

DAO

```

1  package fr.insalyon.dasi.gustatif.dao;
2
3  import javax.persistence.EntityManager;
4  import javax.persistence.EntityManagerFactory;
5  import javax.persistence.Persistence;
6  import javax.persistence.RollbackException;
7
8  /**
9   * Cette classe fournit des méthodes statiques utiles pour accéder aux
10  * fonctionnalités de JPA (Entity Manager, Entity Transaction). Le nom de
11  * l'unité de persistance (PERSISTENCE_UNIT_NAME) doit être conforme à la
12  * configuration indiquée dans le fichier persistence.xml du projet.
13  *
14  * Version du 22 Mars 2016
15  *
16  * @author DASI Team
17  */
18  public class JpaUtil {
19
20      //
21      // *****
22      // * TODO: IMPORTANT -- Adapter le nom de l'Unité de Persistance
23      // * (cf. persistence.xml) *
24      // *****
25      /**
26       * Nom de l'unité de persistance utilisée par la Factory de Entity
27       * Manager.
28       * <br/><strong>Vérifier le nom de l'unité de persistance
29       * (cf.&nbsp;persistence.xml)</strong>
30       */
31      public static final String PERSISTENCE_UNIT_NAME =
32          "fr.insalyon.dasi_Gustatif_database";
33      /**
34       * Factory de Entity Manager liée à l'unité de persistance.
35       * <br/><strong>Vérifier le nom de l'unité de persistance indiquée
36       * dans
37       * l'attribut statique PERSISTENCE_UNIT_NAME
38       * (cf.&nbsp;persistence.xml)</strong>
39       */
40      private static EntityManagerFactory entityManagerFactory =
41          Persistence.createEntityManagerFactory(PERSISTENCE_UNIT_NAME);
42      /**
43       * Gère les instances courantes de Entity Manager liées aux Threads.
44       * L'utilisation de ThreadLocal garantie une unique instance
45       * courante par
46       * Thread.
47       */
48      private static final ThreadLocal<EntityManager>
49          threadLocalEntityManager = new ThreadLocal<EntityManager>() {
50
51          @Override

```

```

44         protected EntityManager initialValue() {
45             return null;
46         }
47     };
48
49     // Pause (sans exception)
50     private static void pause(long milliseconds) {
51         try {
52             Thread.sleep(milliseconds);
53         } catch (InterruptedException ex) {
54             }
55     }
56
57     // Log sur la console
58     // Les flush & pause sont là pour (tenter de) synchroniser les
    sorties sur la console
59     private static void log(String message) {
60         System.out.flush();
61         pause(5);
62         System.err.println("[JpaUtil:Log] " + message);
63         System.err.flush();
64         pause(5);
65     }
66
67     /**
68     * Initialise la Factory de Entity Manager (nécessaire au
    fonctionnement de
69     * JpaUtil dans une Application Web sous Glassfish).
70     * <br><strong>À utiliser uniquement dans la méthode init() de la
    Servlet
71     * Contrôleur (ActionServlet).</strong>
72     */
73     public static synchronized void init() {
74         log("Initialisation de la factory de contexte de persistance");
75         if (entityManagerFactory != null) {
76             entityManagerFactory.close();
77         }
78         entityManagerFactory =
    Persistence.createEntityManagerFactory(PERSISTENCE_UNIT_NAME);
79     }
80
81     /**
82     * Libère la Factory de Entity Manager (pour permettre un futur
    rechargement
83     * propre d'une Application Web sous Glassfish).
84     * <br><strong>À utiliser uniquement dans la méthode destroy() de la
    Servlet
85     * Contrôleur (ActionServlet).</strong>
86     */
87     public static synchronized void destroy() {
88         log("Libération de la factory de contexte de persistance");
89         if (entityManagerFactory != null) {
90             entityManagerFactory.close();
91             entityManagerFactory = null;
92         }

```

```

93     }
94
95     /**
96     * Crée l'instance courante de Entity Manager (liée à ce Thread).
