Changes in trunk/GenericDAOJPA [231:241]

Files:
- trunk/GenericDAOJPA/lib/JPA_Hibernate
- trunk/GenericDAOJPA/nbproject/project.properties (1 diff)
- trunk/GenericDAOJPA/src/com/tecgraf/dao/IGenericDAOJPA.java (1 diff)
- trunk/GenericDAOJPA/src/com/tecgraf/dao/GenericDAOJPA.java (2 diffs)
- trunk/GenericDAOJPA/build
- trunk/GenericDAOJPA/dist

Unmodified Added Removed Modified Copied Moved

trunk/GenericDAOJPA/nbproject/project.properties

Revision 231		Revision 241	
44	javac.deprecation=true	44	javac.deprecation=true
45	javac.source=1.5	45	javac.source=1.5
46	javac.target=1.5	46	javac.target=1.6
47	javac.test.classpath=\	47	javac.test.classpath=\
48	${javac.classpath}:\	48	${javac.classpath}:\

trunk/GenericDAOJPA/src/com/tecgraf/dao/IGenericDAOJPA.java

Revision 231		Revision 241	
28	List<T> findAll(Class<T> c);	28	List<T> findAll(Class<T> c);
29		29	
30	T makePersistent(T entity);	30	T makePersistent(T entity) throws JPADAOException;
31	boolean deletePersistent(Class<T> c, T entity);	31	boolean deletePersistent(Class<T> c, T entity);
32	boolean updatePersistent(Class<T> c, T entity);	32	boolean updatePersistent(T entity);
33		33	
34	public String getEntityClassName();	34	public String getEntityClassName();

trunk/GenericDAOJPA/src/com/tecgraf/dao/GenericDAOJPA.java

Revision 231		Revision 241	
17	import javax.persistence.Query;	17	import javax.persistence.Query;
18		18	
19	/**		
20	/**	19	/**
21	* Class GenericDAOJPA is an implementation of a class that	20	* Class GenericDAOJPA is an implementation of a class that
...		...	

```
29  */                                                  28  */
30  public abstract class GenericDAOJPA<T extends IEntityDAOJPA<ID>, ID   29  public abstract class GenericDAOJPA<T extends IEntityDAOJPA<ID>, ID
    extends Number>                                         extends Number>
31          implements IGenericDAOJPA<T, ID> {          30          implements IGenericDAOJPA<T, ID>
                                                         31  {
32      //-------------------------------------------------32      //-------------------------------------------------
33      // Query methods                                 33      // Query methods
34      //-------------------------------------------------34      //-------------------------------------------------
35      public List<T> queryPositionalParams( String queryString, final   35      public List<T> queryPositionalParams(
    Object... positionalParams ) {
36          em =                                         36              String queryString,
    facadeJPA.getEntityManagerFactory().createEntityManager();
37                                                       37              final Object... positionalParams
38          Query query = em.createQuery(queryString);   38              )
                                                         39      {
                                                         40          EntityManager em =
                                                             facadeJPA.getEntityManagerFactory().createEntityManager();
                                                         41
                                                         42          Query query = em.createQuery( queryString );
                                                         43
39          int i = 0;                                   44          int i = 0;
40          for( Object p : positionalParams ) {         45          for( Object p : positionalParams )
                                                         46          {
41              query.setParameter(++i, p);              47              query.setParameter(++i, p);
42          }                                            48          }
                                                         49
43          @SuppressWarnings("unchecked")               50          @SuppressWarnings("unchecked")
44          List<T> l = query.getResultList();           51          List<T> l = query.getResultList();
45                                                       52
46          em.close();                                  53          em.close();
                                                         54
47          return l;                                    55          return l;
48      }                                                56      }
49                                                       57
50      public List<T> queryNamedParams( String queryString, final   58      public List<T> queryNamedParams(
    Map<String, Object> namedParams ) {
51          em =                                         59              String queryString,
    facadeJPA.getEntityManagerFactory().createEntityManager();
52                                                       60              final Map<String, Object> namedParams
53          Query query = em.createQuery(queryString);   61              )
54          for( Map.Entry<String, Object> entry :       62      {
    namedParams.entrySet() ) {
55              query.setParameter(entry.getKey(), entry.getValue());   63          EntityManager em =
                                                             facadeJPA.getEntityManagerFactory().createEntityManager();
56          }                                            64
                                                         65          Query query = em.createQuery( queryString );
```

