Base Language - Feature #6410

implement additional built-in OO classes/interfaces

05/25/2022 03:49 PM - Greg Shah

Status:	WIP	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Marian Edu	**% Done:**	90%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to Base Language - Feature #4373: finish core OO 4GL support	**New**
Related to Base Language - Bug #7460: implement Progress.Lang.Class:getMethods	**Merge Pending**

History

#1 - 05/25/2022 03:50 PM - Greg Shah

Complete the support for the following classes:

- openedge.core.collections.list (partial/stubs)
- openedge.core.collections.stringcollection (partial/stubs)
- openedge.core.memptr (full/partial)
- openedge.core.string (none/none)
- openedge.core.system.argumenterror (none/none)
- openedge.core.timestamp (none/none)
- openedge.net.http.credentials (full/stubs)
- openedge.net.http.ihttpclientlibrary (partial/partial)
- openedge.net.serverconnection.clientsocket (full/stubs)
- openedge.net.uri (full/partial)
- progress.json.objectmodel.jsonarray (full/partial)
- progress.lang.class (partial/partial)
- progress.lang.parameterlist (full/basic)
- progress.lang.proerror (full/basic)
- progress.reflect.method (full/stubs)

#2 - 05/25/2022 03:51 PM - Greg Shah

- Related to Feature #4373: finish core OO 4GL support added

#3 - 11/02/2022 08:36 AM - Greg Shah

- Assignee set to Marian Edu

The biggest priority here is to achieve full conversion support + runtime stubs for all the above cases. This will enable conversion for a very large application which we are working on now.

#4 - 11/02/2022 08:37 AM - Greg Shah

Support should be added in task branch 6129b.

#5 - 11/02/2022 08:49 AM - Marian Edu

Greg Shah wrote:

> Support should be added in task branch 6129b.

What is the target OE version?

#6 - 11/02/2022 08:52 AM - Greg Shah

OE 11.7.x is in use right now. They can't move to OE 12 without facing serious issues which have not yet been resolved.

#7 - 12/14/2022 09:09 AM - Marian Edu

Greg Shah wrote:

> OE 11.7.x is in use right now. They can't move to OE 12 without facing serious issues which have not yet been resolved.

Greg, should I go ahead with the plan of using PROVERSION for the 4GL OO code base? I hope we can use the same skeleton even if properties/methods were only added in the newest OE versions or we can use PROVERSION with &IF preprocessor in that case.

#8 - 12/14/2022 09:15 AM - Greg Shah

Yes, please use PROVERSION and preprocessing to conditionally implement logic by OE version.

#9 - 12/28/2022 07:27 AM - Marian Edu

Greg Shah wrote:

> Yes, please use PROVERSION and preprocessing to conditionally implement logic by OE version.

Is this the right repo for skeleton - https://proj.goldencode.com/downloads/skeletons/repo?

I'm trying to use preprocessor directives for added/changed methods in those provided classes, there are classes that are added later only and were not present in 11.7, should I put all class body in preprocessor for them? So it looks like the class is still there only there is no body?

On the other hand there are difference in functionality sometimes so there we will probably need to use the proversion function so runtime not compile time since there will only be one FWD implementation is it?

#10 - 02/01/2023 08:52 AM - Marian Edu

- Status changed from New to WIP

#11 - 02/02/2023 03:12 AM - Marian Edu

Greg Shah wrote:

> Yes, please use PROVERSION and preprocessing to conditionally implement logic by OE version.

Greg, we do not have all OE versions available to know exactly what was changed in each version - I have now the skeleton classes generated for 11.7 and 12.2. I can start from 11.7 as baseline and use preprocessor using the major version number as differentiator, something like this:

```
&if entry(1, proversion, '.') = '12' &then
message 'oe12' view-as alert-box.
&else
message 'other' view-as alert-box.
&endif
```

Now, this only has effect at compile time - for FWD will be when the conversion is executed and that normally works fine with 4GL code that is actually converted but afraid the skeleton code doesn't fall into that category so I'm not sure using preprocessor statements in skeleton even makes sense but maybe I'm missing something.