97     * <br><strong>À utiliser uniquement au niveau Service.</strong>
98     */
99     public static void creerEntityManager() {
100         log("Création du contexte de persistance");
101
102         threadLocalEntityManager.set(entityManagerFactory.createEntityManager());
103     }
104
105     /**
106     * Ferme l'instance courante de Entity Manager (liée à ce Thread).
107     * <br><strong>À utiliser uniquement au niveau Service.</strong>
108     */
109     public static void fermerEntityManager() {
110         log("Fermeture du contexte de persistance");
111         EntityManager em = threadLocalEntityManager.get();
112         em.close();
113         threadLocalEntityManager.set(null);
114     }
115
116     /**
117     * Démarre une transaction sur l'instance courante de Entity Manager.
118     * <br><strong>À utiliser uniquement au niveau Service.</strong>
119     */
120     public static void ouvrirTransaction() {
121         log("Début de la transaction");
122         EntityManager em = threadLocalEntityManager.get();
123         if (em.getTransaction().isActive()) {
124             log("ATTENTION: la transaction est déjà ouverte");
125         }
126         em.getTransaction().begin();
127     }
128
129     /**
130     * Valide la transaction courante sur l'instance courante de
131     * Entity Manager.
132     * <br><strong>À utiliser uniquement au niveau Service.</strong>
133     *
134     * @exception RollbackException lorsque le <em>commit</em> n'a pas
135     * réussi.
136     */
137     public static void validerTransaction() throws RollbackException {
138         log("Validation (commit) de la transaction");
139         EntityManager em = threadLocalEntityManager.get();
140         if (!em.getTransaction().isActive()) {
141             log("ATTENTION: la transaction N'est PAS ouverte");
142         }
143         em.getTransaction().commit();
144     }

```

```

143  /**
144  * Annule la transaction courante sur l'instance courante de
145  * Entity Manager.
146  * Si la transaction courante n'est pas démarrée, cette méthode
147  * n'effectue
148  * aucune opération.
149  * <br/><strong>À utiliser uniquement au niveau Service.</strong>
150  */
151  public static void annulerTransaction() {
152      log("Annulation (rollback) de la transaction");
153      EntityManager em = threadLocalEntityManager.get();
154      if (!em.getTransaction().isActive()) {
155          log("ATTENTION: la transaction N'est PAS ouverte =>
156              annulation ignorée par JpaUtil");
157      } else {
158          em.getTransaction().rollback();
159      }
160  }
161
162  /**
163  * Retourne l'instance courante de Entity Manager.
164  * <br/><strong>À utiliser uniquement au niveau DAO.</strong>
165  *
166  * @return instance de Entity Manager
167  */
168  public static EntityManager obtenirEntityManager() {
169      log("Obtention du contexte de persistance");
170      return threadLocalEntityManager.get();
171  }

```

```
1  package fr.insalyon.dasi.gustatif.dao;
2
3  import fr.insalyon.dasi.gustatif.metier.modele.Client;
4  import java.util.List;
5  import javax.persistence.EntityManager;
6  import javax.persistence.Query;
7
8
9  public class ClientDao {
10
11     public void create(Client client) throws Throwable {
12         EntityManager em = JpaUtil.obtenirEntityManager();
13         try {
14             em.persist(client);
15         }
16         catch(Exception e) {
17             throw e;
18         }
19     }
20
21     public Client update(Client client) throws Throwable {
22         EntityManager em = JpaUtil.obtenirEntityManager();
23         try {
24             client = em.merge(client);
25         }
26         catch(Exception e){
27             throw e;
28         }
29         return client;
30     }
31
32     public Client findById(Long id) throws Throwable {
33         EntityManager em = JpaUtil.obtenirEntityManager();
34         Client client = null;
35         try {
36             client = em.find(Client.class, id);
37         }
38         catch(Exception e) {
39             throw e;
40         }
41         return client;
42     }
43
44     public Client findByMailAndPassword(String mail, String password)