		66	
		67	for(Map.Entry<String, Object> entry : namedParams.entrySet())
		68	{
		69	query.setParameter(entry.getKey(), entry.getValue());
		70	}
		71	
57	@SuppressWarnings("unchecked")	72	@SuppressWarnings("unchecked")
58	List<T> l = query.getResultList();	73	List<T> l = query.getResultList();
59		74	
60	em.close();	75	em.close();
		76	
61	return l;	77	return l;
62	}	78	}
		79	
63	//---	80	//---
64	// CRUD methods	81	// CRUD methods
65	//---	82	//---
66	public T findById(Class<T> c, ID primaryKey) {	83	public T findById(Class<T> c, ID primaryKey)
		84	{
67	Logger.getLogger(getEntityClassName()).	85	Logger.getLogger(getEntityClassName()).
68	log(Level.FINE, "lookdig for object id=" + primaryKey + " of class: " + c.getName());	86	log(Level.FINE, "lookdig for object id=" + primaryKey + " of class: " + c.getName());
69		87	
70	em = facadeJPA.getEntityManagerFactory().createEntityManager();	88	EntityManager em = facadeJPA.getEntityManagerFactory().createEntityManager();
71		89	
72	T entity = em.find(c, primaryKey);	90	T entity = em.find(c, primaryKey);
73		91	
74	em.close();	92	em.close();
		93	
75	return entity;	94	return entity;
76	}	95	}
77		96	
78	public List<T> findAll(Class<T> c) {	97	public List<T> findAll(Class<T> c)
		98	{
79	Logger.getLogger(getEntityClassName()).	99	Logger.getLogger(getEntityClassName()).
80	log(Level.FINE, "finding all objects of class: " + c.getName());	100	log(Level.FINE, "finding all objects of class: " + c.getName());
81		101	
82	entityName = c.getName().substring(c.getName().lastIndexOf('.') + 1);	102	entityName = c.getName().substring(c.getName().lastIndexOf('.') + 1);
83		103	
84	return queryPositionalParams("SELECT e FROM " + getEntityName() + " e");	104	return queryPositionalParams("SELECT e FROM " + getEntityName() + " e");

```
85      }                                                       105     }
86                                                              106
87      public T makePersistent( T entity ) {                   107     public T makePersistent(T entity) throws JPADAOException
88          em =                                                108     {
facadeJPA.getEntityManagerFactory().createEntityManager();
89                                                              
90          Logger.getLogger(getEntityClassName()).             109         Logger.getLogger(getEntityClassName()).
91                          log(Level.FINE, "persisting object: " +     110                         log(Level.FINE, "persisting object: " +
entity);                                                        entity);
92                                                              111
93          em.persist(entity);                                 112         EntityManager em =
                                                                facadeJPA.getEntityManagerFactory().createEntityManager();
                                                                113         EntityTransaction tx = em.getTransaction();
                                                                114
                                                                115         try
                                                                116         {
                                                                117             tx.begin();
                                                                118             em.persist( entity );
                                                                119             tx.commit();
                                                                120         }
                                                                121         catch(Exception ex)
                                                                122         {
                                                                123             if ( null != tx && tx.isActive() )
                                                                124             {
                                                                125                 tx.rollback();
                                                                126             }
                                                                127
                                                                128             em.close();
                                                                129
                                                                130             Logger.getLogger(getEntityClassName()).log(
                                                                Level.SEVERE, null, ex );
                                                                131
                                                                132             throw new JPADAOException( "error while persisting
                                                                object " + entity, ex );
                                                                133         }
                                                                134
                                                                135         em.close();
                                                                136
94          return entity;                                      137         return entity;
95      }                                                       138     }
96                                                              139
97      public boolean deletePersistent( Class<T> c, T entity ) {   140     public boolean deletePersistent(Class<T> c, T entity)
98          em =                                                141     {
facadeJPA.getEntityManagerFactory().createEntityManager();
```

```
99          T e = em.find(c, entity.getId()); // brings entity to the PersistenceContext
100         EntityTransaction tx = null;
101         boolean objDeleted = false;
102         if( e != null ) {
103             tx = em.getTransaction();
104             tx.begin();
105             Logger.getLogger(getEntityClassName()).
106                             log(Level.FINE, "deleting object: " + e);
107             try {
108                 em.remove(e);
109                 tx.commit();
110                 objDeleted = true;
111             } catch( RuntimeException ex ) {
112                 if( tx != null && tx.isActive() ) {
113                   tx.rollback();
114                 }
115 Logger.getLogger(getEntityClassName()).log(Level.SEVERE, null, ex);
116                 objDeleted = false;
117             } finally {
118                 em.close();
119             }
120         } else {
```

```
142         EntityManager em = facadeJPA.getEntityManagerFactory().createEntityManager();
143
144         /* brings entity to the PersistenceContext */
145         T e = em.find( c, entity.getId() );
146
147         EntityTransaction tx = null;
148         boolean objDeleted = false;
149
150         if( e != null )
151         {
152             Logger.getLogger(getEntityClassName()).
153                             log(Level.FINE, "deleting object: " + e);
154
155             tx = em.getTransaction();
156             tx.begin();
157
158             try
159             {
160                 em.remove(e);
161                 tx.commit();
162                 objDeleted = true;
163             }
164             catch(RuntimeException ex)
165             {
166                 if( tx != null && tx.isActive() )
167                 {
168                     tx.rollback();
169                 }
170
171 Logger.getLogger(getEntityClassName()).log(Level.SEVERE, null, ex);
172                 objDeleted = false;
173             }
174             finally
175             {
176                 em.close();
177             }
178         }
179         else
180         {
```

```
121             em.close();
122         }
123         return objDeleted;
124     }
125
126     public boolean updatePersistent( Class<T> c, T entity ) {
127         em = facadeJPA.getEntityManagerFactory().createEntityManager();
128         T e = findById(c, entity.getId()); // brings entity to the PersistenceContext
129
130         Logger.getLogger(getEntityClassName()).
131                         log(Level.FINE, "updating object: " + entity);
132
133         em.flush();
134
135         em.close();
```

```
181             em.close();
182         }
183
184         return objDeleted;
185     }
186
187     public boolean updatePersistent(T entity)
188     {
189         @SuppressWarnings("unchecked")
190         Class<T> c = (Class<T>) entity.getClass();
191
192         if ( null == findById(c, entity.getId() ) )
193         {
194             return ( false );
195         }
196
197         Logger.getLogger(getEntityClassName()).
198                         log(Level.FINE, "updating object: " + entity);
199
200         EntityManager em = facadeJPA.getEntityManagerFactory().createEntityManager();
201         EntityTransaction tx = em.getTransaction();
202
203         try
204         {
205             tx.begin();
206             /* brings entity to the PersistenceContext */
207             em.merge( entity );
208             em.flush();
209             tx.commit();
210         }
211         catch(Exception ex)
212         {
213             if ( null != tx && tx.isActive() )
214             {
215                 tx.rollback();
216             }
217
218             Logger.getLogger(getEntityClassName()).log(Level.SEVERE, null, ex);
219
220             em.close();
```

