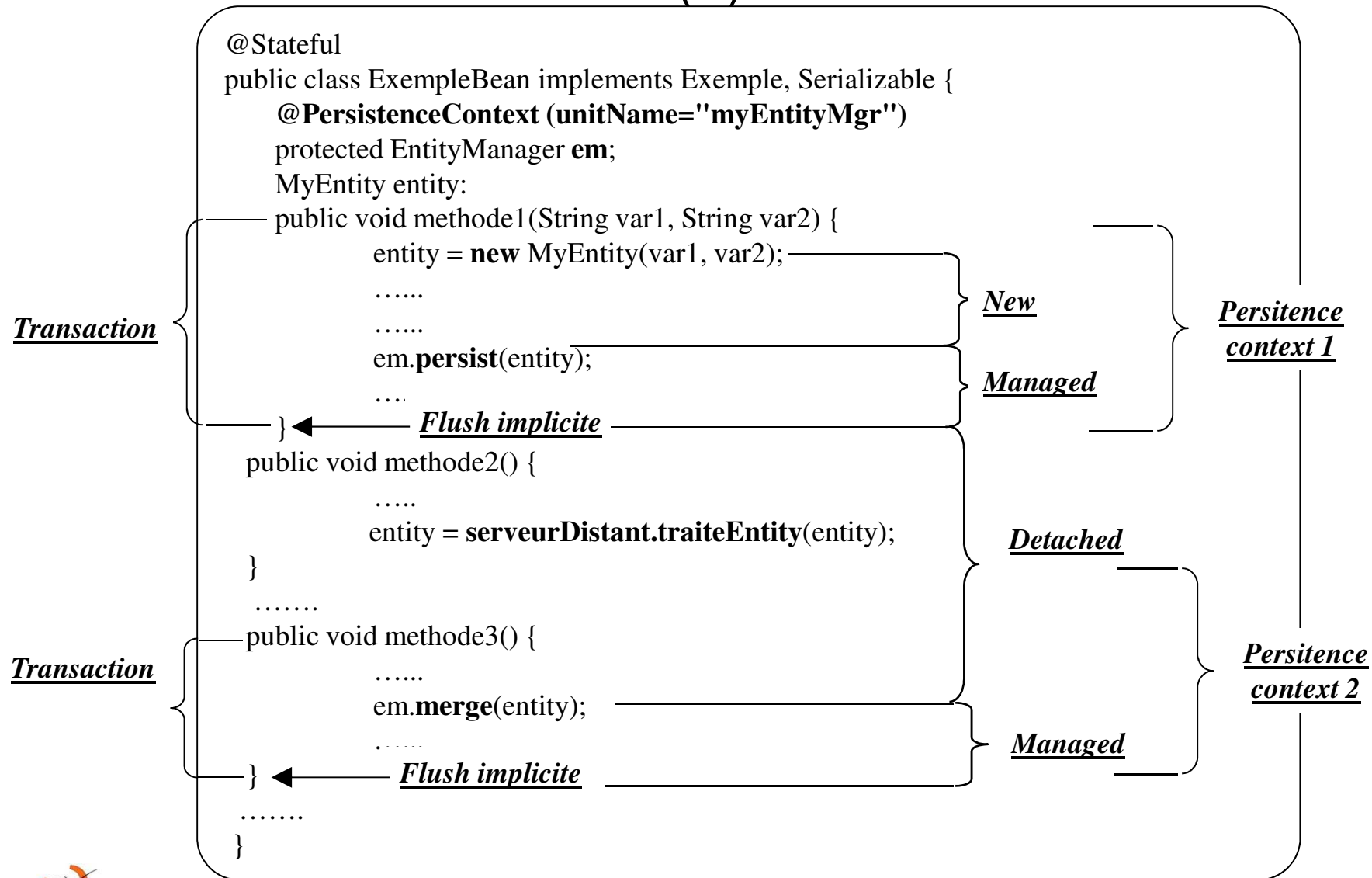


Le Persitence Context (1)

