Database - Bug #7371

proper handling of exception case in cacheExpired

05/22/2023 07:15 AM - Ovidiu Maxiniuc

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

History

#2 - 05/22/2023 07:19 AM - Ovidiu Maxiniuc

This task is about incorrect handling of the PersistenceException in orm/Session.cacheExpired().

Eric Faulhaber wrote:

> Ovidiu, the RuntimeException within the latter block concerns me. This will abend the user session. Is this what we want to do in this case?

Naturally, the PersistenceException should be propagated up to the caller, but the problem is that it cannot pass through the cache API. OTOH, The only way for the expire event to occur is by overflowing the cache by a put()/putIfAbsent() method. There is only one such call, from associate(). So I think the solution is to use the current RuntimeException for tunnelling the PersistenceException to associate() like this:

```
--- a/src/com/goldencode/p2j/persist/orm/Session.java
+++ b/src/com/goldencode/p2j/persist/orm/Session.java    (date 1684547319261)
@@ -1125,7 +1125,23 @@

       checkStale(key, dmo);

-      BaseRecord cached = cache.putIfAbsent(key, dmo);
+      BaseRecord cached = null;
+      try
+      {
+         cached = cache.putIfAbsent(key, dmo);
+      }
+      catch (RuntimeException e)
+      {
+         Throwable cause = e.getCause();
+         if (cause instanceof PersistenceException) // failure in cacheExpired()?
+         {
+            throw (PersistenceException) cause;
+         }
+         else
+         {
+            throw new PersistenceException(e); // other cause
+         }
+      }
       if (cached != null && cached != dmo)
       {
          // cannot have more than one DMO instance associated to the database with the same id
```