Bugs - Bug #3057

bad synchronization in AsyncRequestImpl

04/05/2016 03:11 PM - Ovidiu Maxiniuc

Status:	Closed	**Start date:**	
Priority:	Low	**Due date:**	
Assignee:	Ovidiu Maxiniuc	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	Cleanup and Stablization for Server Features		
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

Related issues:

Related to Base Language - Feature #2181: implement SSL support for sockets	**Closed**

History

#1 - 04/05/2016 03:21 PM - Ovidiu Maxiniuc

I found the following issue in AsyncRequestImpl.java: the class is (too) heavily synchronized.

The synchronized void execute(boolean event) method is the one that will spend a lot of time with the sync lock acquired. It runs on a temporary AssociatedThread. It calls request.run() and, if the call is not canceled, it sets the complete flag to true (in finally clause).

The question:
How can another / the main thread know if the request if completed or not?
The isComplete() method is also synchronized, which means it will not be effectively called until the lock is released, that is, at the end of execute() method. But, at that moment, the complete flag is always set to true. Eventually, the current thread is also blocked until the asynchronous request ends.

The other synchronized methods also are not working properly. If isCancelled() / isQuit() and isStop() somehow make sense only to be called when the execute() is already finished, they will at least block the current thread until this execute() releases the lock if not already finished. The stop() method is useless as it will be able to execute only after the execute() finishes so there is nothing to stop.

#2 - 04/05/2016 03:39 PM - Ovidiu Maxiniuc

To duplicate the issue I used an asynchronous call to a (slow) web-service and stop/ disconnect before it has the time to return successfully:

```
RUN CelsiusToFahrenheit IN hPortType
   ASYNCHRONOUS SET hAsync EVENT-PROCEDURE "c2fCallback" IN THIS-PROCEDURE
   (INPUT cValue, OUTPUT fValue).
[...]
hWebService:DISCONNECT().
```

The called service, intentionally sleeps for a few seconds before returning the result (or this can be achieved suspending the process at a breakpoint).

#3 - 04/06/2016 05:18 AM - Constantin Asofiei

I agree, the AsyncRequestImpl is too synchronized. I think is enough to set volatile for the complete, cancelled, quit, stop, error, valid and requestId fields and remove the synchronized from the methods.

Also, stop() will have to synchronize the wait() block, as in:

```
      while (!complete)
      {
         synchronized (this)
         {
            try
            {
               this.wait();
            }
            catch (InterruptedException e)
            {
               break;
            }
         }
      }
```

#4 - 04/20/2017 08:20 AM - Ovidiu Maxiniuc

- Priority changed from Normal to Low

- Assignee set to Ovidiu Maxiniuc

- Status changed from New to WIP

The update for this issue was part of branch 3130a.
It passed conversion testing on devsrv01 and ETF on local workstation. It was merged to trunk as revision 11149.

#5 - 04/20/2017 08:21 AM - Greg Shah

Can I close this?

#6 - 04/20/2017 08:24 AM - Ovidiu Maxiniuc

Greg Shah wrote:

> Can I close this?

Yes, please.

#7 - 04/20/2017 08:25 AM - Greg Shah

- Status changed from WIP to Closed

- Target version set to Cleanup and Stablization for Server Features

- % Done changed from 0 to 100