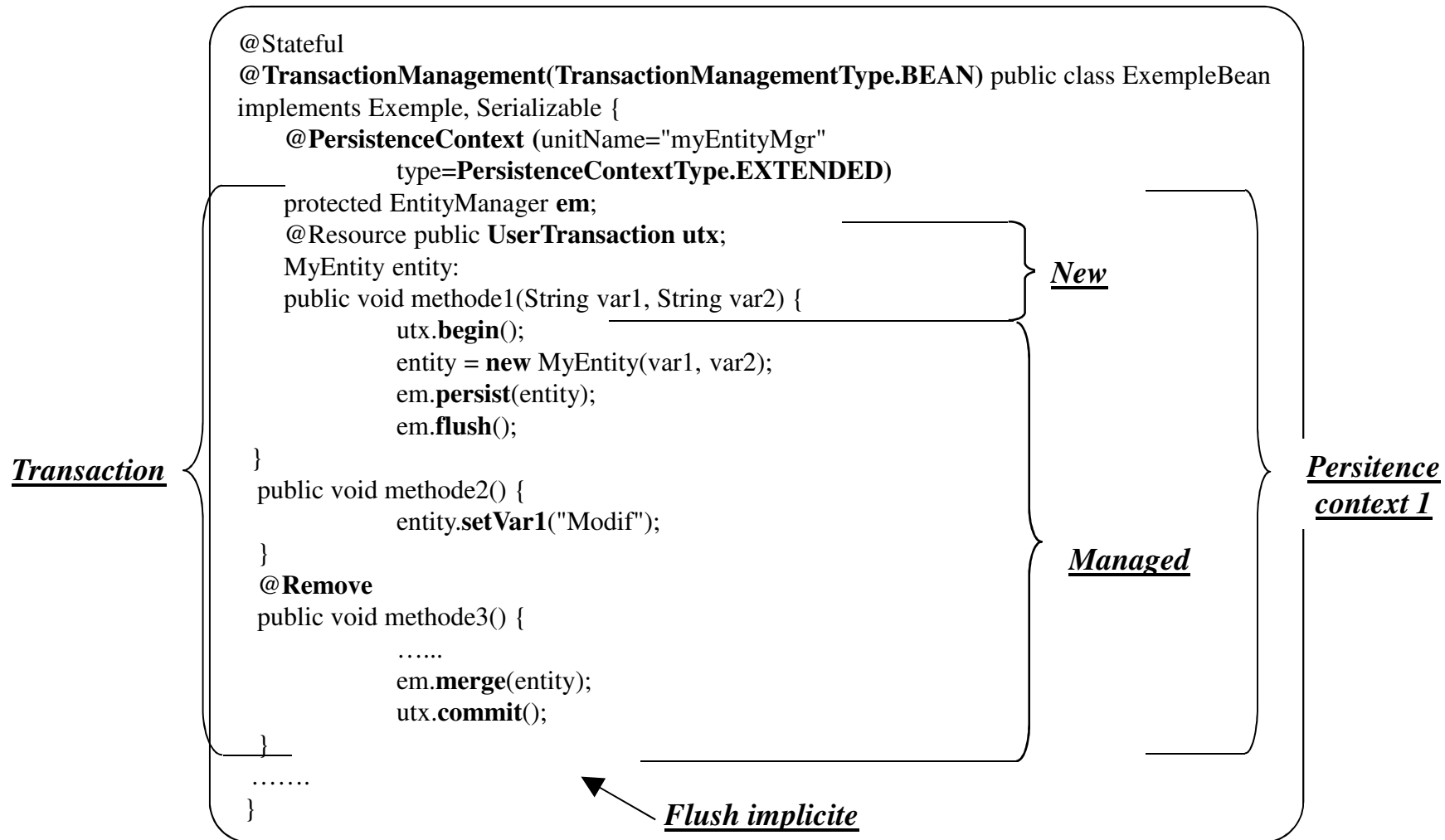




Le Persitence Context (2)



Le Persitence Context (3)





Les 'Callbacks'



Les "call-backs" (1)

Annotations destinées à solliciter certaines méthodes de l'EJB lors d'évènements spécifiques du cycle de vie d'un EJB Entity.

- **@PrePersist**
- **@PostPersist:**
 - *Sont définis pour les méthodes qui seront appelées avant et après la persistance de l'EJB*
- **@PreRemove:**
- **@PostRemove:**
 - *Sont définis pour les méthodes qui seront appelées avant et après la suppression de l'EJB*
- **@PreUpdate:**
- **@PostUpdate:**
 - *Sont définis pour les méthodes qui seront appelées avant et après la mise à jour de l'EJB*
- **@PostLoad:**
 - *Est défini pour une méthode qui sera appelée après le chargement de l'EJB.*



Les "call-backs" (2)

Les 'callbacks' dans l'EJB

```
@Entity
public class MyEntity implements Serializable {

    // ... ..
    @PostLoad
    public void controle() {
        // ... ..
    }
}
```

Les 'callbacks' dans un EntityListener

```
@Entity
@EntityListeners(MyEntityListener.class)
public class ExempleBean implements Exemple, Serializable {

    // ... ..
}
```

```
public class MyEntityListener {

    // ... ..
    @PostLoad
    public void controle(MyEntity entity) {
        // ... ..
    }
}
```



Les 'Verrous'



Optimistic Locking (1)

Code de l'EJB Entity

```
@Entity
@Table(name = "client")
public class EClient implements Serializable {

    private PkeyEm cp;
    private String donnees;
    private Integer version;
    @EmbeddedId
    public PKeyEm getCp() {return cp;}
    public void setCp(PKeyEm cp) {this.cp = cp;}
    public EClient(PKeyEm cp, Integer region, String donnees) {
        setCp(cp);
        setDonnees(donnees);
    }
    .....
    @Version
    @Column(name = "verrou")
    public Integer getVersion() {return version;}
    public void setVersion(Integer version) {this.version = version;}
}
```

Optimistic Locking (2)

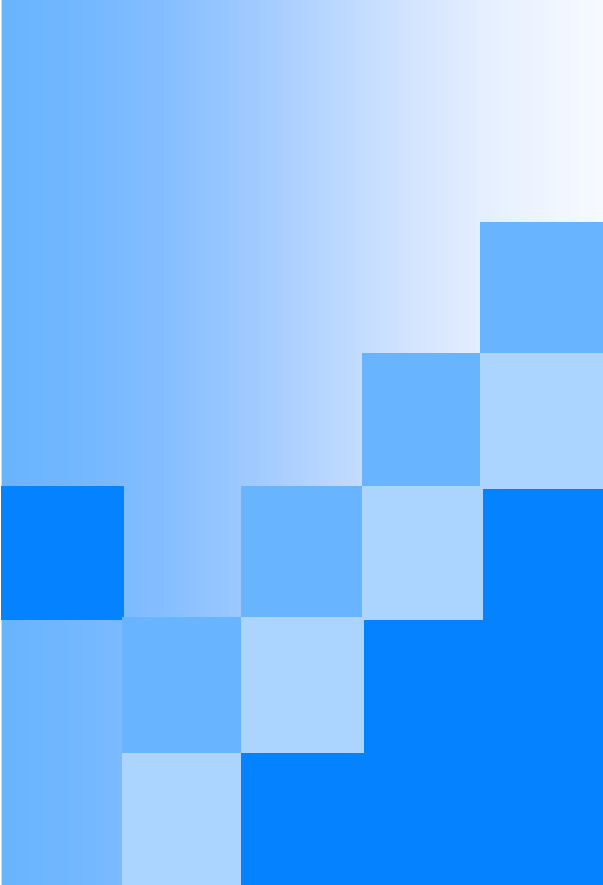
Code de l'EJB Session

```
@PersistenceContext (unitName="myEntityMgr")
protected EntityManager em;
EClient client = null;

.....

public void premiereMethode (String noCli, Integer region) {
    PKeyEm cp = new PKeyEm(noCli, region);
    client = em.find(EClient.class, cp);
    .....
}

//Entre les deux méthodes quelqu'un d'autre a pu modifier le contenu de l'EJB
@Remove
public void derniereMethode (String donnees) {
    client.setDonnees(donnees);
    try {
        ....
        em.merge(client);
        em.flush(); —————> (obligatoire si on veut capturer l'exception)
    } catch (RuntimeException e){
        if (StaleObjectStateException.class.isInstance(e)) {
            throw new TraitementException("...");
            .....
            ou OptimisticLockException
        }
    }
}
```



Les 'Clé Primaires Composite'



Clé Primaire Composite (1)

Code d'une Clé Primaire

```
public class PKey implements Serializable {  
  
    public Pkey() { }  
    public PKey(String noCli, Integer region) {  
        setNoCli(noCli);  
        setRegion(region);  
    }  
    private String noCli;  
    private Integer region;  
    public String getNoCli() { .... }  
    public void setNoCli(String noCli) { .... }  
    public Integer getRegion() { .... }  
    public void setRegion(Integer region) { .... }  
    public boolean equals(Object autre) {  
        if (autre instanceof Pkey) {  
            return (noCli.equals( ((PKey)autre).noCli) &&  
                region.equals( ((PKey)autre).region);  
        }  
        return false;  
    }  
    public int hashCode() {  
        return noCli.concat(region.toString()).hashCode();  
    }  
}
```



Clé Primaire Composite (2)

Code de l'entity utilisant la clé

```
@Entity
@Table(name = "client")
@IdClass(PKey.class)
public class EClient implements Serializable {

    private String noCli;
    private Integer region;
    private String donnees;
    public String getNoCli() {...}
    public void setNoCli(String noCli) {...}
    public Integer getRegion() {...}
    public void setRegion(Integer region) {...}
    public String getDonnees() {...}
    public void setDonnees(String donnees) {...}
    public EClient(String noCli, Integer region, String donnees) {
        setNoCli(noCli);
        setRegion(region);
        setDonnees(donnees);
    }
    .....
}
```

