



Expert  
EXPERT

# JPA et Java Hibernate

Apprenez le mapping  
objet-relationnel (ORM)  
avec Java

Fichiers complémentaires  
à télécharger



Martial BANON

Les éléments à télécharger sont disponibles à l'adresse suivante :  
**<http://www.editions-eni.fr>**  
Saisissez la référence de l'ouvrage **EIJHJPA** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

**Avant-propos**

- 1. Introduction ..... 15
- 2. Contenu de l'ouvrage ..... 16

**Chapitre 1**  
**Environnement de développement**

- 1. Installation du JDK ..... 17
- 2. Installation de NetBeans ..... 19
- 3. Installation d'Hibernate ..... 26
- 4. Installation de MySQL ..... 27
- 5. Premier lancement de NetBeans ..... 48
  - 5.1 Mise en place des libraires ..... 49
  - 5.2 Création du projet ..... 54

**Chapitre 2**  
**Concept des ORM**

- 1. Introduction ..... 57
- 2. Le concept ..... 58
  - 2.1 La norme : JPA ..... 59
  - 2.2 L'implémentation : Hibernate ..... 60
- 3. La structuration des données ..... 60
  - 3.1 Introduction au mapping ..... 60
  - 3.2 Importance du mapping ..... 61

# 2 \_\_\_\_\_ JPA et Java Hibernate

Apprenez le mapping objet-relationnel (ORM) avec Java

3.3	Différentes techniques de mapping . . . . .	61
4.	La connexion aux données . . . . .	63
4.1	Introduction aux sessions . . . . .	63
4.2	Les différents types de sessions . . . . .	64
4.3	L'utilisation d'une session. . . . .	64
4.3.1	RESOURCE_LOCAL avec Java SE . . . . .	64
4.3.2	RESOURCE_LOCAL avec Java EE . . . . .	65
4.3.3	JTA avec Java EE . . . . .	66
5.	L'interrogation des données . . . . .	67
5.1	Introduction aux requêtes . . . . .	67
5.2	Les requêtes natives . . . . .	68
5.3	Les requêtes objets . . . . .	68
5.4	Les requêtes nommées . . . . .	69
5.5	Les requêtes "implicites". . . . .	70
5.6	L'API Criteria. . . . .	71
6.	Le cycle de vie des données . . . . .	73
6.1	Introduction au cache . . . . .	73
6.2	Les différents niveaux de caches . . . . .	74
6.3	Cache de niveau 1 (L1). . . . .	75
6.4	Cache de niveau 2 (L2). . . . .	75
6.5	À retenir . . . . .	76

## Chapitre 3

### Préparation d'un projet

1.	Présentation du projet du livre . . . . .	77
1.1	Description . . . . .	77
1.2	Schéma de la base de données . . . . .	78
2.	Les possibilités de mapping . . . . .	78
2.1	Fichier de mapping . . . . .	79
2.2	Annotation . . . . .	80

3.	Paramétrage de l'ORM . . . . .	81
3.1	Création du fichier persistence.xml. . . . .	81
3.2	L'unité de persistance . . . . .	92
3.3	L'implémentation de JPA utilisée. . . . .	93
3.4	Les entités à mapper. . . . .	93
3.5	Les propriétés du fichier de persistance. . . . .	94
3.5.1	Le driver de connexion à la base de données . . . . .	94
3.5.2	L'URL de la base de données . . . . .	95
3.5.3	Le login . . . . .	95
3.5.4	Le mot de passe . . . . .	95
3.6	Le chargement du contexte de persistance . . . . .	96
4.	Les types persistables . . . . .	97
4.1	Classes définies par l'utilisateur. . . . .	97
4.1.1	Classe d'entité . . . . .	97
4.1.2	Superclasse mappée . . . . .	99
4.1.3	Classe intégrable . . . . .	100
4.2	Données Java simples . . . . .	101
4.3	Valeur multiple . . . . .	102
4.4	Divers . . . . .	103
5.	Les différents champs . . . . .	104
5.1	Les champs temporaires . . . . .	104
5.2	Les champs persistables . . . . .	105
5.2.1	L'annotation @Basic . . . . .	105
5.2.2	L'annotation @Column . . . . .	107
5.2.3	L'annotation @Temporal . . . . .	109
6.	Les clés primaires . . . . .	111
6.1	Clé primaire simple. . . . .	111
6.2	Clé primaire composée . . . . .	112
7.	Les valeurs générées . . . . .	113
7.1	Valeur fixe . . . . .	113
7.2	Valeur incrémentielle . . . . .	114
7.2.1	GenerationType.AUTO . . . . .	114

