```java
package fr.insalyon.dasi.gustatif.dao;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.RollbackException;

/**
 * Cette classe fournit des méthodes statiques utiles pour accéder aux
 * fonctionnalités de JPA (Entity Manager, Entity Transaction). Le nom de
 * l'unité de persistance (PERSISTENCE_UNIT_NAME) doit être conforme à la
 * configuration indiquée dans le fichier persistence.xml du projet.
 *
 * Version du 22 Mars 2016
 *
 * @author DASI Team
 */
public class JpaUtil {

    //**********************************************************************
    // * TODO: IMPORTANT -- Adapter le nom de l'Unité de Persistance (cf. persistence.xml) *
    //**********************************************************************
    /**
     * Nom de l'unité de persistance utilisée par la Factory de Entity Manager.
     * <br/><strong>Vérifier le nom de l'unité de persistance
     * (cf. persistence.xml)</strong>
     */
    public static final String PERSISTENCE_UNIT_NAME = "fr.insalyon.dasi_Gustatif_database";
    /**
     * Factory de Entity Manager liée à l'unité de persistance.
     * <br/><strong>Vérifier le nom de l'unité de persistance indiquée dans
     * l'attribut statique PERSISTENCE_UNIT_NAME
     * (cf. persistence.xml)</strong>
     */
    private static EntityManagerFactory entityManagerFactory = Persistence.createEntityManagerFactory(PERSISTENCE_UNIT_NAME);
    /**
     * Gère les instances courantes de Entity Manager liées aux Threads.
     * L'utilisation de ThreadLocal garantie une unique instance courante par
     * Thread.
     */
    private static final ThreadLocal<EntityManager> threadLocalEntityManager = new ThreadLocal<EntityManager>() {

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

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