(Idem à ceux de Pkey)

L'appel par le client

```
....
EClient client = new EClient(noCli, region, donnees);
....
```



Clé Primaire Composite Embeddable (1)

Code d'une Clé Primaire Embeddable

@Embeddable

```
public class PKeyEm implements Serializable {

    public PKeyEm() { }
    public PKeyEm(String noCli, Integer region) {
        setNoCli(noCli);
        setRegion(region);
    }
    private String noCli;
    private Integer region;
    public String getNoCli() {...}
    public void setNoCli(String noCli) {...}
    public Integer getRegion() {...}
    public void setRegion(Integer region) {...}
    public boolean equals(Object autre) {
        if (autre instanceof PKeyEm) {
            return (noCli.equals( ((PKeyEm) autre).noCli) &&
                region.equals( ((PKeyEm) autre).region));
        }
        return false;
    }
    public int hashCode() {
        return noCli.concat(region.toString()).hashCode();
    }
}
```



Clé Primaire Composite Embeddable (2)

Code de l'entité
utilisant la clé

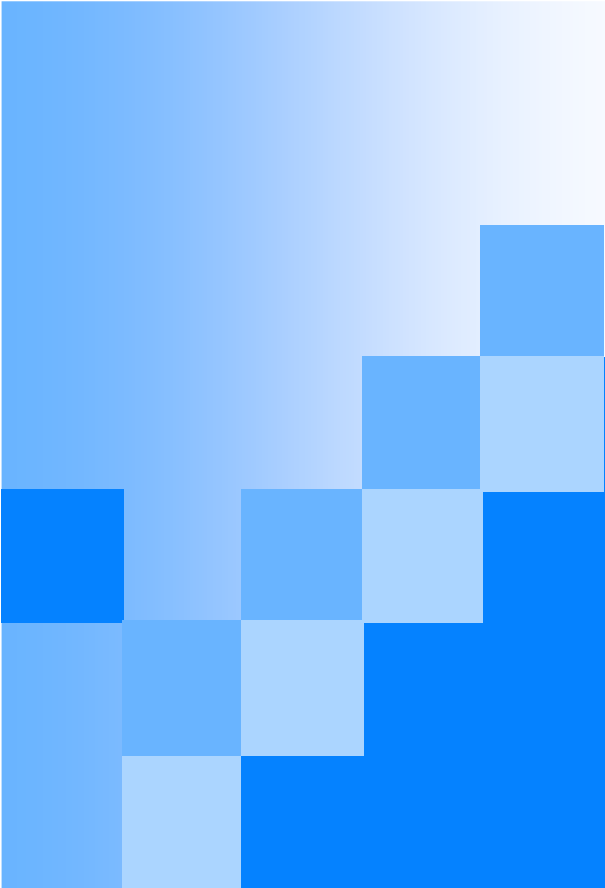
```
@Entity
@Table(name = "client")
public class EClient implements Serializable {

    private PkeyEm cp;
    private String donnees;
    @EmbeddedId
    public PKeyEm getCp() {return cp;}
    public void setCp(PKeyEm cp) {this.cp = cp;}

    public EClient(PKeyEm cp, Integer region, String donnees) {
        setCp(cp);
        setDonnees(donnees);
    }
    .....
}
```

L'appel par le client

```
PKeyEm cp = new PKeyEm(noCli, region);
EClient client = new EClient(cp, region, donnees);
.....
```



L'Héritage entre *EJBs Entity*

L'Héritage: "une table par hiérarchie" (1)

```
@Entity
@Table(name="Client")
@Inheritance(strategy=
    InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name="disc",
discriminatorType=STRING,length=3)
@DiscriminatorValue("Cli")
public class EClient {
    protected String noCompte;
    protected String adresse;
    public EClient() {}
    public EClient(String noCompte, String adresse)
    {
        setNoCompte(noCompte);
        setAdresse(adresse);
    }
    @Id
    public String getNoCompte() {.....}
    public void setNoCompte(String noCompte)
    {.....}
    public String getAdresse() {.....}
    public void setAdresse(String adresse) {.....}
}
```

```
@Entity
@Inheritance( discriminatorValue = "Cpt")
public class ECompte extends EClient {
    protected Float solde;
    public ECompte() {}
    public ECompte(String noCompte, String adresse,
        Float solde ) {
        super(noCompte, adresse);
        setSolde(solde);
    }
    public Float getSolde() {.....}
    public void setSolde(Float solde) {.....}
}
```

```
@Entity
@Inheritance( discriminatorValue = "Crd")
public class ECredit extends EClient {
    protected Float mensualite;
    public ECredit() {}
    public ECredit(String noCompte, String adresse,
        Float mensualite ) {
        super(noCompte, adresse);
        setMensualite(mensualite);
    }
    public Float getMensualite() {.....}
    public void setMensualite(Float mensualite) {..}
}
```



L'Héritage: "une table par hiérarchie" (2)

```
@PersistenceContext (unitName="myEntityMgr")
protected EntityManager em;

.....
    EClient client = new EClient("0001", "28 rue des Chataigners...");
    ECompte compte = new ECompte("0002", "28 rue des Chataigners...",
                                new Float(0.0));
    ECredit credit = new ECredit("0003", "28, rue des Chataigners..."
                                new Float(115.25));

    em.persist(client);
    em.persist(compte);
    em.persist(credit);

.....
```

Table: Client

nocompte	disc	adresse	solde	mensualite
0001	Cli	28, rue des Chataigners...		
0002	Cpt	28, rue des Chataigners...	0.0	
0003	Crd	28, rue des Chataigners...		115.25

L'Héritage: "une table par classe" (1)

```
@Entity
@Table(name="Client")
@Inheritance(strategy=
    InheritanceType.TABLE_PER_CLASS)
public class EClient {
    protected String noCompte;
    protected String adresse;
    public EClient() {}
    public EClient(String noCompte, String adresse)
    {
        setNoCompte(noCompte);
        setAdresse(adresse);
    }
    @Id
    public String getNoCompte() {.....}
    public void setNoCompte(String noCompte)
    {.....}
    public String getAdresse() {.....}
    public void setAdresse(String adresse) {.....}
}
```

```
@Entity
@Table(name="Compte")
public class ECompte extends EClient {
    protected Float solde;
    public ECompte() {}
    public ECompte(String noCompte, String adresse,
        Float solde ) {
        super(noCompte, adresse);
        setSolde(solde);
    }
    public Float getSolde() {.....}
    public void setSolde(Float solde) {.....}
}
```

```
@Entity
@Table(name="Credit")
public class ECredit extends EClient {
    protected Float mensualite;
    public ECredit() {}
    public ECredit(String noCompte, String adresse,
        Float mensualite ) {
        super(noCompte, adresse);
        setMensualite(mensualite);
    }
    public Float getMensualite() {.....}
    public void setMensualite(Float mensualite) {..}
}
```



L'Héritage: "une table par classe" (2)

```
@PersistenceContext (unitName="myEntityMgr")
protected EntityManager em;

.....
EClient client = new EClient("0001", "28 rue des Chataigners...");
ECompte compte = new ECompte("0002", "28 rue des Chataigners...",
                             new Float(0.0));
ECredit credit = new ECredit("0003", "28, rue des Chataigners..."
                             new Float(115.25));

em.persist(client);
em.persist(compte);
em.persist(credit);

.....
```

Table: Client

nocompte	adresse
0001	28, rue des Chataigners...