I was looking at the differences between generated skeleton classes for OE11.7 and OE12.2 and while new members added to classes could be somehow implemented using conditional processing in OO implementation (would probably have to be done at runtime) there are some cases that I don't see how we could accommodate with only one 'version independent' implementation. For instance the inheritance chain changed in `ABLUnit` packages (`Model` and `Reflection`) where existing classes now extends other base classes or implement new interfaces. Say we somehow have a 'least common multiple' in FWD implementation so we have all members available in all OE versions (supported) and we manage using runtime version check to mimic the same functionality as in the target OE version, there will still be issues with reflection/type-of/cast because of this potential change in the inheritance.

Right now I don't think `polluting` the implementation with conditional code to support different OE version in one code base is the best idea, why don't you go with the same approach as PSC on this one and have it as an add-on (really separate project out of FWD core) and implement OE versions as needed using separate branches? Once a OE version ships the branch is sealed anyway, one can still 'upgrade' to another (newer) version proven this one don't use some new version specific additions that makes it incompatible with the previous OE core.

Please let me know which way you like us to proceed, just though to mention that (again) although I know you said you don't like version specific implementation.

#12 - 02/02/2023 09:25 AM - Greg Shah

11.7 and 12.2 is fine.

> Now, this only has effect at compile time - for FWD will be when the conversion is executed and that normally works fine with 4GL code that is actually converted but afraid the skeleton code doesn't fall into that category so I'm not sure using preprocessor statements in skeleton even makes sense but maybe I'm missing something.

In FWD terms, the preprocessor affects parsing and the skeletons are ONLY used for parsing. The idea here is to use the preprocessor to represent the API properly between versions so we can properly parse an application that is dependent upon the API in a specific OE version.

> For instance the inheritance chain changed in `ABLUnit` packages (`Model` and `Reflection`) where existing classes now extends other base classes or implement new interfaces. Say we somehow have a 'least common multiple' in FWD implementation so we have all members available in all OE versions (supported) and we manage using runtime version check to mimic the same functionality as in the target OE version, there will still be issues with reflection/type-of/cast because of this potential change in the inheritance.

I understand that PSC has made poor decisions on things and then broken the API contract over time to take a different approach. We will have to take each case as it comes.

Right now, we are primarily focused on 11.7 because the customers using these features are on that version. The 12.x stuff does not matter as much (or at all), yet. I'm sure that someday (maybe soon), it will matter when there are more 12.x customers.

> Right now I don't think `polluting` the implementation with conditional code to support different OE version in one code base is the best idea, why don't you go with the same approach as PSC on this one and have it as an add-on (really separate project out of FWD core) and implement OE versions as needed using separate branches?

I prefer not to have different FWD branches for different OE versions. Do we really need to support 12.x right now?

#13 - 02/02/2023 09:41 AM - Marian Edu

Greg Shah wrote:

> I prefer not to have different FWD branches for different OE versions.

Understood, then only one version can do it but imho it worth nothing for FWD to mimic multiple versions of OE for those OO classes provided by PSC.

> Do we really need to support 12.x right now?

If there are no customers using it then most probably no. We still have OE11.7 so we can work with that version but as far as I remember the implementation was updated to 12.2, I see Vladimir already moved the assertions to 12.6 so will try to see how far can we get using preprocessors but we can then keep OE11.7 as main target and OE12.2 'extensions/fixes' are only nice to have?

#14 - 02/02/2023 09:58 AM - Greg Shah

> we can then keep OE11.7 as main target and OE12.2 'extensions/fixes' are only nice to have?

Correct. If there are incompatibilities, please keep a careful list of the things we will have to rework later.

We do have the "luxury" of detecting problems at conversion time and mapping the resulting code completely differently at runtime. This may indeed be a problem for the casting case and it is nasty because it means we have to duplicate some amount of the class hierarchy, but it can be done. We will defer this problem until later.