# 4 JPA et Java Hibernate

Apprenez le mapping objet-relationnel (ORM) avec Java

7.2.2	GenerationType.IDENTITY	115
7.2.3	GenerationType.TABLE	115
7.2.4	GenerationType.SEQUENCE	116
8.	Les relations	117
8.1	Les relations 1-1 (OneToOne)	117
8.2	Les relations n-1 (ManyToOne)	120
8.3	Les relations 1-n (OneToMany)	120
8.4	Les relations n-n (ManyToMany)	122
8.5	Les relations n-n avec données sur la jointure	126
9.	Type de chargement des données	129
9.1	Lazy	129
9.2	Eager	130
9.3	Mise en place	130
10.	Particularités sur les entités	131
10.1	Sérialisation	131
10.2	Equals et hashCode	132
10.2.1	Implémentation par défaut	132
10.2.2	Implémentation basée sur la clé primaire	132
10.2.3	Implémentation basée sur une clé métier	133
10.2.4	Conclusion	133

## Chapitre 4

### Manipulation des données

1.	Préparation	135
2.	Établissement de la connexion	136
2.1	EntityManagerFactory	136
2.1.1	Avec conteneur JEE, en RESOURCE_LOCAL	136
2.1.2	Sans conteneur JEE, en RESOURCE_LOCAL	137
2.2	EntityManager	144
2.2.1	Avec conteneur JEE, en JTA	144
2.2.2	Avec conteneur JEE, en RESOURCE_LOCAL	145

2.2.3	Sans conteneur JEE, en RESOURCE_LOCAL . . . . .	145
2.3	État d'une entité . . . . .	147
3.	Les transactions . . . . .	147
3.1	Exécution d'une transaction . . . . .	148
3.2	Méthodes diverses sur les transactions . . . . .	149
3.2.1	Vérifier que la transaction est active . . . . .	149
3.2.2	Empêcher la modification via la transaction . . . . .	150
4.	Création d'une entité . . . . .	152
4.1	Création d'une entité simple . . . . .	152
4.2	Création d'une arborescence d'entités . . . . .	154
4.2.1	Relation avec identifiant différent . . . . .	154
4.2.2	Relation avec identifiant partagé . . . . .	157
4.2.3	L'annotation @PrePersist . . . . .	160
4.2.4	Le générateur générique d'Hibernate . . . . .	162
4.3	Cascade avec une table de jointure . . . . .	164
4.4	Création d'une entité avec clé primaire composée . . . . .	167
5.	Récupération d'une entité . . . . .	169
5.1	Entité avec une clé primaire simple . . . . .	170
5.2	Entité avec une clé primaire composée . . . . .	171
5.3	Depuis une entité déjà chargée . . . . .	171
5.3.1	FetchType.EAGER . . . . .	172
5.3.2	FetchType.LAZY . . . . .	173
5.4	Depuis une requête spécifique . . . . .	175
5.5	Référence d'une entité . . . . .	176
5.5.1	Chargement différé . . . . .	176
6.	Suppression d'une entité . . . . .	178
6.1	Suppression simple . . . . .	179
6.2	Suppression en cascade . . . . .	179
6.3	Suppression d'une relation . . . . .	182

# 6 \_\_\_\_\_ JPA et Java Hibernate

Apprenez le mapping objet-relationnel (ORM) avec Java

7.	Modification d'une entité. . . . .	184
7.1	Modification des champs d'une entité . . . . .	184
7.1.1	Depuis une entité managée. . . . .	185
7.1.2	Depuis une entité non managée. . . . .	185
7.2	Modification des relations d'une entité. . . . .	186
7.2.1	Cas général. . . . .	186
7.2.2	Cas particulier : @ManyToMany . . . . .	187
7.3	Cas particuliers . . . . .	189
7.3.1	Entité managée supprimée . . . . .	189
7.3.2	Suite de modification d'une entité non managée . . . . .	190
8.	Synchronisation . . . . .	193
8.1	De l'ORM vers la base de données. . . . .	193
8.2	De la base de données vers l'entité . . . . .	194