Table: Compte

nocompte	adresse	solde
0002	28, rue des Chataigners...	0.0

Table: Credit

nocompte	adresse	mensualite
0003	28, rue des Chataigners...	115.25

L'Héritage: "par jointure" (1)

```
@Entity
@Table(name="Client")
@Inheritance(strategy=
    InheritanceType. JOINED)
public class EClient {
    protected String noCli;
    protected String adresse;
    public EClient() {}
    public EClient(String noCli, String adresse) {
        setNoCli(noCli);
        setAdresse(adresse);
    }
    @Id
    public String getNoCli() { ..... }
    public void setNoCli(String noCli) { ..... }
    public String getAdresse() { ..... }
    public void setAdresse(String adresse) { ..... }
}
```

```
@Entity
@Table(name="Compte")
@PrimaryKeyJoinColumn(name="nocompte",
    referencedColumnName="nocli")
public class ECompte extends EClient {
    protected Float solde;
    public ECompte() {}
    public ECompte(String noCompte, String adresse,
        Float solde ) {
        .....
    }
    public Float getSolde() { ..... }
    public void setSolde(Float solde) { ..... }
}
```

```
@Entity
@Table(name="Credit")
@PrimaryKeyJoinColumn(name="nocredit",
    referencedColumnName="nocli")
public class ECredit extends EClient {
    protected Float mensualite;
    public ECredit() {}
    public ECredit(String noCompte, String adresse,
        Float mensualite ) {
        .....
    }
    public Float getMensualite() { ..... }
    public void setMensualite(Float mensualite) { .. }
}
```



L'Héritage: "une table par classe" (2)

```
@PersistenceContext (unitName="myEntityMgr")
protected EntityManager em;

.....
EClient client = new EClient("0001", "28 rue des Chataigners...");
ECompte compte = new ECompte("0001", "28 rue des Chataigners...",
                             new Float(0.0));
ECredit credit = new ECredit("0001", "28, rue des Chataigners..."
                             new Float(115.25));

em.persist(client);
em.persist(compte);
em.persist(credit);

.....
```

Table: Client

nocli	adresse
0001	28, rue des Chataigners...

Table: Credit

nocredit	mensualite
0001	115.25

Table: Compte

nocompte	solde
0001	0.0



Les Relations *entre EJBs*

Relations unidirectionnelles 'One to One'

```
@Entity
@Table(name="TABLEP")
public class EnrParent Serializable {
    private int idP;
    private EnrFils enrFils;
    private String varP;
    .....
    @Id(generate = GenerationType.Auto)
    @Column(name = "IDP", nullable =false)
    public int getIdP() { ..... }
    @OneToOne(targetEntity=EnrFils
               cascade ={ CascadeType.PERSIST, CascadeType.MERGE}
               fetch = FetchType.EAGER)
    @JoinColumn(name="IDP", referencedColumnName="IDF")
    public EnrFils getEnrFils() { ..... }
    public void setEnrFils (EnrFils enrFils) { ..... }
    .....
}
```

```
@Entity
@Table(name="TABLEF")
public class EnrFils Serializable {
    private int idF;
    private String varF;
    .....
    @Id(generate = GenerationType.Auto)
    @Column(name = "IDF", nullable =false)
    public int getIdF() { ..... }
    public String getVarF() { ..... }
    public void setVarF(String varF) { ..... }
    .....
}
```


Relations bidirectionnelles 'One to One' (1)

```
@Entity
@Table(name="TABLEP")
public class EnrParent Serializable {
    private int idP;
    private EnrFils enrFils;
    private String varP;

    .....

    @Id(generate = GenerationType.Auto)
    @Column(name = "IDP", nullable =false)
    public int getIdP() { ..... }
    @OneToOne(targetEntity=EnrFils)
    @JoinColumn(name="IDP", referencedColumnName="IDF")
    public EnrFils getEnrFils() { ..... }
    public void setEnrFils (EnrFils enrFils) { ..... }

    .....
}
```

```
@Entity
@Table(name="TABLEF")
public class EnrFils Serializable {
    private int idF;
    private EnrParent enrParent;
    private String varF;

    .....

    @Id(generate = GenerationType.Auto)
    @Column(name = "IDF", nullable =false)
    public int getIdF() { ..... }
    @OneToOne(targetEntity=EnrParent)
    @JoinColumn(name="IDF", referencedColumnName="IDP")
    public EnrParents getEnrParent() { ..... }
    public void setEnrParent (EnrParent enrParent) { ..... }

    .....
}
```

Relations bidirectionnelles 'One to One' (2)

@Entity

```
public class Parent Serializable {
    private int idP;
    private Fils monFils;
    private String varP;

    .....

    @Id(generate = GenerationType.Auto)
    public int getIdP() {.....}
    @OneToOne
    public Fils getMonFils() {.....}
    public void setMonFils (Fils monFils) {...}

    .....
}
```

@Entity

```
public class Fils Serializable {
    private int idF;
    private Parent monPere;
    private String varF;

    .....

    @Id(generate = GenerationType.Auto)
    public int getIdF() {.....}
    @OneToOne(mappedBy=monFils)
    public Parent getmonPere() {.....}
    public void setMonPere (Parent monPere) {...}

    .....
}
```

Table: Parent

idp	varp	monfils_idf
0002	"AAAA"	0001

Table: Fils

idf	varf
0001	"BBBB"

"Contrainte de clé unique"

Relations unidirectionnelles 'One to Many' (1)

```
@Entity
@Table(name="TABLEP")
public class EnrParent Serializable {
    private int idP;
    private Collection <EnrFils> enrFils;
    private String varP;

    .....

    @Id(generate = GenerationType.Auto)
    @Column(name = "IDP", nullable =false)
    public int getIdP() { .....}
    @OneToMany(targetEntity=EnrFils, cascade=ALL)
    @JoinColumn(name="IDP", referencedColumnName="IDF")
    public Collection <EnrFils> getEnrFils { .....}
    public void setEnrFils (Collection <EnrFils> enrFils) { .....}

    .....
}
```

```
@Entity
@Table(name="TABLEF")
public class EnrFils Serializable {
    .....
}
```

```
@Entity
@Table(name="TABLEF")
public class EnrFils Serializable {
    .....
}
```