#15 - 02/21/2023 09:13 AM - Marian Edu

Greg, is there any update on method/ctor resolution during conversion in regard with Java erasure issue caused by the `object` parameters/return type? For ctor's this doesn't seem to work at all and for methods the order of methods in the 4GL source code seems to be important so I would think updating the skeleton might need a full application code reconversion.

I've made a small change in the conversion rule for handling *poly* as method return type so I could get the generated skeleton to convert but for ctor erasure issue I remember you said something about reworking the whole thing at some point so thought to check with you first.

#16 - 02/21/2023 09:36 AM - Greg Shah

> is there any update on method/ctor resolution during conversion in regard with Java erasure issue caused by the `object` parameters/return type? For ctor's this doesn't seem to work at all and for methods the order of methods in the 4GL source code seems to be important so I would

> think updating the skeleton might need a full application code reconversion.

Are you working off of trunk rev 14484?

#17 - 02/21/2023 09:46 AM - Marian Edu

Greg Shah wrote:

> Are you working off of trunk rev 14484?

Conversion was ran with trunk #14483, working on 6129b with changes.

#18 - 02/21/2023 10:05 AM - Greg Shah

> For ctor's this doesn't seem to work at all

6129b which is now in trunk revision 14484 does have all my previous work on method overloading. C'tor processing is still an issue.

> for methods the order of methods in the 4GL source code seems to be important so I would think updating the skeleton might need a full application code reconversion

Yes, this order is important. We probably should eliminate this dependency but it will require a different algorithm for name/signature conflict resolution.

#19 - 02/24/2023 04:30 AM - Marian Edu

Greg Shah wrote:

> 6129b which is now in trunk revision 14484 does have all my previous work on method overloading. C'tor processing is still an issue.
> Yes, this order is important. We probably should eliminate this dependency but it will require a different algorithm for name/signature conflict resolution.

Was thinking about this, the only issue right now is in JsonArray ctor that can takes either an array of JsonObject or JsonArray so the erasure on object must be solved somehow... but, I imagine in real life application code this could be a common scenario as well.

I'm sure this might have been considered before, early mornings thoughts only looks good at times, but thought to mention it just in case... the basic idea would be to generate (during conversion) simple wrappers for OO classes that only extends object (which can be made abstract) for a specific PLO type and then use those wrappers in ctor/method signature during conversion. Of course this will require the same handling for when a variable/property of given type is defined/instantiated but it will probably alleviate (if not completely fix) the whole mapping processing during conversion.

```
public class JsonArrayRef extends object<JsonArray> { ... }

public class JsonObjectRef extends object<JsonObject> { ... }

...
public void __progress_json_objectmodel_JsonArray_constructor__(final JsonObjectRef[] _objectRef) { ... }
public void __progress_json_objectmodel_JsonArray_constructor__(final JsonArrayRef[] _objectRef) { ... }
```

#20 - 04/04/2023 10:33 AM - Greg Shah

Marian: Please summarize the current level of conversion support for all of the classes #6410-1. We need all of these to be finished (conversion support with runtime stubs) by the end of April 2023.

#21 - 04/05/2023 07:01 AM - Marian Edu

Greg Shah wrote:

> Marian: Please summarize the current level of conversion support for all of the classes #6410-1. We need all of these to be finished (conversion support with runtime stubs) by the end of April 2023.

Greg, will pick that up tomorrow and should be done by end of next week for sure, just have to check if I've lost something after the server crash :(

That ctor erasure issue remains for JsonArray, guess my proposals doesn't make much sense so not sure how to get around this one right now.

#22 - 04/27/2023 12:28 PM - Greg Shah

Branch 6410a has been branched from trunk 14552. I've pushed it to xfer (/opt/fwd/6410a).