## Chapitre 5

### Requêtes : les langages JPQL et HQL

1.	Introduction . . . . .	197
2.	Généralités . . . . .	198
3.	Types de requêtes . . . . .	198
3.1	SELECT . . . . .	199
3.2	UPDATE . . . . .	199
3.3	DELETE. . . . .	200
3.4	INSERT . . . . .	200
4.	La clause SELECT . . . . .	201
5.	La clause FROM . . . . .	202
5.1	Variables d'identification . . . . .	202
5.2	Référence à l'entité root . . . . .	202
5.3	Les jointures . . . . .	203
6.	La clause WHERE . . . . .	205

- 7. Les expressions ..... 206
  - 7.1 Alias ..... 206
  - 7.2 PATH ..... 206
  - 7.3 Paramètres ..... 206
  - 7.4 Littéral ..... 207
    - 7.4.1 Le littéral NULL ..... 207
    - 7.4.2 Le littéral booléen ..... 208
    - 7.4.3 Le littéral numérique ..... 208
    - 7.4.4 Le littéral string ..... 209
    - 7.4.5 Le littéral date ..... 209
    - 7.4.6 Le littéral énum ..... 209
    - 7.4.7 Le littéral d'entité ..... 209
  - 7.5 Type d'entité ..... 210
  - 7.6 Arithmétique ..... 210
  - 7.7 Fonctions d'agrégations ..... 211
  - 7.8 Fonctions scalaires ..... 211
    - 7.8.1 CONCAT ..... 212
    - 7.8.2 SUBSTRING ..... 212
    - 7.8.3 UPPER ..... 212
    - 7.8.4 LOWER ..... 212
    - 7.8.5 TRIM ..... 213
    - 7.8.6 LENGTH ..... 213
    - 7.8.7 LOCATE ..... 213
    - 7.8.8 ABS ..... 214
    - 7.8.9 MOD ..... 214
    - 7.8.10 SQRT ..... 214
    - 7.8.11 CURRENT\_DATE ..... 214
    - 7.8.12 CURRENT\_TIME ..... 215
    - 7.8.13 CURRENT\_TIMESTAMP ..... 215
  - 7.9 Collections ..... 215
    - 7.9.1 Vérifier qu'une liste est vide ..... 215
    - 7.9.2 Taille d'une collection ..... 216
    - 7.9.3 Contrôler la présence d'un élément ..... 216



# 8 \_\_\_\_\_ JPA et Java Hibernate

Apprenez le mapping objet-relationnel (ORM) avec Java

8. Les clauses GROUP BY et HAVING . . . . .	217
9. La clause ORDER BY . . . . .	218
10. L'UPDATE . . . . .	219
10.1 Mise à jour de toutes les données . . . . .	220
10.2 Mise à jour restrictive . . . . .	220
11. Le DELETE . . . . .	221
11.1 Supprimer toutes les données . . . . .	221
11.2 Suppressions restrictives . . . . .	222
12. L'INSERT . . . . .	222

## **Chapitre 6** **L'API Criteria**

1. Introduction . . . . .	223
2. L'API Metamodel . . . . .	225
2.1 Le métamodèle dynamique . . . . .	226
2.2 Le métamodèle statique . . . . .	226
2.2.1 Attribut simple . . . . .	227
2.2.2 Attribut basé sur une collection . . . . .	227
2.2.3 Conclusion . . . . .	229
3. Généralités . . . . .	229
3.1 Utilisation du métamodèle . . . . .	230
4. Le type d'opération . . . . .	232
4.1 CriteriaQuery . . . . .	233
4.1.1 Le type de retour . . . . .	233
4.1.2 Paramétrage du périmètre . . . . .	234
4.1.3 Paramétrage de la restriction . . . . .	234
4.1.4 Paramétrage du regroupement . . . . .	235
4.1.5 Paramétrage du retour . . . . .	235
4.1.6 Paramétrage du tri . . . . .	236
4.1.7 Préparation de la requête . . . . .	237
4.1.8 Exécution de la requête . . . . .	238