Relations unidirectionnelles 'One to Many' (2)

```
@Entity
public class Parent Serializable {
    private int idP;
    private Collection <Fils> mesFils;
    private String varP;
    .....
    @Id(generate = GenerationType.Auto)
    @Column(name = "IDP", nullable =false)
    public int getIdP() { ..... }
    @OneToMany
    public Collection <Fils> getMesFils { ..... }
    public void setMesFils (Collection <Fils> mesFils) { ... }
    .....
}
```

```
@Entity
public class Fils Serializable {
    private int idF;
    private String varF;
    .....
    @Id(generate = GenerationType.Auto)
    public int getIdF() { ..... }
    .....
}
```

```
@Entity
public class Fils Serializable {
    private int idF;
    private String varF;
    .....
    @Id(generate = GenerationType.Auto)
    public int getIdF() { ..... }
    .....
}
```

Table: Parent

idp	varp
0002	"AAAA"

Table de jointure: Parent_Fils

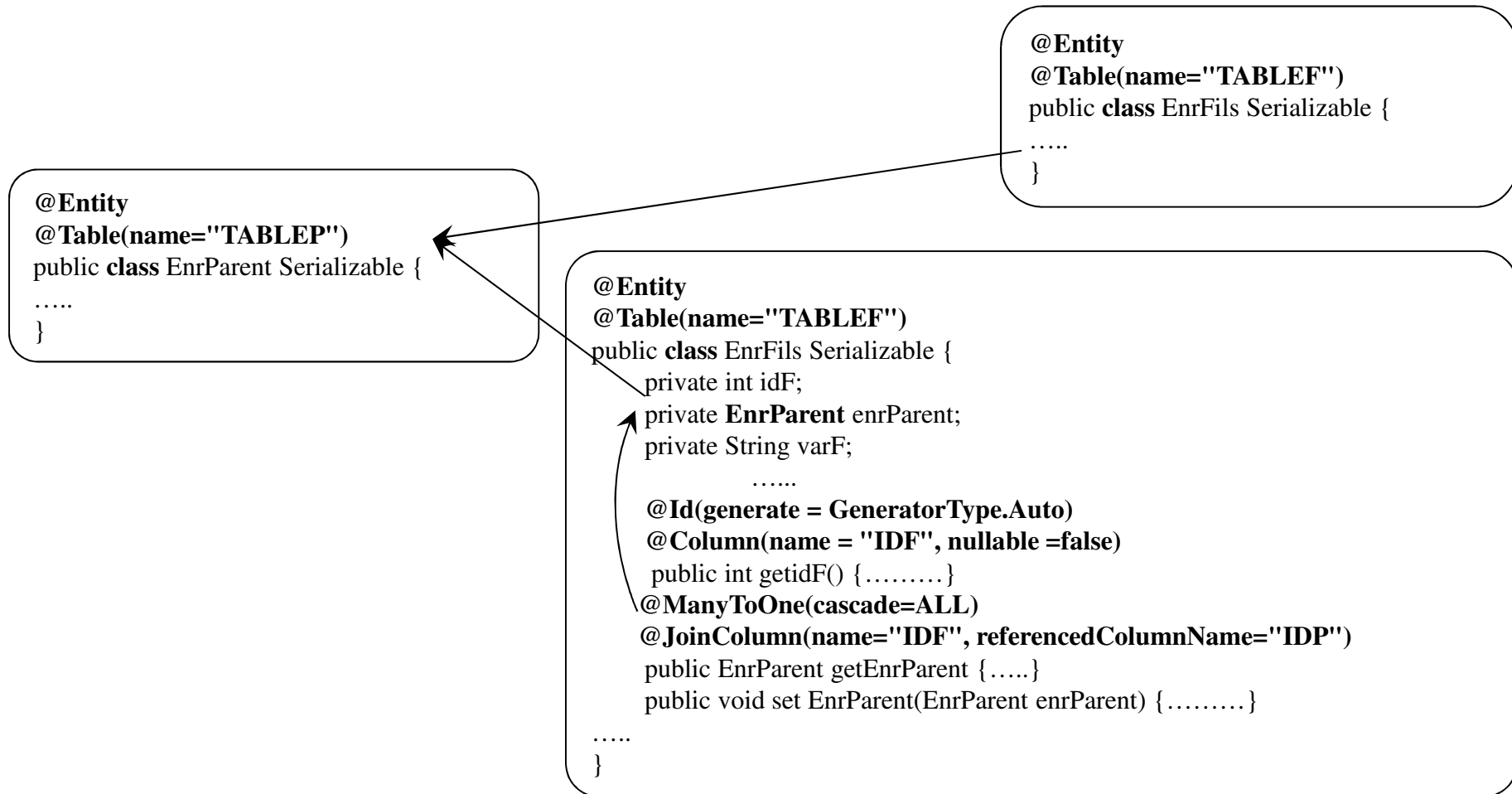
parents_idp	fils_idf
0002	0001
0002	0002

Table: Fils

idf	varf
0001	"BBBB"
0002	"CCCC"



Les relations unidirectionnelle 'Many to One' (1)



Les relations unidirectionnelle 'Many to One' (2)

```
@Entity
public class Fils Serializable {
    ....
}
```

```
@Entity
public class Parent Serializable {
    ....
}
```

```
@Entity
public class EnrFils Serializable {
    private int idF;
    private Parent monPere;
    private String varF;
    ....

    @Id(generate = GenerationType.Auto)
    @Column(name = "IDF", nullable = false)
    public int getIdF() { ..... }

    @ManyToOne(cascade = ALL)
    public Parent getMonPere() { ..... }
    public void setMonPere(Parent monPere) { ..... }
    ....
}
```

Table: Parent

idp	varp
0002	"AAAA"

Table: Fils

monpere_idp	idf	varf
0002	0001	"BBBB"
0002	0002	"CCCC"

Les relations 'One to Many-Many to One'