#23 - 05/01/2023 05:57 AM - Constantin Asofiei

7199c rev 14559 contains the implementation for OpenEdge.Core.System.ArgumentError. Required by #7199

#24 - 05/02/2023 02:46 AM - Marian Edu

When converting skeleton classes there seems to be some issue with classes that implements interfaces with properties, only if the interface is part of `OESDK`, if the interface is also part of application code the conversion works fine.

This works:

```
interface convert.ITest:
   define public property Connected as logical no-undo
      public get.
end.

class convert.TestImpl implements convert.ITest:
   define public property Connected as logical no-undo
      public get.
end.
```

This doesn't convert, the skeleton interface (`ServerConnection`) has the property Connected:

```
class convert.TestImpl implements OpenEdge.Core.ServerConnection.IServerConnection:
   define public property Connected as logical no-undo
      public get.
end.
```

When converting this it gives the following error and it fails to convert:

```
Failure in file './convert/TestImpl.cls':
com.goldencode.ast.AstException: Error processing ./convert/TestImpl.cls
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:1025)
    at com.goldencode.p2j.uast.ScanDriver.lambda$0(ScanDriver.java:398)
    at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:434)
    at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:263)
    at com.goldencode.p2j.convert.TransformDriver.runScanDriver(TransformDriver.java:390)
    at com.goldencode.p2j.convert.TransformDriver.front(TransformDriver.java:255)
    at com.goldencode.p2j.convert.TransformDriver.executeJob(TransformDriver.java:986)
    at com.goldencode.p2j.convert.ConversionDriver.main(ConversionDriver.java:1284)
    at testcases.Convert.main(Convert.java:22)
Caused by: java.lang.NullPointerException: null value for annotation 'set-javaname':define [DEFINE_PROPERTY]:1
546188226574 @2:4
   public [KW_PUBLIC]:1546188226576 @2:11
   Connected [SYMBOL]:1546188226580 @2:27
   as [KW_AS]:1546188226582 @2:37
      logical [KW_LOGICAL]:1546188226584 @2:40
   no-undo [KW_NO_UNDO]:1546188226586 @2:48
   get [KW_GET]:1546188226588 @3:14
      public [KW_PUBLIC]:1546188226594 @3:7

    at com.goldencode.ast.XmlFilePlugin.writeSingleAnnotation(XmlFilePlugin.java:1154)
    at com.goldencode.ast.XmlFilePlugin.writeAnnotations(XmlFilePlugin.java:1115)
```

For what is worth, changing the name of property in class solves the issue.

#25 - 05/02/2023 02:49 AM - Marian Edu

Constantin Asofiei wrote:

> 7199c rev 14559 contains the implementation for OpenEdge.Core.System.ArgumentError. Required by #7199

Can that be pushed in 6410a on xfer?

#26 - 05/02/2023 03:06 AM - Constantin Asofiei

Marian Edu wrote:

> Constantin Asofiei wrote:
>
> > 7199c rev 14559 contains the implementation for OpenEdge.Core.System.ArgumentError. Required by #7199
>
> Can that be pushed in 6410a on xfer?

Done in 6410a/14553. Please review.

Regarding the conversion issue with the interface property defined in a skeleton class: this needs to be fixed in FWD. Please test this change in ClassDefinition.registerPropertyMethod line 820:

```
      else if (override && (javaname = lookupOverrideName(legacySig, override, true)) != null )
      {
         // first time in (for an override); register only if the parent defines this accessor
         addJavaMethodName(javaname, jparms, legacySig);

         node.putAnnotation(key, javaname);
      }
```

#27 - 05/02/2023 05:14 AM - Marian Edu

Constantin Asofiei wrote:

> Done in 6410a/14553. Please review.

Got it, thanks, will see if there is anything missing.

> Regarding the conversion issue with the interface property defined in a skeleton class: this needs to be fixed in FWD. Please test this change in ClassDefinition.registerPropertyMethod line 820:
> [...]

Yes, that works, thanks.