- 4.2 CriteriaUpdate ..... 239
  - 4.2.1 Type d'entité modifié ..... 239
  - 4.2.2 Paramétrage du périmètre ..... 240
  - 4.2.3 Initialisation des valeurs à modifier ..... 241
  - 4.2.4 Paramétrage de la restriction ..... 241
  - 4.2.5 Exécution de la requête ..... 242
- 4.3 CriteriaDelete ..... 242
  - 4.3.1 Type d'entité modifié ..... 243
  - 4.3.2 Paramétrage du périmètre ..... 243
  - 4.3.3 Paramétrage de la restriction ..... 243
  - 4.3.4 Exécution de la requête ..... 244
- 5. Le périmètre (FROM) ..... 244
  - 5.1 Root ..... 245
  - 5.2 Multiple Root ..... 245
  - 5.3 Jointures ..... 247
    - 5.3.1 Type de jointure ..... 247
    - 5.3.2 Modifier la condition de jointure ..... 248
    - 5.3.3 Multiple jointure ..... 249
    - 5.3.4 Fetch ..... 250
- 6. Les restrictions (WHERE) ..... 251
  - 6.1 Avec Expression <Boolean> ..... 252
  - 6.2 Avec un Predicate ..... 252
  - 6.3 Avec plusieurs Predicate ..... 253
  - 6.4 Avec Predicate[] ..... 254
  - 6.5 Sans paramètre ..... 256
- 7. Le regroupement (GROUP BY et HAVING) ..... 256
  - 7.1 GROUP BY ..... 258
  - 7.2 HAVING ..... 259
- 8. Le tri (ORDER BY) ..... 261
  - 8.1 L'objet Order ..... 262
  - 8.2 ORDER BY ..... 263

# 10 \_\_\_\_\_ JPA et Java Hibernate

Apprenez le mapping objet-relationnel (ORM) avec Java

9. Les expressions .....	265
9.1 PATH.....	265
9.2 Paramètres.....	266
9.3 Tests et comparaisons (Predicate).....	267
9.3.1 La nullité .....	267
9.3.2 Le booléen .....	268
9.3.3 La négation.....	269
9.3.4 L'égalité.....	269
9.3.5 L'infériorité et la supériorité .....	270
9.3.6 Le LIKE .....	271
9.3.7 Le BETWEEN.....	271
9.3.8 AND et OR .....	271
9.4 Opérations arithmétiques .....	273
9.5 Fonctions d'agrégation .....	274
9.6 Fonctions scalaires .....	276
9.6.1 Literal .....	276
9.6.2 CONCAT.....	277
9.6.3 SUBSTRING .....	277
9.6.4 UPPER .....	277
9.6.5 LOWER .....	278
9.6.6 TRIM .....	278
9.6.7 LENGTH .....	279
9.6.8 LOCATE.....	279
9.6.9 CASE .....	279
9.6.10 CURRENT_DATE .....	281
9.6.11 CURRENT_TIME.....	281
9.6.12 CURRENT_TIMESTAMP .....	281
9.7 Collections .....	282
9.7.1 Vérifier qu'une liste est vide .....	282
9.7.2 Taille d'une collection .....	282
9.7.3 Contrôler la présence d'un élément .....	283

- 10. Exemples ..... 285
  - 10.1 Écriture minimum ..... 285
  - 10.2 Écriture successive des méthodes ..... 285
  - 10.3 Requête dynamique ..... 286

**Chapitre 7**  
**Pour aller plus loin**

- 1. Maven ..... 289
  - 1.1 NetBeans ..... 289
    - 1.1.1 Configuration ..... 289
    - 1.1.2 Création d'un projet ..... 292
  - 1.2 Hibernate ..... 295
  - 1.3 MySQL ..... 295
  - 1.4 Exemple de fichier POM ..... 295
- 2. Génération automatique ..... 296
  - 2.1 Génération des entités ..... 296
  - 2.2 Génération des tables ..... 303
    - 2.2.1 Configuration de l'action ..... 303
    - 2.2.2 Suppression et création ..... 304
    - 2.2.3 Chargement des données ..... 305
    - 2.2.4 Résumé ..... 306
  - 2.3 Génération du métamodèle ..... 307
    - 2.3.1 Avec l'annotation processor de NetBeans ..... 307
    - 2.3.2 Avec Maven ..... 312
- 3. Gestion des caches ..... 315
  - 3.1 Généralités ..... 315
  - 3.2 Outils de gestion de cache de second niveau ..... 320
    - 3.2.1 Installation d'Ehcache ..... 320
    - 3.2.2 Configuration d'Ehcache ..... 321
    - 3.2.3 Paramétrage d'Hibernate ..... 322
    - 3.2.4 Entité et relation de type entité ..... 322
    - 3.2.5 Entité et relation de type liste ..... 326