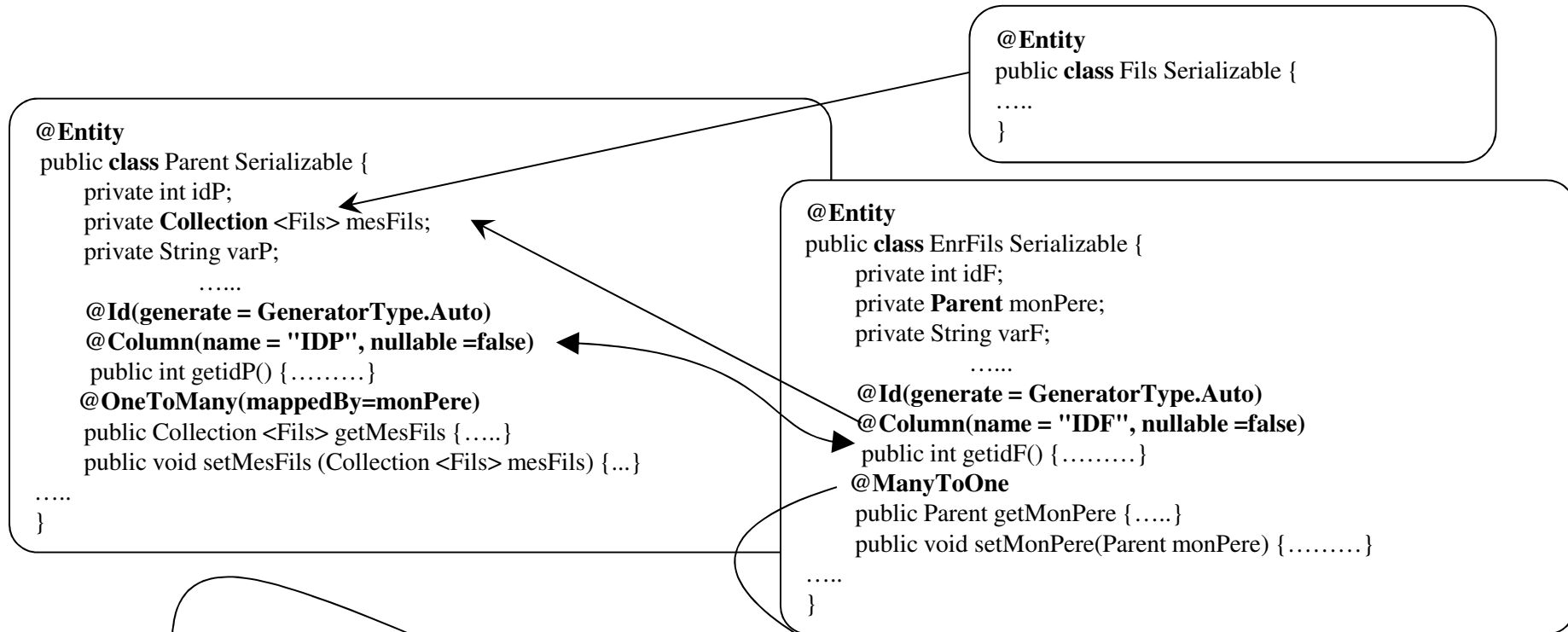


Table: Parent

idp	varp
0002	"AAAA"

Table: Fils

monpere_idp	idf	varf
0002	0001	"BBBB"
0002	0002	"CCCC"

Les relations bidirectionnelles 'Many to Many'

```
@Entity
public class Parent Serializable {
    private int idP;
    private Collection<Fils> mesFils;
    private String varP;
    .....
    @Id(generate = GenerationType.Auto)
    @Column(name = "IDP", nullable =false)
    public int getidP() { ..... }
    @ManyToMany(mappedBy=mesParents)
    public Collection<Fils> getMesFils { ..... }
    public void setMesFils (Collection<Fils> mesFils) { ... }
    .....
}
```

```
@Entity
public class EnrFils Serializable {
    private int idF;
    private Collection<Parent> mesParents;
    private String varF;
    .....
    @Id(generate = GenerationType.Auto)
    @Column(name = "IDF", nullable =false)
    public int getidF() { ..... }
    @ManyToMany
    public Collection<Parent> getMesParents { ..... }
    public void setMesParents(Collection<Parent>
        mesParents) { ..... }
    .....
}
```

Table de jointure: mesParents_mesFils

Table: Parent

idp	varp
0002	"AAAA"
0003	"EEEE"

mesparents_idp	mesfils_idf
0002	0001
0003	0001
0002	0002
0003	0002

Table: Fils

idf	varf
0001	"BBBB"
0002	"CCCC"



EJB-QL



Les méthodes de recherche de l'EntityManager (1)

```
@PersistenceContext (unitName="myEntityMgr")
protected EntityManager em;
....
MyEntity retEntity = em.find(MyEntity.class, Integer.valueOf(clePrimaire));
```

```
@PersistenceContext (unitName="myEntityMgr")
protected EntityManager em;
....
Query qry = em.createQuery("SELECT avg(m.age) FROM MyEntity m");
Number ageMoyen = (Number) qry.getSingleResult();
```

```
@PersistenceContext (unitName="myEntityMgr")
protected EntityManager em;
....
Query qry = em.createQuery("SELECT m FROM MyEntity m WHERE m.age >= 5 AND m.age <= 8");
Collection<MyEntity> col = (Collection<MyEntity>) qry.getResultList();
```



Les méthodes de recherche de l'EntityManager (2)

```
@PersistenceContext (unitName="myEntityMgr")
```

```
protected EntityManager em;
```

```
....
```

```
Query qry = em.createQuery("SELECT m FROM MyEntity m WHERE m.age >= :bas AND m.age <= :haut ");
```

```
qry = qry.setParameter ("bas", 5);
```

```
qry = qry.setParameter ("haut",8);
```

```
Collection<MyEntity> col = (Collection<MyEntity>) qry.getResultList();
```

// ou bien :

```
Collection <MyEntity> col = (Collection<MyEntity>) em.createQuery("SELECT m"+  
"FROM MyEntity m WHERE m.age >= :bas AND m.age <= :haut ") .  
    setParameter ("bas", 5).  
    setParameter ("haut",8).getResultList();
```

```
@PersistenceContext (unitName="myEntityMgr")
```

```
protected EntityManager em;
```

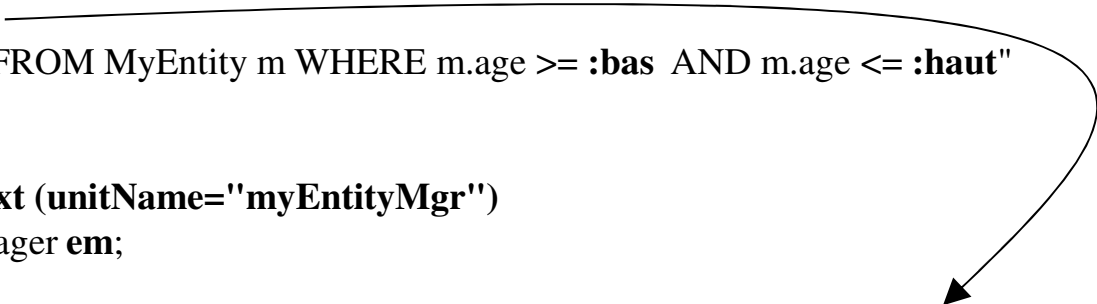
```
....
```

```
Collection <MyEntity> col = (Collection<MyEntity>) em.createQuery("SELECT m"+  
"FROM MyEntity m WHERE m.age >= ?1 AND m.age <= ?2 ") .  
    setParameter (1, 5).  
    setParameter (2,8).getResultList();
```