#28 - 05/03/2023 12:57 PM - Marian Edu

- File 6410a.patch added

- File skeleton.6410a.patch added

- Status changed from WIP to Review

Attached the patches for skeleton and FWD stubs for #6410a.

There are methods name change in JsonArray that lead to some other changes, this means application code will need to be reconverted if those are applied - otherwise maybe I revert those changes and keep the names as before, not sure if this can cause any conversion errors after skeleton updates.

There were two extra methods in JsonArray for clob and blob data types, those doesn't exist in 4GL and those data types are only valid in a data structure but are used in com.goldencode.p2j.persist.serial.JsonExport. The one for clob I've could remove since clob extends longchar and there is an add method for longchar, the one for blob I've had to kept it for now because blob doesn't extend memptr (it probably could though) and making a new memptr instance out of the blob during serialization could affect performance.

For extent properties the current conversion generates the helper (length, resize...) methods a bit different - I didn't update the names hoping the signature will solve this but not sure if those needs to be updated. I did however added the extra setter method that takes the non-extent data type argument - the setAll SETTER methods in ClientSocket.

The change in common-progress.rules is merely to help with conversion of skeleton for 'poly' return type methods, that had to be mapped to Object to avoid conversion errors.

The invoke methods in LegacyClass were changed to return Object (poly) because it can also return array (extent) but the ObjectOps.invoke keep returning BaseDataType - there is a TODO for adding support for extent though.

Please review and let me know if there is anything that needs to be changed.

#29 - 05/19/2023 10:20 AM - Constantin Asofiei

Marian, what skeleton revision did you use for the skeleton patch? I can't apply it cleanly. I've pushed skeleton rev 116 to xfer - please create the patch on this version.

#30 - 05/19/2023 11:23 AM - Constantin Asofiei

Marian, something else to note: I'm looking at the patch, and I don't understand something: the Java method name for some OO skeleton methods were renamed. What reason was for this? We need to 'freeze' this for existing skeletons implemented or stubbed in FWD. FWD takes care of using whatever Java method name is at the implementation, automatically, for a OE method call.

Also, 6410a branch is on xfer - do you still have problems using bzr on xfersrv01? If not, please use 6410a for the changes.

#31 - 05/24/2023 06:20 AM - Marian Edu

Constantin Asofiei wrote:

> Marian, what skeleton revision did you use for the skeleton patch? I can't apply it cleanly. I've pushed skeleton rev 116 to xfer - please create the patch on this version.

I think whatever was on xfer when I did the patch, I will create a new patch after updating from xfer.

#32 - 05/24/2023 06:23 AM - Marian Edu

Constantin Asofiei wrote:

> Marian, something else to note: I'm looking at the patch, and I don't understand something: the Java method name for some OO skeleton methods were renamed. What reason was for this? We need to 'freeze' this for existing skeletons implemented or stubbed in FWD. FWD takes care of using whatever Java method name is at the implementation, automatically, for a OE method call.

Understood, I can rollback those changes, wasn't sure if we must change the code to match the new output from current conversion or not... it does look like the new one is smarter though :)

> Also, 6410a branch is on xfer - do you still have problems using bzr on xfersrv01? If not, please use 6410a for the changes.

I was using 6410a from xfer, are you saying this one was updated in between?

#33 - 05/24/2023 06:24 AM - Constantin Asofiei

Marian Edu wrote:

> Also, 6410a branch is on xfer - do you still have problems using bzr on xfersrv01? If not, please use 6410a for the changes.

I was using 6410a from xfer, are you saying this one was updated in between?

I mean instead of posting patches, commit directly to 6410a on xfer.