# 12 \_\_\_\_\_ JPA et Java Hibernate

Apprenez le mapping objet-relationnel (ORM) avec Java

3.3	Conclusion	329
4.	Les contrôles entre JPA et la base de données	329
4.1	Les EntityListener	329
4.2	Les convertisseurs	332
5.	Pool de connexions	334
5.1	Installation	335
5.2	Configuration	335
6.	Divers	337
6.1	Affichage des requêtes	337
6.2	Multithreading	339
6.3	Récupérer une session de l'implémentation	340
6.4	MVC	341

## Chapitre 8

### Réalisation d'un projet

1.	Introduction	347
1.1	Import du projet	348
2.	Définition du projet	350
2.1	L'application dans le système d'information	350
2.1.1	Java SE ou Java EE	350
2.1.2	Modification de données	351
2.1.3	Nombre d'exécutions de l'application	351
2.2	Fonctionnement de l'application	351
2.2.1	Gestion de la connexion	351
2.2.2	Temps de veille de l'application	352
2.2.3	Nombre de requêtes simultanées	352
2.2.4	Données de référence	353
2.2.5	Requêtes dynamiques	353
2.3	Création de l'application	353
2.3.1	Configuration initiale du projet	354
2.3.2	Création de JpaUtil	356

2.3.3	Vérification de la configuration	357
3.	Mise en place du modèle	357
3.1	Création du modèle	357
3.2	Personnalisation du modèle	357
3.2.1	Vérifier les types	358
3.2.2	Vérifier les relations entre les entités	358
3.2.3	Modifier equals() et hashCode()	359
3.2.4	Ajouter des requêtes nommées	359
3.2.5	Mettre en place des générateurs spécifiques	359
3.2.6	Utiliser les listeners d'entités (@PrePersist...)	359
3.2.7	Cache de second niveau	360
3.3	Mise en place du métamodèle	361
4.	Réalisation des contrôleurs	363
4.1	Création des contrôleurs	363
4.2	Personnalisation des contrôleurs	364
4.2.1	La construction du contrôleur	364
4.2.2	La gestion des opérations en cascade	365
4.2.3	Les données de référence	366
4.3	Contrôleur de plusieurs entités	367
5.	L'utilisation	368
5.1	Création	369
5.2	Récupération de données	371
5.3	Modification	373
5.4	Suppression	376
6.	La préparation de la livraison	378
6.1	Vérification des fichiers de configuration	378
6.1.1	Connexions à la base de données	378
6.1.2	Pool de connexions (c3p0)	378
6.1.3	L'implémentation	379
6.1.4	Cache de second niveau (ehcache)	379
6.1.5	Conclusion	379
6.2	Vérification des dépendances	380

# 14 \_\_\_\_\_ JPA et Java Hibernate

Apprenez le mapping objet-relationnel (ORM) avec Java

Index .....	381
-------------	-----

## Chapitre 4

# Manipulation des données

### 1. Préparation

Afin de continuer sur ce chapitre, il faut avoir à disposition le fichier `persistence.xml` et les différentes entités créées dans le projet commencé à la section Premier lancement de NetBeans - Création du projet du chapitre Environnement de développement et continué tout au long du chapitre Préparation d'un projet. Pour cela, le fichier `persistence.xml` et les entités sont à remplacer par ceux trouvés dans l'exemple `PROJET_ENI_MAVEN.zip` téléchargeable sur le site des Éditions ENI.

Une fois les fichiers mis en place, il faut modifier les informations de connexion dans le fichier `persistence.xml` pour correspondre à vos paramètres de connexion à la base de données comme expliqué à la section Paramétrage de l'ORM - Les propriétés du fichier de persistance du chapitre Préparation d'un projet :

```
<property name="javax.persistence.jdbc.url" value="..."/>
<property name="javax.persistence.jdbc.user" value="..."/>
<property name="javax.persistence.jdbc.driver" value="..."/>
<property name="javax.persistence.jdbc.password" value="..."/>
```

Afin de voir les requêtes envoyées par l'ORM au système de gestion de base de données, il faut ajouter une propriété dans le fichier de persistance qui est propre à chaque implémentation, il n'y a pas de norme JPA pour ce paramètre.