45         throws Throwable {
46         EntityManager em = JpaUtil.obtenirEntityManager();
47         List<Client> clients = null;
48         Client c = null;
49         try {
50             Query q = em.createQuery("SELECT c FROM Client c WHERE
51                                     c.mail = "
52                                     + " :mail AND c.motDePasse = :password");
53             q.setParameter("mail", mail);
54             q.setParameter("password", password);
```

```
54         clients = (List<Client>) q.getResultList();
55     }
56     catch(Exception e) {
57         throw e;
58     }
59     try {
60         c = clients.get(0);
61     } catch (Exception e){
62     }
63     return c;
64 }
65
66 public Client findByMail(String mail) throws Throwable {
67     EntityManager em = JpaUtil.obtenirEntityManager();
68     List<Client> clients = null;
69     Client c = null;
70     try {
71         Query q = em.createQuery("SELECT c FROM Client c "
72             + "WHERE c.mail = :mail");
73         q.setParameter("mail", mail);
74         clients = (List<Client>) q.getResultList();
75     }
76     catch(Exception e) {
77         throw e;
78     }
79     try {
80         c = clients.get(0);
81     } catch (Exception e){
82     }
83     return c;
84 }
85
86 public List<Client> findAll() throws Throwable {
87     EntityManager em = JpaUtil.obtenirEntityManager();
88     List<Client> clients = null;
89     try {
90         Query q = em.createQuery("SELECT c FROM Client c ORDER BY c.nom");
91         clients = (List<Client>) q.getResultList();
92     }
93     catch(Exception e) {
94         throw e;
95     }
96     return clients;
97 }
98 }
99
```

```
1 package fr.insalyon.dasi.gustatif.dao;
2
3
4 import fr.insalyon.dasi.gustatif.metier.modele.Restaurant;
5 import java.util.List;
6 import javax.persistence.EntityManager;
7 import javax.persistence.Query;
8
9
10 public class RestaurantDao {
11
12     public void create(Restaurant restaurant) throws Throwable {
13         EntityManager em = JpaUtil.obtenirEntityManager();
14         try {
15             em.persist(restaurant);
16         }
17         catch (Exception e) {
18             throw e;
19         }
20     }
21
22     public Restaurant update(Restaurant restaurant) throws Throwable {
23         EntityManager em = JpaUtil.obtenirEntityManager();
24         try {
25             restaurant = em.merge(restaurant);
26         }
27         catch (Exception e) {
28             throw e;
29         }
30         return restaurant;
31     }
32
33     public Restaurant findById(Long id) throws Throwable {
34         EntityManager em = JpaUtil.obtenirEntityManager();
35         Restaurant restaurant = null;
36         try {
37             restaurant = em.find(Restaurant.class, id);
38         }
39         catch (Exception e) {
40             throw e;
41         }
42         return restaurant;
43     }
44
45     public List<Restaurant> findAll() throws Throwable {
46         EntityManager em = JpaUtil.obtenirEntityManager();
47         List<Restaurant> restaurants = null;
48         try {
49             Query q = em.createQuery("SELECT r FROM Restaurant r ORDER BY r.denomination");
50             restaurants = (List<Restaurant>) q.getResultList();
51         }
52         catch (Exception e) {
53             throw e;
```



```
54         }  
55  
56         return restaurants;  
57     }  
58  
59  
60 }  
61
```

```
1  package fr.insalyon.dasi.gustatif.dao;
2
3  import fr.insalyon.dasi.gustatif.metier.modele.Produit;
4  import java.util.List;
5  import javax.persistence.EntityManager;
6  import javax.persistence.Query;
7
8  public class ProduitDao {
9      public void create(Produit produit) throws Throwable {
10         EntityManager em = JpaUtil.obtenirEntityManager();
11         try {
12             em.persist(produit);
13         }
14         catch(Exception e) {
15             throw e;
16         }
17     }
18
19     public Produit update(Produit produit) throws Throwable {
20         EntityManager em = JpaUtil.obtenirEntityManager();
21         try {
22             produit = em.merge(produit);
23         }