Les méthodes de recherche de l'EntityManager (3)

```
@NamedQuery(  
    name="recherche",  
    query="SELECT m FROM MyEntity m WHERE m.age >= :bas AND m.age <= :haut"  
)  
  
@PersistenceContext (unitName="myEntityMgr")  
protected EntityManager em;  
....  
Collection <MyEntity> col = (Collection<MyEntity>) em.createNamedQuery(" recherche ") .  
    setParameter ("bas", 5) .  
    setParameter ("haut", 8)) .getResultList();
```

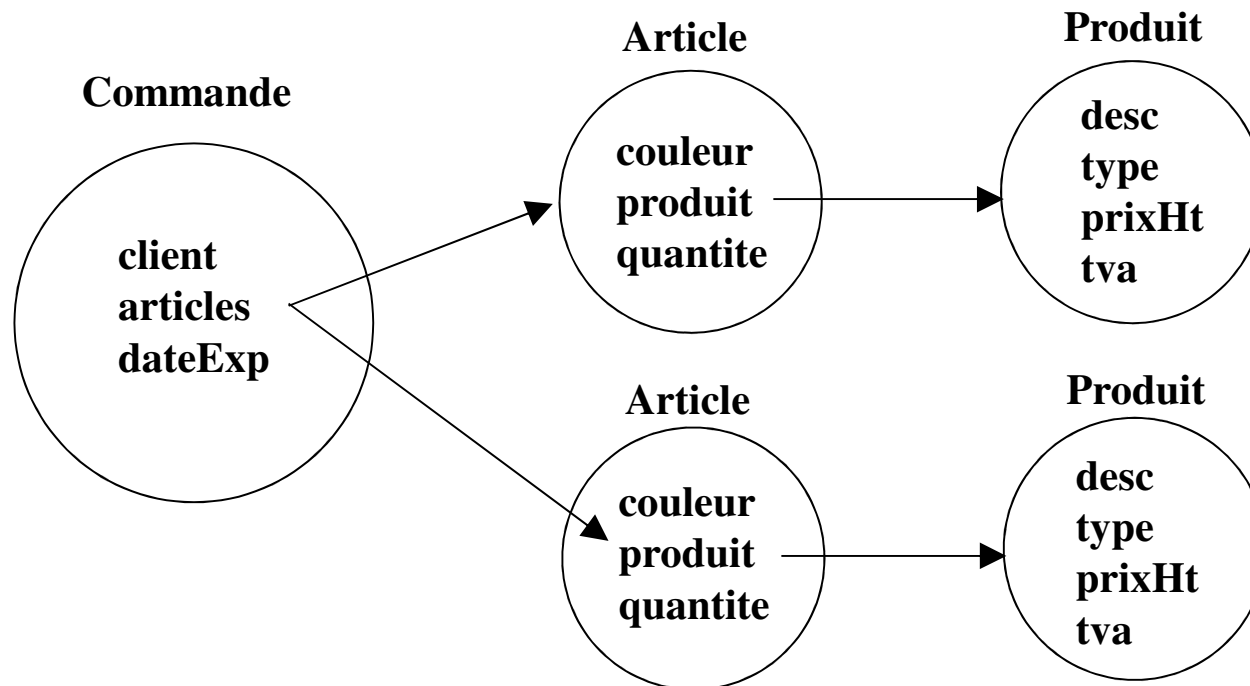


```
@PersistenceContext (unitName="myEntityMgr")  
protected EntityManager em;  
....  
Collection <MyEntity> col = (Collection<MyEntity>) em.createNativeQuery("SELECT iD, age"+  
    "FROM Mytable ", MyEntity.class) .getResultList();
```



Traversée de relations (1)

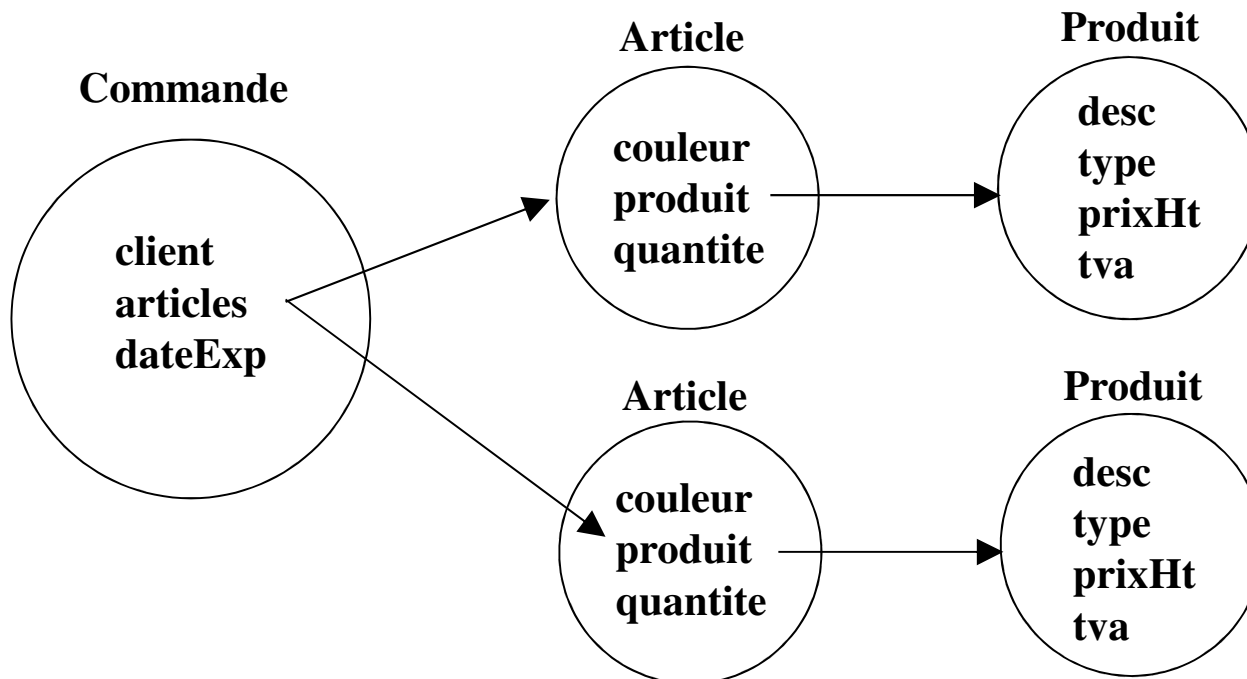
```
@PersistenceContext (unitName="myEntityManager")
protected EntityManager em;
....
Query qry = em.createQuery("SELECT DISTINCT cm FROM Commande cm"+
                           "JOIN cm.articles ar JOIN ar.produit pr"+
                           "WHERE pr.type = :ptype");
qry.setParameter("ptype", "Bureautique");
List<Commande> commandes = (List<Commande>) q.getResultList();
```





