Bugs - Bug #1577

Server CPU resource rise up to 100% when admin client is closed

10/05/2012 11:33 AM - Ovidiu Maxiniuc

Status:	Closed	**Start date:**	10/05/2012
Priority:	Normal	**Due date:**	
Assignee:	Ovidiu Maxiniuc	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#1 - 10/05/2012 11:35 AM - Ovidiu Maxiniuc

Each time an admin applet logs off or is forcefully disconnected (by closing the browser tab or network fail), one thread handling the client enter a continuous loop taking all available CPU. After 4 or more disconnects the server's java process takes 400% of a quad core CPU making the PC very slow.
The only solution is to kill the process and restart the server again.

#2 - 10/05/2012 11:48 AM - Ovidiu Maxiniuc

I successfully managed to identify the thread that was causing the CPU busy. Here is the call-stack:

```
Daemon Thread [Reader [00000001:admin]]
    DirectoryService.unbind() line: 4494
    DirectoryService$BindRef.cleanup() line: 5384
    DirectoryService$2.cleanup(DirectoryService$BindRef) line: 154
    DirectoryService$2.cleanup(Object) line: 150
    ContextLocal$Wrapper<E>.cleanup() line: 358
    SecurityContext.cleanup() line: 437
    SecurityManager.endContext(SecurityContextStack, SecurityContext) line: 6710
    SecurityManager.popContextWorker() line: 6648
    SecurityManager.popAndRestoreSecurityContext() line: 3885
    RouterSessionManager.restoreContext() line: 1041
    Queue.stop(Exception, boolean) line: 414
    Protocol$Reader.run() line: 416
    Thread.run() line: 662
```

I identified where the loop happens: in the while from com.goldencode.p2j.directory.DirectoryService.java:5384.
The DirectoryService.unbind() method called two lines bellow does not reach the line 4497 (if (bindRef != null && --bindRef.count == 0)) so the counter on which the while is looping is never decremented.

#3 - 10/05/2012 11:52 AM - Ovidiu Maxiniuc

- Assignee set to Ovidiu Maxiniuc

#4 - 10/05/2012 11:53 AM - Ovidiu Maxiniuc

- Status changed from New to WIP

#5 - 10/11/2012 02:15 AM - Constantin Asofiei

This bug was exposed by the #1455 changes. Problem is, with #1455, any Cleanable.cleanup code must not rely on context-local variables, as at the time of this call, the context-local variables were deleted from the SecurityContext (by the SecurityContext.cleanup code).

The problem with DirectoryService.unbind (called by BindRef.cleanup) is that it needs access to two context-local variables - bound and activeBatch. So, we need access to the current values for both these variables, when DS.unbind is called.

The solution is made of two parts. First, we need to group these two variables in a WorkArea (implements Cleanable), which will have two instance fields, one BindRef (for bound variable) and one BatchRef (for activeBatch variable). Thus, cleanup will be called only once (for the WorkArea instance) which will do the same work as BindRef.cleanup.

The second part is to add a private unbindWorker method which does not use the context-local variables. Instead, the unbindWorker will receive as parameters a BindRef instance and a BatchRef instance. When called from DS.unbind, these will be set to the values obtained from the WorkArea context-local variable. When called from WorkArea.cleanup, these will be set to the instances referenced by the WorkArea.bound and WorkArea.activeBatch fields.

#6 - 10/11/2012 08:24 AM - Greg Shah

I am fine with the proposed changes. In preparing the changes, please make sure about the following:

1. Leave behind very clear comments that explain why the code is implemented this way AND which highlight aspects that cannot be changed back without causing a problem. I don't want someone coming along later and "simplifying" the code, thus causing this problem again.

2. I wonder if our documentation suitably documents these problems. I can think of 2 places: JavaDoc (for the ContextLocal, Cleanable...) and the "Context-Local Data" section of the "Runtime Hooks and Plugins" chapter of the Developer Guide. Please review those carefully and enhance them where needed to make it clear how to use context-local safely. Note that Stanislav is about to check in a new version of "Runtime Hooks and Plugins" so you had better coordinate any changes with him.

3. This problem opens the question: if our context-local support is so sensitive to poor implementations, then what can be done to improve the design or implementation of it to make it less vulnerable? Please provide more information as to why we were consuming 100% CPU. I assume there was some failure in cleanup and then the same object is called for cleanup again in an endless loop? Or was something else happening? I suspect we can make our context-local support better to avoid the 100% utilization and limit the damage to a failure of the cleanup process (when the developer implements it poorly).

Finally, Constantin will review/approve the changes for check in. The changes will have to go through regression testing first, like anything else.

#7 - 10/16/2012 10:07 AM - Constantin Asofiei

Ovidiu,

Here is the review result:
- please name the update file all lowercase
- the update should have a structure so that, assuming you have a ~/work/p2j folder, you can place the update in the ~/work/ folder and unzip it. So, please put the root com/ folder you have in your current update in a p2j/src/ root folder (so that the update is p2j/src/com/goldencode/etc)
- please change the year in the copyright date for all files, if needed, as in:
Copyright (c) 2005-2010, Golden Code Development Corporation.
becomes
Copyright (c) 2005-2012, Golden Code Development Corporation.
- please improve the javadoc for DirectoryService.workArea so that it has no reference of redmine cases (is best to explicitly say why it was added).
- you have some typos in javadoc for unbind(workarea)
- code formatting issues:
- keep the lines wrapped to 78 chars
- when comparing, keep the constant on the right side, i.e. object != null
- add javadoc to document the new parameter for BindRef.cleanup and BatchRef.cleanup
- on line 1829 you've added a local var which is not needed

#8 - 10/18/2012 07:45 AM - Constantin Asofiei

Update passed regression testing, applied to staging P2J and rebuilt.

#9 - 10/18/2012 08:04 AM - Ovidiu Maxiniuc

- File om_upd20121016b.zip added

- Status changed from WIP to Review

Update committed to CVS.

#10 - 10/22/2012 02:10 PM - Greg Shah

- Status changed from Review to Closed

#11 - 11/01/2012 04:51 PM - Greg Shah

- File om_upd20121016a.zip added

Files

om_upd20121016b.zip	31.3 KB	10/18/2012	Ovidiu Maxiniuc
om_upd20121016a.zip	30.7 KB	11/01/2012	Greg Shah