24         catch(Exception e){
25             throw e;
26         }
27         return produit;
28     }
29
30     public Produit findById(Long id) throws Throwable {
31         EntityManager em = JpaUtil.obtenirEntityManager();
32         Produit produit = null;
33         try {
34             produit = em.find(Produit.class, id);
35         }
36         catch(Exception e) {
37             throw e;
38         }
39         return produit;
40     }
41
42     public List<Produit> findAll() throws Throwable {
43         EntityManager em = JpaUtil.obtenirEntityManager();
44         List<Produit> produits = null;
45         try {
46             Query q = em.createQuery("SELECT p FROM Produit p");
47             produits = (List<Produit>) q.getResultList();
48         }
49         catch(Exception e) {
50             throw e;
51         }
52         return produits;
53     }
54 }
```

```

1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package fr.insalyon.dasi.gustatif.dao;
7
8  import fr.insalyon.dasi.gustatif.metier.modele.Livreur;
9  import java.util.List;
10 import javax.persistence.EntityManager;
11 import javax.persistence.Query;
12
13 /**
14  *
15  * @author Nico
16  */
17 public class LivreurDao {
18     public void create(Livreur livreur) throws Throwable {
19         EntityManager em = JpaUtil.obtenirEntityManager();
20         try {
21             em.persist(livreur);
22         }
23         catch (Exception e) {
24             throw e;
25         }
26     }
27
28     public Livreur update(Livreur livreur) throws Throwable {
29         EntityManager em = JpaUtil.obtenirEntityManager();
30         try {
31             livreur = em.merge(livreur);
32         }
33         catch (Exception e) {
34             throw e;
35         }
36         return livreur;
37     }
38
39     public Livreur findById(Long id) throws Throwable {
40         EntityManager em = JpaUtil.obtenirEntityManager();
41         Livreur livreur = null;
42         try {
43             livreur = em.find(Livreur.class, id);
44         }
45         catch (Exception e) {
46             throw e;
47         }
48         return livreur;
49     }
50
51
52     public List<Livreur> findAll() throws Throwable {
53         EntityManager em = JpaUtil.obtenirEntityManager();

```

```
54         List<Livreur> livreurs = null;
55         try {
56             Query q = em.createQuery("SELECT l FROM Livreur l");
57             livreurs = (List<Livreur>) q.getResultList();
58         }
59         catch(Exception e) {
60             throw e;
61         }
62         return livreurs;
63     }
64
65     public List<Livreur> findAllDisponible() throws Throwable
66     {
67         EntityManager em = JpaUtil.obtenirEntityManager();
68         List<Livreur> livreurs = null;
69         try {
70             Query q = em.createQuery("SELECT l FROM Livreur l WHERE "
71                                     + "l.disponible = TRUE");
72             livreurs = (List<Livreur>) q.getResultList();
73         }
74         catch(Exception e) {
75             throw e;
76         }
77         return livreurs;
78     }
79
80     public List<Livreur> findAllDisponible(Double commandeWeight) throws Throwable
81     {
82         EntityManager em = JpaUtil.obtenirEntityManager();
83         List<Livreur> livreurs = null;
84         try {
85             Query q = em.createQuery("SELECT l FROM Livreur l WHERE "
86                                     + "l.disponible = TRUE AND l.chargeMaxEnGrammes >= "
87                                     + ":commandeWeight");
88             q.setParameter("commandeWeight", commandeWeight);
89             livreurs = (List<Livreur>) q.getResultList();
90         }
91         catch(Exception e) {
92             throw e;
93         }
94         return livreurs;
95     }
96 }
```

```
1 package fr.insalyon.dasi.gustatif.dao;
2
3 import fr.insalyon.dasi.gustatif.metier.modele.Commande;
4 import fr.insalyon.dasi.gustatif.metier.modele.LigneDeCommande;
5 import java.util.List;
6 import javax.persistence.EntityManager;
7 import javax.persistence.Query;
8
9 /*
10  * To change this license header, choose License Headers in Project
11  * Properties.
