Base Language - Bug #5025

syserror, jsonerror...

11/27/2020 10:34 AM - Greg Shah

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

Description

History

#1 - 11/27/2020 10:35 AM - Greg Shah

From Marian:

> In 4GL the JSON parser is throwing JsonError/JsonParserError which is something one can't instantiate from 4GL since the c'tors there are all private – same for the other 'internal' errors like SysError. I'm trying to change the JSON parser in FWD to throw the same kind of exceptions and started by throwing SysError instead of JsonError because there we already have c-tors available.
>
> Looking at what valid means it looks to me we can simply define those errors as not being 'tracked' so always considered valid, as such we do not need those 4GL style c-tors or we can just leave them as private to get the same 'reflection' data but other that that we can create instances from Java just fine. Does this make sense? Otherwise leaving the c-tors as public will just need to be adjusted for reflection, no 4GL code allows calling those c-tors anyway because of syntax error.

#2 - 11/27/2020 10:36 AM - Greg Shah

- File sys_error.patch added

From Marian:

> Something along the lines… have tracked turned off for SysError so all instances are always valid and then drop the 4GL style c-tors and instantiate the objects normally not using ObjectOps.newInstance.
>
> This seems to work as far as I've could test but there might be something obvious that I'm missing

#3 - 11/27/2020 10:42 AM - Greg Shah

Using isTracked() is a good solution for any object that MUST stay out of the FIRST-OBJECT/LAST-OBJECT linked list. If these errors never appear in the object list, then this part should be OK. Likewise, any non-tracked objects do not need the standard constructor processing.

I don't have an objection to using protected Java constructors.

Constantin: thoughts?

#4 - 12/02/2020 03:10 AM - Marian Edu

Greg Shah wrote:

> Using isTracked() is a good solution for any object that MUST stay out of the FIRST-OBJECT/LAST-OBJECT linked list. If these errors never appear in the object list, then this part should be OK. Likewise, any non-tracked objects do not need the standard constructor processing.

Argh, then this might not be the right approach since all objects including system errors are present in that linked list :(

On the other hand, simply wrapping the system error we're instantiating in an ObjectResource seems to `interlink` the object into the chain.

#5 - 12/04/2020 08:48 AM - Greg Shah

- Assignee set to Constantin Asofiei

#6 - 12/17/2020 08:36 AM - Marian Edu

Can I go ahead with using protected ctor's and then simply wrap the new instance in an ObjectResource just to add it to the session objects list or we stick to using 4GL ctor's and then hide those in reflection if needed?

Error messages localization is done for system errors so maybe SysError is the place to handle something like that if ever needed (maybe Dutch error messages).

#7 - 12/17/2020 01:12 PM - Constantin Asofiei

Marian, regarding the JsonError/JsonParserError and SysError in general - I think the correct solution is to use 4GL-style 'execute' and 'constructor' Java methods, and create this instance via ObjectOps.newInstance - add static helpers like SysError.newInstance to hide some of the overhead.

Why I say this: legacy object instances in 4GL act like pseudo-persistent-programs - and they act like this in FWD, too. We have some reference counting and other mechanisms built on top of the persistent procedure support, but otherwise they are almost the same. You can make these 'constructors methods' private, as FWD can invoke them safely. Also, in FWD the only correct mechanism to create a new instance is via ObjectOps.newInstance - never use the Java new operator.

If we are creating instances without setting up all the needed state - bypassing ObjectOps.newInstance and the 'execute/constructor' methods- then we can get in trouble. See what is done in ProcedureManager$WorkArea.scopeStart, ObjectOps.newInstance and others. You don't need to explicitly create a ObjectResource, as ObjectOps will take care of this.

Also, for the instance to survive the 4GL-style block where it was created, it must be assigned to something outside of it, like a real 4GL-style variable, a temp-table field or output parameter. This is all hidden via ObjectVar class, e.g. TypeFactory.object when declaring a 4GL-style object variable.

#8 - 12/18/2020 01:45 AM - Marian Edu

Constantin Asofiei wrote:

> Marian, regarding the JsonError/JsonParserError and SysError in general - I think the correct solution is to use 4GL-style 'execute' and 'constructor' Java methods, and create this instance via ObjectOps.newInstance - add static helpers like SysError.newInstance to hide some of the overhead.

Understood, if we can make those ctor's private (probably protected in syserror though) then think this is as close as possible to the 4GL implementation.

#9 - 01/27/2021 05:52 AM - Marian Edu

SysError and all others extending it keeps the 4gl style ctor's and instances are created the usual. No 4GL code that tries to instantiate any of those classes will compile so most probably those ctors will only be called from FWD internals. Maybe a new annotation can be added to LegacySignature so we can actually hide those ctors in reflection using a generic way instead of special handling for those classes.

Not sure how to 'close' this, setting status to 'feedback' just doesn't seems right :)

#10 - 01/27/2021 10:44 AM - Greg Shah

- % Done changed from 0 to 100

Constantin: I will close this unless you think there is some remaining cleanup item or feature/optimization to pursue.

#11 - 01/27/2021 01:06 PM - Constantin Asofiei

I'm OK with closing it.

#12 - 01/27/2021 01:17 PM - Greg Shah

- Status changed from New to Closed

Files

sys_error.patch	5.61 KB	11/27/2020	Greg Shah