Par exemple, pour Hibernate la propriété est :

```
■ <property name = "hibernate.show_sql" value = "true" />
```



## 2. Établissement de la connexion

Maintenant que le mapping des données a été réalisé, il n'y a plus qu'à se connecter à la base de données pour pouvoir les manipuler. C'est-à-dire les créer, les lire, les modifier ou les supprimer. Pour cela, il a été vu au chapitre Concept des ORM, que l'application doit obligatoirement interroger les données via un `EntityManager`, que cet `EntityManager` doit être fourni par un `EntityManagerFactory` qui lui-même connaît le mapping objet-relationnel (l'unité de persistance).

### 2.1 EntityManagerFactory

Pour créer l'`EntityManagerFactory`, il faut le nom de l'unité de persistance qui se trouve dans le fichier `persistence.xml`.

Par exemple, dans le fichier fourni, le nom est `projetEni` et se trouve à la ligne suivante :

```
■ <persistence-unit name="projetEni" transaction-type="RESOURCE_LOCAL">
```

L'`EntityManagerFactory` ne peut pas être instancié simplement car cela voudrait dire qu'il faut instancier l'`EntityManagerFactory` de l'implémentation de JPA, ce qui équivaldrait à cloisonner l'application et perdre le niveau d'abstraction que JPA permet.

Il a été vu que l'unité de persistance peut être paramétrée soit en mode `RESOURCE_LOCAL` soit en mode `JTA`. En mode `JTA`, l'unité de persistance est gérée par le conteneur JEE, il n'y a donc pas d'`EntityManagerFactory` à paramétrer au niveau du code source. En mode `RESOURCE_LOCAL`, elle doit être gérée par l'application. Deux solutions sont possibles selon s'il y a un conteneur JEE ou non.

#### 2.1.1 Avec conteneur JEE, en RESOURCE\_LOCAL

Lorsqu'un conteneur JEE est disponible, il est recommandé de laisser le serveur gérer l'`EntityManager`. Pour cela, il faut l'injecter dans le code source en utilisant l'annotation `@PersistenceUnit` juste avant la déclaration de l'`EntityManagerFactory`.

L'exemple ci-dessous montre comment configurer l'`EntityManagerFactory` avec l'unité de persistance "projetEni" :

```
@PersistenceUnit(unitName = "projetEni")
private EntityManagerFactory emf;
```

Cette utilisation de JPA, qui demande un conteneur JEE, n'est pas davantage détaillée dans ce chapitre et est donnée à titre informatif.

### 2.1.2 Sans conteneur JEE, en RESOURCE\_LOCAL

Lorsqu'il n'y a pas de conteneur JEE (ou qu'il n'est pas utilisé pour JPA), il faut récupérer une instance d'`EntityManagerFactory`. Pour cela, il faut utiliser la classe `javax.persistence.Persistence` fournie par JPA.

Par exemple, la récupération d'un `EntityManagerFactory` pour l'unité de persistance "projetEni" s'écrit de la manière suivante :

```
Persistence.createEntityManagerFactory("projetEni");
```

Il a été vu qu'un `EntityManagerFactory` doit être unique durant la vie de l'application.

Par exemple, créer une classe utilitaire contenant cette instance unique permet de répondre à cette contrainte. Pour cela, il faut créer une classe `JpaUtil` dans le package `com.eni.jpa.util` :

```
public class JpaUtil {

    private static EntityManagerFactory emf = null;

    private JpaUtil() {
    }

    public static EntityManagerFactory getEmf() {
        if(emf == null){
            emf =
Persistence.createEntityManagerFactory("projetEni");
        }
        return emf;
    }
}
```

Comme les ressources sont gérées par l'application, il faut penser à les libérer avant de fermer l'application afin de fermer les connexions à la base de données. Pour cela, l'`EntityManagerFactory` possède la méthode `close()` qui permet de libérer les ressources.

Par exemple, comme une classe utilitaire `JpaUtil` a été créée, il suffit d'ajouter cette méthode et d'affecter `null` à la variable si jamais cette méthode est appelée en cours d'application pour pouvoir reconstruire l'`EntityManagerFactory`.

```
/**
 * Classe utilitaire pour JPA
 */
public class JpaUtil {

    /**
     * Singleton de l'EntityManagerFactory de l'application
     */
    private static EntityManagerFactory emf = null;

    /**
     * Permet de récupérer l'EntityManagerFactory de
     * l'application tout en le créant s'il n'existe pas
     *
     * @return l'EntityManagerFactory unique de l'application
     */
    public static EntityManagerFactory getEmf() {
        if(emf == null){
            emf = Persistence.createEntityManagerFactory("jpaTest");
        }
        return emf;
    }

    /**
     * Libère les ressources et détruit l'EntityManagerFactory
     * si jamais il faut le recréer.
     */
    public static void close(){
        if(emf!=null){
            emf.close();
            emf=null;
        }
    }
}
```

Finalement, avec cette classe l'initialisation de l'`EntityManagerFactory` se fait bien une seule fois lors du premier appel à la méthode `JpaUtil.getEmf()`, il est possible de le récupérer à tout moment de l'application et de libérer les ressources soit pour le recréer, soit lors de la fermeture de l'application.

