Gemstone Notifications

Implementation of an abstract Gemstone notification mechanism

presented by
Alfred Wullschleger
Swiss National Bank

The Author

- Smalltalker since 1992
- Project OVID at Fides Informatik (1992-1999)
  - OVID currently in production more than 11 years
- Project OASE at Swiss National Bank (since 1999 in production)
  - Financial Statistics from Swiss Banks and Companies
  - based on Gemstone/S and VisualWorks
  - ongoing development under full production

Gemstone Notification Basics

- a GS session declares interest in change notification for anObject by:
  - System addToNotifySet: anObject
    - whenever anObject changes and is commited (typically by another session), execution of
  - System signaledObjects
    - returns an array including anObject and the signaledObjects are cleared by this operation
    - by this you can poll for changed objects if you like

Gemstone Notification Handler

- using

– System enableSignaledObjectsError

- allows for the following Gemstone exception:

– Exception installStaticException:
[:theException :cat :num :args |
„... do here whatever is interesting... „
System enableSignaledObjectsError]
category: GemStoneError
number: (ErrorSymbols at: #rtErrSignalCommit).

Goal: Abstract Notification

- We want to have a highly standardized way of notification
  - do not use the objects, that are just there from application perspective
- it has to be fully conflict free
  - any session can commit independently
  - there can be as many notifiers as you like

Mechanisms

„Event Side“:

Objects that are committed
and want to inform
other Objects
about this

use tpzNotify as an „Event“
mediated by a TpzNotification

„Event Handler Side“:

a Session, that wants to
know, when certain Objects
are changed and commited

uses a TpzNotificationReceiver
as „Event handler“
to execute the necessary actions

complete separation of
Generation and Handling

Event Side: 2 Classes

- TpzNotification
  - notifierSymbol: defines a notification by a symbol
  - sessionValues:
    - an array of size System maxSessionId.
    - containing a TpzNotificationElement for each possible session
- TpzNotificationElement
  - has a value (anInteger) and a sessionId. The sessionId is used for quick lookup in the sessionValues

Event Side: Notification

- TpzNotification>>tpzNotify
  - (sessionValues at: System session) tpzNotify
- TpzNotificationElement>>tpzNotify
  - value := value + 1
    - could also use value := value not
- since all parallel GS sessions have different sessionIds, there cannot occur any collision
- since index lookup in the array is fast, no problem even for 1000 sessions

Handler Side: 2 Classes

- Where do we put TpzNotificationReceivers (the handlers for the notifications)?
  - into session state
  - can use index 20 (user slot in session state)
- define a holder for all notificationReceivers: TpzNotificationReceiverCollection
  - collects all notification receivers
  - implements the notification exception
  - dispatches notifications to the corresponding receivers

TpzNotificationReceiver

- Defines a process loop, which is activated by its corresponding TpzNotification
  - registerForNotification: aSymbol
    - gets the corresponding TpzNotification
    - inits the process loop
    - inserts itself into the TpzNotificationReceiverCollection
    - executes the first update

TpzNotificationReceiverCollection

- main Methods:
  - addNotificationReceiver: aNotificationReceiver
    ```
    receivers at: aNotificationReceiver class
                put: aNotificationReceiver
    ```
  - notifierSignaled
    ```
    | notelems |
    notelems := System signaledObjects.
    receivers do: [:each |
        each notifierSignaled:  notelems].
    ```
    - called from the exception

Dispatch of signaledObjects

- in TpzNotificationReceiver:
  - notifierSignaled: signaledObjects
    ```
    (tpzNotification hasSignaledObjects: signaledObjects)
        ifFalse: [signalingInformation := nil. ^self].
    signalingInformation := signaledObjects.
    processSemaphore signal
    ```
- in TpzNotification:
  - hasSignaledObjects: signaledObjects
    ```
    signaledObjects detect: [ :elem |
    (sessionValues at: elem sessionId ) == elem ]
        ifNone: [ ^false ].
    ^true
    ```
  - is quick, since typically few signaledObjects

Receivers for specific Notifications

- each specific receiver is a subclass of TpzNotificationReceiver
  - must implement 2 methods:
    - class side: notifierSymbol
      - to declare, which Notification is accepted by this receiver
    - instance side: executeUpdateFromNotification
      - the activation of the code for which the notification is all about

Demo

- We have two Handlers for the same notifierSymbol #SimpleNotification:
  - SimpleNotificationHandler
    - waits 15 seconds, then returns a string
  - SecondSimpleNotificationHandler
    - waits 5 seconds, then returns some other string
  - during the waiting, the results of the handlers are nil

Thank you for listening

Questions?