12  * To change this template file, choose Tools | Templates
13  * and open the template in the editor.
14  */
15 /**
16  *
17  * @author Nico
18  */
19 public class LigneDeCommandeDao {
20
21     public void create(LigneDeCommande ligneDeCommande) throws
22     Throwable {
23         EntityManager em = JpaUtil.obtenirEntityManager();
24         try {
25             em.persist(ligneDeCommande);
26         }
27         catch (Exception e) {
28             throw e;
29         }
30     }
31
32     public LigneDeCommande update(LigneDeCommande ligneDeCommande)
33     throws Throwable {
34         EntityManager em = JpaUtil.obtenirEntityManager();
35         try {
36             ligneDeCommande = em.merge(ligneDeCommande);
37         }
38         catch (Exception e){
39             throw e;
40         }
41         return ligneDeCommande;
42     }
43
44     public LigneDeCommande findById(Long id) throws Throwable {
45         EntityManager em = JpaUtil.obtenirEntityManager();
46         LigneDeCommande ligneDeCommande = null;
47         try {
48             ligneDeCommande = em.find(LigneDeCommande.class, id);
49         }
50         catch (Exception e) {
51             throw e;
52         }
53         return ligneDeCommande;
54     }
55 }
```

```
53     }
54
55     public List<LigneDeCommande> findByCommandeID(Long commandeId)
56         throws Throwable {
57         EntityManager em = JpaUtil.obtenirEntityManager();
58         List<LigneDeCommande> lignesDeCommande = null;
59
60         try {
61             Commande c = em.find(Commande.class, commandeId);
62             if(c != null)
63             {
64                 lignesDeCommande = c.getLignesDeCommande();
65                 if(lignesDeCommande.isEmpty())
66                 {
67                     lignesDeCommande = null;
68                 }
69             }
70         }
71         catch(Exception e) {
72             throw e;
73         }
74         return lignesDeCommande;
75     }
76
77     public List<LigneDeCommande> findAll() throws Throwable {
78         EntityManager em = JpaUtil.obtenirEntityManager();
79         List<LigneDeCommande> lignesDeCommande = null;
80         try {
81             Query q = em.createQuery("SELECT l FROM LigneDeCommande l");
82             lignesDeCommande = (List<LigneDeCommande>) q.getResultList();
83         }
84         catch(Exception e) {
85             throw e;
86         }
87         return lignesDeCommande;
88     }
89 }
90
```

```
1 package fr.insalyon.dasi.gustatif.dao;
2
3 import fr.insalyon.dasi.gustatif.metier.modele.LivreurDrone;
4 import java.util.List;
5 import javax.persistence.EntityManager;
6 import javax.persistence.Query;
7
8 /*
9  * To change this license header, choose License Headers in Project Properties.
10  * To change this template file, choose Tools | Templates
11  * and open the template in the editor.