Traversée de relations (2)

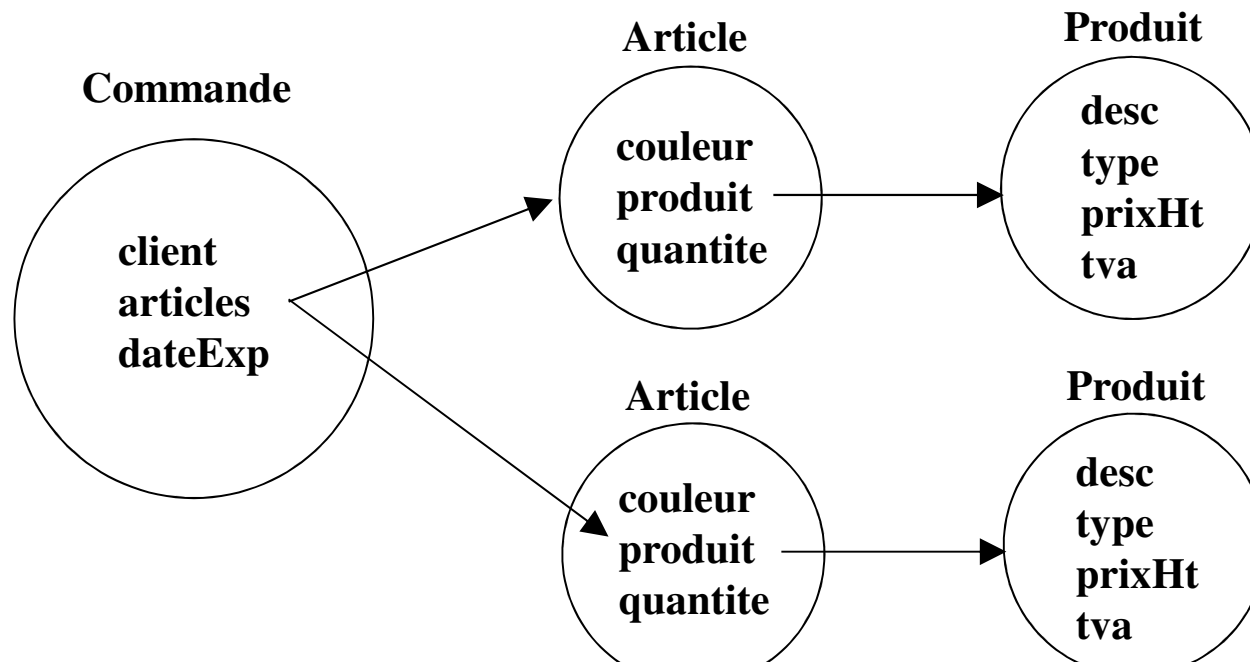
```
@PersistenceContext (unitName="myEntityManager")
protected EntityManager em;
....
Query qry = em.createQuery("SELECT DISTINCT cm FROM Commande cm"+
                           "IN (cm.articles) ar"+
                           "WHERE ar.produit.type = :ptype");
qry.setParameter("ptype", "Bureautique");
List<Commande> commandes = (List<Commande>) q.getResultList();
```



Mises à Jour massives

```
@PersistenceContext (unitName="myEntityManager")
protected EntityManager em;
....
Query qry = em.createQuery("UPDATE Produit pr SET pr.tva = :taxe");
    qry.setParameter("taxe", 5.5);

int Cnt = qry.executeUpdate();
```

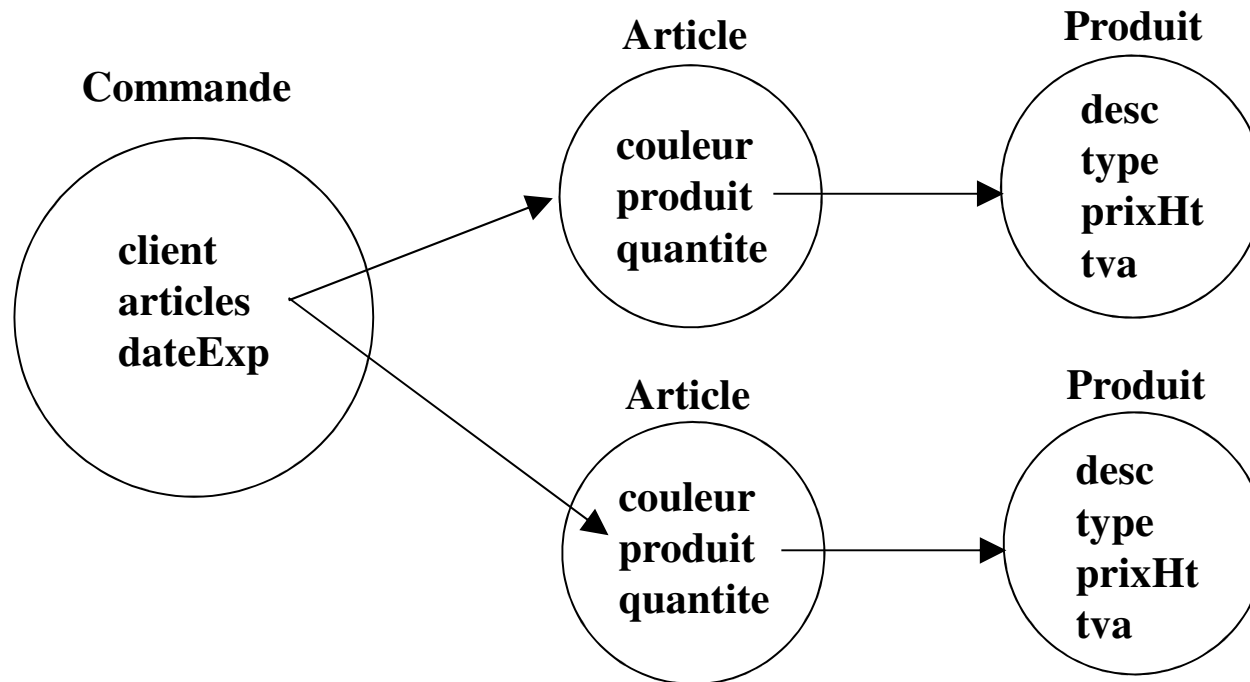




Suppressions massives

```
@PersistenceContext (unitName="myEntityMgr")
protected EntityManager em;
....

int Cnt = qry.executeUpdate();
Query qry = em.createQuery("DELETE FROM Commande cm WHERE cm.dateExp != null");
int Cnt = q.executeUpdate();
```





Les Message Driven Bean(s)



Principe