#34 - 06/05/2023 10:30 AM - Constantin Asofiei

Constantin Asofiei wrote:

> 7199c rev 14559 contains the implementation for OpenEdge.Core.System.ArgumentError. Required by #7199

7199c was merged to trunk rev 14610 and archived

#35 - 06/22/2023 03:34 PM - Greg Shah

- Related to Bug #7460: implement Progress.Lang.Class:getMethods added

#36 - 06/23/2023 02:37 AM - Marian Edu

Constantin Asofiei wrote:

> Marian Edu wrote:
>
> > Also, 6410a branch is on xfer - do you still have problems using bzr on xfersrv01? If not, please use 6410a for the changes.
>
> I was using 6410a from xfer, are you saying this one was updated in between?
>
> I mean instead of posting patches, commit directly to 6410a on xfer.

Completely forgot about this, reverted method signature changes and pushed rev #14554 of 6410a on xfer.

#37 - 06/27/2023 07:47 AM - Greg Shah

Constantin: Have you reviewed 6410a? I'd like to get it into trunk.

Marian: We urgently need the runtime for Progress.Lang.Class:getMethods() (#7460). How quickly can that be ready?

#38 - 06/27/2023 08:19 AM - Constantin Asofiei

Marian, please upload a skeleton patch using the latest skeleton project from xfer (rev 116).

#39 - 06/27/2023 08:41 AM - Constantin Asofiei

Greg, I've taken a look at 6410a, and the main issue is changing the return type for i.e. LegacyClass.invoke, from BaseDataType to Object. I understand this is to support extent return values, but this is not that simple. The ripples go beyond just changing the return type. For example, this code:

```
def var lc as progress.lang.class.
def var l as oo.foo.
l = new oo.foo().
l:p1 = "abc".

lc = get-class("oo.foo").

function func0 returns int (input v1 as char).
   message "func0" v1.
end.

func0(lc:getPropertyValue(l, "p1")).
func0(lc:invoke(l, "m1")).
```

now (with trunk) gets converted to:

```
         func0(new character(lc.ref().getPropertyValue(l, new character("p1"))));
         func0(new character(lc.ref().invoke(l, new character("m1"))));
```

Currently, as getPropertyValue returns Object, this line will not compile in Java. If we change invoke to return also Object, func0(new character(lc.ref().invoke(l, new character("m1")))); will no longer compile, also. I can't say for sure which customers will be affected, but if we take this 6410a version as is, we can't merge to trunk until we conversion test all apps.

#40 - 06/27/2023 08:53 AM - Greg Shah

What do you suggest?

#41 - 06/27/2023 09:18 AM - Constantin Asofiei

We've discussed this POLY issue with extent in #5729. But I can't find if we made a dedicated task for making POLY work with extents.

I can clean up 6410a to merge it to trunk, but I need to see the latest skeleton patch, to have a better view of the changes (as they are linked) and to determine if apps would need a full reconversion or not.

#42 - 07/06/2023 04:14 AM - Marian Edu

Constantin Asofiei wrote:

> Marian, please upload a skeleton patch using the latest skeleton project from xfer (rev 116).

Constantin, I did update skeleton to rev #116 and the skeleton diff look exactly the same as the one I've uploaded before. The content was generated from 4GL using reflection, the methods were ordered alphabetically so that's why it might looks like there are more changes than what you've expected? Imho there should be some generator to create the skeleton based on certain OE version and then use that as-is instead of manually adding methods/properties... it's a pain in the *** to find out what was changed :(

I think the generator previously ordered the methods/properties as defined in the source code, only at some point PSC decided to use include files for quite a few of those classes and the order completely changed, a simple diff using text compare is next to useless hence the change to use the alphabetic order.

#43 - 07/06/2023 05:30 AM - Marian Edu

Did reverted back to use BDT instead of Object for `invoke` method, some fixes for `setParameter` in `ParameterList` to accommodate the 2 parameters overload - committed rev #14555 of 6410a on xfer, please review.

Let me know if the `getMethods` implementation should be done on this branch or another one.