12  */
13
14 /**
15  *
16  * @author Nico
17  */
18 public class LivreurDroneDao extends LivreurDao {
19
20     public List<LivreurDrone> findAllLivresDrones() throws Throwable {
21         EntityManager em = JpaUtil.obtenirEntityManager();
22         List<LivreurDrone> livres = null;
23         try {
24             Query q = em.createQuery("SELECT l FROM LivreurDrone l");
25             livres = (List<LivreurDrone>) q.getResultList();
26         }
27         catch(Exception e) {
28             throw e;
29         }
30         return livres;
31     }
32
33     public List<LivreurDrone> findAllLivresDronesDisponibles() throws Throwable {
34         EntityManager em = JpaUtil.obtenirEntityManager();
35         List<LivreurDrone> livres = null;
36         try {
37             Query q = em.createQuery("SELECT l FROM LivreurDrone l
38                                     WHERE "
39                                     + "l.disponible = TRUE");
40             livres = (List<LivreurDrone>) q.getResultList();
41         }
42         catch(Exception e) {
43             throw e;
44         }
45         return livres;
46     }
47
48     public List<LivreurDrone> findAllLivresDronesDisponibles(Double commandeWeight) throws Throwable {
49
50         EntityManager em = JpaUtil.obtenirEntityManager();
```

```
51     List<LivreurDrone> livreurs = null;
52     try {
53         Query q = em.createQuery("SELECT l FROM LivreurDrone l
54                                   WHERE "
55                                   + "l.disponible = TRUE AND l.chargeMax >=
56                                   :commandeWeight");
57         q.setParameter("commandeWeight", commandeWeight);
58         livreurs = (List<LivreurDrone>) q.getResultList();
59     }
60     catch(Exception e) {
61         throw e;
62     }
63     return livreurs;
64 }
65
66 public List<LivreurDrone>
67 findLivreursDronesByMailAndPassword(String mail, String password)
68     throws Throwable {
69     EntityManager em = JpaUtil.obtenirEntityManager();
70     List<LivreurDrone> livreurs = null;
71     try {
72         Query q = em.createQuery("SELECT l FROM LivreurDrone l
73                                   WHERE l.mail = :mail"
74                                   + " AND l.motDePasse = :password");
75         q.setParameter("mail", mail);
76         q.setParameter("password", password);
77         livreurs = (List<LivreurDrone>) q.getResultList();
78         if(livreurs.isEmpty()) {
79             livreurs = null;
80         }
81     }
82     catch(Exception e) {
83         throw e;
84     }
85     return livreurs;
86 }
```



```

1  /*
2  * To change this license header, choose License Headers in Project
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package fr.insalyon.dasi.gustatif.dao;
7
8  import fr.insalyon.dasi.gustatif.metier.modele.LivreurVelo;
9  import java.util.List;
10 import javax.persistence.EntityManager;
11 import javax.persistence.Query;
12
13 /**
14  *
15  * @author Nico
16  */
17 public class LivreurVeloDao extends LivreurDao {
18
19     public List<LivreurVelo> findAllLivreursVelos() throws Throwable {
20         EntityManager em = JpaUtil.obtenirEntityManager();
21         List<LivreurVelo> livreurs = null;
22         try {
23             Query q = em.createQuery("SELECT l FROM LivreurVelo l");
24             livreurs = (List<LivreurVelo>) q.getResultList();
25         }
26         catch(Exception e) {
27             throw e;
28         }
29         return livreurs;
30     }
31
32     public List<LivreurVelo> findAllLivreursVelosDisponibles() throws
33     Throwable {
34         EntityManager em = JpaUtil.obtenirEntityManager();
35         List<LivreurVelo> livreurs = null;
36         try {
37             Query q = em.createQuery("SELECT l FROM LivreurVelo l WHERE "
38                                     + "l.disponible = TRUE");
39             livreurs = (List<LivreurVelo>) q.getResultList();
40         }
41         catch(Exception e) {
42             throw e;
43         }
44         return livreurs;
45     }
46
47     public List<LivreurVelo> findAllLivreursVelosDisponibles(Double
48     commandeWeight)
49     throws Throwable
50     {
51         EntityManager em = JpaUtil.obtenirEntityManager();
52         List<LivreurVelo> livreurs = null;
53         try {

```

```
52         Query q = em.createQuery("SELECT l FROM LivreurDrone l
WHERE "
53         + "l.disponible = TRUE AND l.chargeMax >=
:commandeWeight");
54         q.setParameter("commandeWeight", commandeWeight);
55         livreurs = (List<LivreurVelo>) q.getResultList();
