Runtime Infrastructure - Bug #8383

Some JsonObject methods create a LOG-MANAGER record in silent

03/04/2024 07:32 AM - Galya B

Status:	Review	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Galya B	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

Related issues:

Related to Base Language - Feature #4384: Builtin OO Implementation	**Closed**

History

#1 - 03/04/2024 07:36 AM - Galya B

- Subject changed from JsonObject throws in silent with some errors to Some JsonObject methods create a LOG-MANAGER record in silent

com.goldencode.p2j.oo.json.objectmodel.JsonObject throws errors using undoThrow, that is explicit LegacyErrorException and the logs get recorded by LOG-MANAGER even with no-error.

Reproducible in testcases with:

```
using Progress.Json.ObjectModel.JsonObject from propath.

define variable jsonObj as JsonObject.
define variable objRes as JsonObject.

jsonObj = new JsonObject().

objRes = jsonObj:GetJsonObject("no_name":U) no-error.
```

produces the log: (Procedure: 'GetJsonObject Progress.Json.ObjectModel.JsonObject') Call to Progress.Json.ObjectModel.JsonObject:GetJsonObject( ) failed. Property 'no_name' was not found. (16058)

#3 - 03/14/2024 03:54 AM - Galya B

- Related to Feature #4384: Builtin OO Implementation added

#5 - 03/14/2024 04:49 AM - Galya B

- Status changed from New to WIP

- Assignee set to Galya B

#6 - 03/14/2024 05:28 AM - Galya B

TransactionManager does a trick that is not taken into consideration in BlockManager.processLegacyError, when logging errors:

```
         // disable silent mode if it was enabled because silent mode on a
         // RUN statement or for a function call is something that only
         // suppresses errors during the dispatching itself and NOT during the
         // execution of the called code
         if (blk.suppressError)
         {
            wa.em.setSilent(false);
         }
```

#7 - 03/14/2024 05:29 AM - Galya B

I'll have to try and find examples of code to test the different branches of processLegacyError to be able to 100% cover similar issues.

#8 - 03/14/2024 06:52 AM - Galya B

There is another issue: The error is logged twice (no matter if silent or not). That is because BlockManager.processLegacyError is first called by the functionBlock and then by the topLevelBlock: both executions have different paths, but both log the error. Truth be told I have no memory what exceptions were tested initially with implementing LOG-MANAGER that lead to introducing those log writes.

#9 - 03/15/2024 07:18 AM - Galya B

To run tests:

- oo/progress/lang/test_apperror.p
- oo/json/object/get_json_object.p

#10 - 03/15/2024 08:05 AM - Galya B

- Status changed from WIP to Review

- % Done changed from 0 to 100

8383a based on trunk r15055.

8383a r15058 ready for review.

I've decided on an imperfect solution that will hopefully work for all cases. Since I'm not able to find criteria when an exception will be handled by two (or more) blocks, I've just added a flag to mark when the error is already displayed and not repeat it. There are a few more issues solved, these are regressions I've found in my test procedures: error message added with AddMessage was not displayed (and I know it worked in the past), so here it is back. Also for some reason it seems the callstack has the procedure name in the last instead of the first entry, at least in the JsonObject example, so it's fixed as well.

#11 - 04/01/2024 05:09 AM - Constantin Asofiei

Galya, some notes:

- we need to be careful with naming FWD internal methods added to skeleton classes/interfaces (in p2j.oo) - in LegacyError, isDisplayed and setDisplayed names may collide to some app-level implementation of the AppError. I don't have a good solution for this, other name prefixing them with an underscore.
- in OE, SysError is used for internal errors (via ErrorManager for FWD). But, there is nothing stopping the application saving the error instance in a variable and re-throwing it. The point is: the instance can be re-used, and having the 'displayed' flag at the instance doesn't look OK.

Also, what files exactly do I need to convert to run the test_apperror.p test?

#12 - 04/01/2024 05:21 AM - Galya B

- Status changed from Review to WIP

- % Done changed from 100 to 0

Constantin Asofiei wrote:

> - in OE, SysError is used for internal errors (via ErrorManager for FWD). But, there is nothing stopping the application saving the error instance in a variable and re-throwing it. The point is: the instance can be re-used, and having the 'displayed' flag at the instance doesn't look OK.

In this case the solution won't work. It's impossible to follow all the routes of errors in FWD and reason about solutions that work for all, that's why old functionality gets broken often. The fact that an error is thrown, caught, re-thrown, caught on another level is bothersome.

> Also, what files exactly do I need to convert to run the test_apperror.p test?

Probably:

```
oo/progress/lang/apperror/test_catch.p
oo/progress/lang/apperror/test_constructor.p
oo/progress/lang/apperror/test_methods.p
oo/progress/lang/apperror/test_no_error.p
oo/progress/lang/apperror/test_properties.p
```

After having a better look at the test I think it's not related to the changes. Nevertheless I ran it just to check if any random regression didn't appear, since it was the closest.

#13 - 04/01/2024 08:02 AM - Galya B

Constantin Asofiei wrote:

- in OE, SysError is used for internal errors (via ErrorManager for FWD). But, there is nothing stopping the application saving the error instance in a variable and re-throwing it. The point is: the instance can be re-used, and having the 'displayed' flag at the instance doesn't look OK.

On second thought is a caught - re-thrown error supposed to be displayed twice automatically? Can you think of such example?

#14 - 04/15/2024 03:45 AM - Galya B

- Status changed from WIP to Review

- % Done changed from 0 to 100

Constantin Asofiei wrote:

> Galya, some notes:
>
> - we need to be careful with naming FWD internal methods added to skeleton classes/interfaces (in p2j.oo) - in LegacyError, isDisplayed and setDisplayed names may collide to some app-level implementation of the AppError. I don't have a good solution for this, other name prefixing them with an underscore.

r15059 up, renaming the methods to have underscore.

- in OE, SysError is used for internal errors (via ErrorManager for FWD). But, there is nothing stopping the application saving the error instance in a variable and re-throwing it. The point is: the instance can be re-used, and having the 'displayed' flag at the instance doesn't look OK.

I really can't think of a way to throw the same error twice without catching it. When the error is caught or in silent, it is not supposed to be displayed. When the error is thrown, but not caught, it's displayed, but it ends the execution of the application, so there is no chance of re-throwing / re-displaying it. Do you have something else in mind?

This example adds only one log in OE:

```
block-level on error undo, throw.
using Progress.Json.ObjectModel.JsonObject from propath.

define variable jsonObj as JsonObject   no-undo.

jsonObj = new JsonObject().

do on error undo, throw:
  jsonObj:GetJsonObject("non-existent-key").
  catch e as Progress.Lang.Error :
    message "in catch".
    undo, throw e.
  end catch.
end.
```

#15 - 04/25/2024 04:14 AM - Constantin Asofiei

Galya, 8383a/15059 does not compile.

#16 - 04/25/2024 04:24 AM - Galya B

Constantin Asofiei wrote:

> Galya, 8383a/15059 does not compile.

The notorious gradlew all doesn't work, while gradlew ant-jar is fine after the rename.

Fix in r15060.