		221	
		222	return false;
		223	}
		224	
		225	em.close();
		226	
136	return true;	227	return true;
137	}	228	}
138		229	
139	public String getEntityClassName() {	230	public String getEntityClassName()
		231	{
140	return entityClassName;	232	return entityClassName;
141	}	233	}
142		234	
143	public String getEntityName() {	235	public String getEntityName()
		236	{
144	return entityName;	237	return entityName;
145	}	238	}
		239	
146	//--	240	//--
147	// Protected constructor	241	// Protected constructor
148	//--	242	//--
149	protected GenericDAOJPA(Class<T> c, FacadeJPA f) {	243	protected GenericDAOJPA(Class<T> c, FacadeJPA f)
		244	{
150	this.facadeJPA = f;	245	this.facadeJPA = f;
151	entityClassName = c.getName();	246	entityClassName = c.getName();
152	entityName = entityClassName.substring(entityClassName.lastIndexOf('.') + 1);	247	entityName = entityClassName.substring(entityClassName.lastIndexOf('.') + 1);
153	}	248	}
		249	
154	private String entityClassName;	250	private String entityClassName;
155	private String entityName;	251	private String entityName;
156	protected FacadeJPA facadeJPA;	252	protected FacadeJPA facadeJPA;
157	protected EntityManager em;	253	
158	}	254	}