56     }
57     catch (Exception e) {
58         throw e;
59     }
60     return livreurs;
61 }
62
63 public LivreurVelo findLivreurVeloByMailAndPassword(String mail,
String password)
64     throws Throwable {
65     EntityManager em = JpaUtil.obtenirEntityManager();
66     List<LivreurVelo> livreurs = null;
67     LivreurVelo lv = null;
68     try {
69         Query q = em.createQuery("SELECT l FROM LivreurVelo l WHERE "
70         + "l.mail = :mail AND l.motDePasse = :password");
71         q.setParameter("mail", mail);
72         q.setParameter("password", password);
73         livreurs = (List<LivreurVelo>) q.getResultList();
74     }
75     catch (Exception e) {
76         throw e;
77     }
78     try {
79         lv = livreurs.get(0);
80     } catch (Exception e){
81     }
82     return lv;
83 }
84 }
85 }
```

```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package fr.insalyon.dasi.gustatif.dao;
7
8  import fr.insalyon.dasi.gustatif.metier.modele.Commande;
9  import fr.insalyon.dasi.gustatif.metier.modele.Livreur;
10 import java.util.List;
11 import javax.persistence.EntityManager;
12 import javax.persistence.Query;
13
14 /**
15  *
16  * @author Nico
17  */
18 public class CommandeDao {
19     public void create(Commande commande) throws Throwable {
20         EntityManager em = JpaUtil.obtenirEntityManager();
21         try {
22             em.persist(commande);
23         }
24         catch (Exception e) {
25             throw e;
26         }
27     }
28
29     public Commande update(Commande commande) throws Throwable {
30         EntityManager em = JpaUtil.obtenirEntityManager();
31         try {
32             commande = em.merge(commande);
33         }
34         catch (Exception e) {
35             throw e;
36         }
37         return commande;
38     }
39
40     public Commande findById(Long id) throws Throwable {
41         EntityManager em = JpaUtil.obtenirEntityManager();
42         Commande commande = null;
43         try {
44             commande = em.find(Commande.class, id);
45         }
46         catch (Exception e) {
47             throw e;
48         }
49         return commande;
50     }
51
52     public Commande findNotEndedByLivreur(Livreur livreur) throws
```

```

54     Throwable {
55         EntityManager em = JpaUtil.obtenirEntityManager();
56         Commande commande = null;
57         List<Commande> commandes = null;
58         try {
59             Query q = em.createQuery("SELECT c FROM Commande c WHERE
60                                     c.dateFin "
61                                     + "IS NULL AND c.livreur = :livreur");
62             q.setParameter("livreur", livreur);
63             commandes = (List<Commande>) q.getResultList();
64             if(!commandes.isEmpty()) {
65                 commande = commandes.get(0);
66             }
67         } catch (Exception e) {
68             throw e;
69         }
70         return commande;
71     }
72
73     public List<Commande> findAll() throws Throwable {
74         EntityManager em = JpaUtil.obtenirEntityManager();
75         List<Commande> commandes = null;
76         try {
77             Query q = em.createQuery("SELECT c FROM Commande c");
78             commandes = (List<Commande>) q.getResultList();
79         }
80         catch (Exception e) {
81             throw e;
82         }
83         return commandes;
84     }
85
86     public List<Commande> findByLivreurID(Long livreurId) throws
87     Throwable {
88         EntityManager em = JpaUtil.obtenirEntityManager();
89         List<Commande> commandes = null;
90
91         try {
92             Livreur l = em.find(Livreur.class, livreurId);
93             if(l != null)
94             {
95                 commandes = l.getCommandes();
96                 if(commandes.isEmpty())
97                 {
98                     commandes = null;
99                 }
100             }
101         } catch (Exception e) {
102             throw e;
103         }
104         return commandes;

```

```
105     }
106
107     public List<Commande> findAllNotEnded() throws Throwable {
108         EntityManager em = JpaUtil.obtenirEntityManager();
109         List<Commande> commandes = null;
110         try {
111             Query q = em.createQuery("SELECT c FROM Commande c WHERE "
112                                     + "c.dateFin IS NULL");
113             commandes = (List<Commande>) q.getResultList();
114         }
115         catch (Exception e) {
116             throw e;
117         }
118         return commandes;
119     }
120 }
121
```