En créant une simple méthode `main`, il est possible de tester dans un premier temps que les paramètres sont corrects :

```
public static void main(String[] args) {
    //1
    System.out.println("Création de l'emf");
    JpaUtil.getEmf();
    //2
    System.out.println("Emf créé, nouvelle récupération de l'emf");
    JpaUtil.getEmf();
    //3
    System.out.println("fermeture de l'emf");
    JpaUtil.close();
    //4
    System.out.println("emf fermé, recréation d'un autre emf");
    JpaUtil.getEmf();
    //5
    System.out.println("fermeture de l'emf");
    JpaUtil.close();
    //6
    System.out.println("emf fermé, arrêt de l'application");
}
```

Ce qui donne dans la console, lors de l'exécution de la première étape, le chargement de JPA et donc son implémentation Hibernate avec les paramètres saisis dans le fichier de persistance.

```
Création de l'emf
oct. 25, 2016 4:08:57 PM org.hibernate.jpa.internal.util.LogHelper
logPersistenceUnitInformation
INFO: HHH000204: Processing PersistenceUnitInfo [
    name: projetEni
    ...]
oct. 25, 2016 4:08:57 PM org.hibernate.Version logVersion
INFO: HHH000412: Hibernate Core {5.2.3.Final}
oct. 25, 2016 4:08:57 PM org.hibernate.cfg.Environment <clinit>
INFO: HHH000206: hibernate.properties not found
oct. 25, 2016 4:08:57 PM org.hibernate.cfg.Environment
buildBytecodeProvider
INFO: HHH000021: Bytecode provider name : javassist
```

```

oct. 25, 2016 4:08:57 PM
org.hibernate.annotations.common.reflection.java.JavaReflectionManager
<clinit>
INFO: HCANN000001: Hibernate Commons Annotations {5.0.1.Final}
oct. 25, 2016 4:08:57 PM org.hibernate.engine.jdbc.connections.internal.
DriverManagerConnectionProviderImpl configure
WARN: HHH10001002: Using Hibernate built-in connection pool (not for
production use!)
oct. 25, 2016 4:08:57 PM org.hibernate.engine.jdbc.connections.internal.
DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001005: using driver [com.mysql.jdbc.Driver] at URL
[jdbc:mysql://localhost:3306]
oct. 25, 2016 4:08:57 PM org.hibernate.engine.jdbc.connections.internal.
DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001001: Connection properties: {user=root, password=****}
oct. 25, 2016 4:08:57 PM org.hibernate.engine.jdbc.connections.internal.
DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
oct. 25, 2016 4:08:57 PM
org.hibernate.engine.jdbc.connections.internal.PooledConnections <init>
INFO: HHH000115: Hibernate connection pool size: 20 (min=1)
Tue Oct 25 16:08:57 CEST 2016 WARN: Establishing SSL connection without
server's identity verification is not recommended. According to MySQL
5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established
by default if explicit option isn't set. For compliance with existing
applications not using SSL the verifyServerCertificate property is set to
'false'. You need either to explicitly disable SSL by setting useSSL=false,
or set useSSL=true and provide truststore for server certificate
verification.
oct. 25, 2016 4:08:57 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL5Dialect

```

Pour la deuxième étape, il n'y a rien de particulier car l'EntityManagerFactory a déjà été créé.

■ Emf créé, nouvelle récupération de l'emf

Pour la troisième étape, la connexion avec la base de données est bien fermée.

```

fermeture de l'emf
oct. 25, 2016 4:08:57 PM
org.hibernate.engine.jdbc.connections.internal.DriverManager
ConnectionProviderImpl stop
INFO: HHH10001008: Cleaning up connection pool
[jdbc:mysql://localhost:3306]

```

Pour la quatrième étape, comme l'EntityManagerFactory a été détruit juste avant, il est recréé.