Base Language - Feature #1636

implement peristent trigger support

10/21/2012 11:24 AM - Greg Shah

Status:	Closed	**Start date:**	01/31/2013
Priority:	Normal	**Due date:**	
Assignee:	Constantin Asofiei	**% Done:**	100%
Category:		**Estimated time:**	24.00 hours
Target version:	Runtime Support for Server Features		
billable:	No	**vendor_id:**	GCD

Description

History

#1 - 10/31/2012 01:38 PM - Greg Shah

- Target version set to Milestone 7

#2 - 01/30/2013 04:17 PM - Greg Shah

- Assignee set to Constantin Asofiei

Just implement the conversion support for now, unless it is trivial to do the runtime too.

#3 - 01/31/2013 08:35 AM - Constantin Asofiei

The conversion support is done, and a major part of the runtime (i.e. trigger invocation). The only runtime problem remaining is the fact that the trigger must survive the procedure where it was defined (i.e. they need to be added to a global scope) - is there such a global scope in LogicalTerminal (and I guess on ThinClient too) where I can add them ?

Also, I've looked in the documentation and I've tried deleting them using the ON ... REVERT, and I don't understand if these kind of triggers can be deleted or they remain active until the user session ends.

#4 - 01/31/2013 08:52 AM - Constantin Asofiei

- File ca_upd20130131c.zip added

Adds conversion support; runtime support is missing implementation for the "persistent" nature of these triggers. Built using bzr revision 10158.

#5 - 01/31/2013 09:25 AM - Greg Shah

In regard to the global scope question for LT and TC:

1. The TriggerManager is scope-aware, but it currently has no concept of the global scope. This could be added easily.
2. The LogicalTerminal is aware of the global scope, but it doesn't do anything special related to triggers. See LT.scopeStart() and look for TransactionManager.isGlobalBlock().
3. The TC is barely scope aware. It gets scope notifications (see markFrames() and look for MarkEntry.OPEN_SCOPE and MarkEntry.CLOSE_SCOPE) which are used to do some frame-related maintenance. But as a whole, the TC doesn't keep its own scope or really do much in that regard. And I don't think you will need to add such in this case. The triggers that are active are managed on the server and when a call is made down to the client, the active trigger events are listed and sent down. For the lifetime of that primitive UI operation (e.g. UPDATE), that set of events cannot be changed, unless there is a nested trigger and recursive call down to the client, in which case a new set of active events can be sent down. But the call stack naturally unwinds all this as we return back up. Unless you find something different, it should be enough to manage this server-side.

#6 - 01/31/2013 10:31 AM - Greg Shah

Code feedback:

In registerTriggerPersistent() it says:

```
    * @param    params
    *           The parameters to be passed to the procedure (INPUT only).
...
   // each parameter is evaluated at runtime
```

Both of these comments need some clarification. The key point here is that the original parameters list is fully evaluated in the caller and is not re-evaluated at the time the trigger is called. The already-evaluated values are passed unchanged to the procedure.

Otherwise it is good.

#7 - 01/31/2013 12:16 PM - Constantin Asofiei

Some other notes related to scoping:

1. if there is a non-persistent and a persistent trigger for the same event, the local one will be executed
2. after the non-persistent trigger gets out-of-scope, the non-persistent trigger is "reactivated"

The persistent triggers I think will need to be maintained in the same ScopedDictionary as the normal ones, because we need to preserve the order in which the triggers were defined. What I think we should do is to put the persistent triggers on the scope where they are defined, and when that scope is finished, we move them one level down instead of discarding them.

#8 - 01/31/2013 12:49 PM - Constantin Asofiei

- Status changed from New to WIP

#9 - 01/31/2013 12:49 PM - Constantin Asofiei

- Start date set to 01/31/2013

#10 - 01/31/2013 02:33 PM - Greg Shah

Our ScopedSymbolDictionary has a global scope facility that should do most of the work here. The only question I have is this: is there ever a case where the persistent trigger registration is treated as if it is BOTH in the local scope and in the global scope? If so, then it may be solved by adding it to both the local scope as well as the global scope. Then the natural scoping behaviors of the dictionary should take care of the rest.

#11 - 01/31/2013 02:39 PM - Constantin Asofiei

I don't think that would do the job, as a non-persistent trigger can hide a persistent trigger; the persistent trigger will be active again only when the non-persistent trigger gets out of scope. Thus, proper chaining of trigger definitions is required (regardless of how the trigger is, persistent or non-persistent). At this time, I think the best place to keep the persistent triggers is the same dictionary as the normal triggers, with the notes that

persistent triggers defined in current scope will be moved to previous scope, when current scope ends and that we will have a global set where the id's of all persistent triggers are kept.

#12 - 01/31/2013 02:45 PM - Constantin Asofiei

PS for 11: and there is no need to save them in a global scope too, because when a persistent trigger is defined for the same events as an existing persistent trigger, the existing one gets deleted. so, saving them in the global scope will be of no help, as we will still have to walk the local scopes to remove any existing trigger for the same events.

#13 - 02/01/2013 07:50 AM - Constantin Asofiei

- File ca_upd20130201a.zip added

- File ca_upd20130201b.zip added

Attached are testcases and P2J sources which implement both conversion and runtime support for persistent triggers.

#14 - 02/01/2013 07:55 AM - Constantin Asofiei

- Status changed from WIP to Review

#15 - 02/02/2013 04:49 AM - Constantin Asofiei

Conversion regression testing has passed (no changes in the generated sources).

#16 - 02/07/2013 01:48 AM - Constantin Asofiei

Regression testing has passed, applied to bzr revision 10162.

#17 - 02/07/2013 01:48 AM - Constantin Asofiei

- % Done changed from 0 to 100

#18 - 02/07/2013 08:22 AM - Greg Shah

- Status changed from Review to Closed

#19 - 11/16/2016 11:43 AM - Greg Shah

- Target version changed from Milestone 7 to Runtime Support for Server Features

Files

ca_upd20130131c.zip	95.5 KB	01/31/2013	Constantin Asofiei
ca_upd20130201a.zip	100 KB	02/01/2013	Constantin Asofiei
ca_upd20130201b.zip	1.78 KB	02/01/2013	Constantin Asofiei