#44 - 07/12/2023 07:41 AM - Marian Edu

- Status changed from Review to WIP

Working on implementation of reflection for supporting class's methods, having issues converting some of the test cases we have for that. One of the issues I've managed to isolate boils down to bitwise operations on flags enums, the following code doesn't convert:

```
def var oFlags as Progress.Reflect.Flags.
oFlags = Progress.Reflect.Flags:Public or Progress.Reflect.Flags:Instance.
```

This is the conversion stack trace...

```
EXPRESSION EXECUTION ERROR:
---------------------------
castType = execLib("enum_java_type", this.getChildAt(0))
           ^  { Expression error [jname = sprintf("object<? extends %s>", cdef.getQualifiedName())] }
---------------------------
Elapsed job time:  00:00:04.874
ERROR:
com.goldencode.p2j.pattern.TreeWalkException: ERROR!  Active Rule:
-----------------------
      RULE REPORT
-----------------------
Rule Type :   WALK
Source AST:  [ or ] BLOCK/CLASS_DEF/BLOCK/METHOD_DEF/BLOCK/ASSIGNMENT/ASSIGN/EXPRESSION/BITWISE_OR/ @134:48 {1
41733921864}
Copy AST  :  [ or ] BLOCK/CLASS_DEF/BLOCK/METHOD_DEF/BLOCK/ASSIGNMENT/ASSIGN/EXPRESSION/BITWISE_OR/ @134:48 {1
41733921864}
Condition :  jname = sprintf("object<? extends %s>", cdef.getQualifiedName())
Loop      :  false
--- END RULE REPORT ---

    at com.goldencode.p2j.pattern.PatternEngine.run(PatternEngine.java:1093)
    at com.goldencode.p2j.convert.TransformDriver.processTrees(TransformDriver.java:586)
    at com.goldencode.p2j.convert.ConversionDriver.back(ConversionDriver.java:585)
    at com.goldencode.p2j.convert.TransformDriver.executeJob(TransformDriver.java:998)
    at com.goldencode.p2j.convert.ConversionDriver.main(ConversionDriver.java:1284)
    at testcases.Convert.main(Convert.java:22)
Caused by: com.goldencode.expr.ExpressionException: Expression execution error @1:12 [BITWISE_OR id=1417339218
64]
    at com.goldencode.p2j.pattern.AstWalker.walk(AstWalker.java:275)
    at com.goldencode.p2j.pattern.AstWalker.walk(AstWalker.java:210)
    at com.goldencode.p2j.pattern.PatternEngine.apply(PatternEngine.java:1685)
    at com.goldencode.p2j.pattern.PatternEngine.processAst(PatternEngine.java:1571)
    at com.goldencode.p2j.pattern.PatternEngine.processAst(PatternEngine.java:1504)
    at com.goldencode.p2j.pattern.PatternEngine.run(PatternEngine.java:1056)
    ... 5 more
Caused by: com.goldencode.expr.ExpressionException: Expression execution error @1:12
    at com.goldencode.expr.Expression.execute(Expression.java:484)
    at com.goldencode.p2j.pattern.Rule.apply(Rule.java:500)
    at com.goldencode.p2j.pattern.Rule.executeActions(Rule.java:751)
    at com.goldencode.p2j.pattern.Rule.coreProcessing(Rule.java:717)
    at com.goldencode.p2j.pattern.Rule.apply(Rule.java:537)
    at com.goldencode.p2j.pattern.Rule.executeActions(Rule.java:751)
    at com.goldencode.p2j.pattern.Rule.coreProcessing(Rule.java:717)
    at com.goldencode.p2j.pattern.Rule.apply(Rule.java:537)
    at com.goldencode.p2j.pattern.RuleContainer.apply(RuleContainer.java:590)
    at com.goldencode.p2j.pattern.RuleSet.apply(RuleSet.java:98)
    at com.goldencode.p2j.pattern.RuleContainer.apply(RuleContainer.java:590)
    at com.goldencode.p2j.pattern.RuleSet.apply(RuleSet.java:98)
    at com.goldencode.p2j.pattern.AstWalker.walk(AstWalker.java:262)
    ... 10 more
Caused by: com.goldencode.expr.ExpressionException: Expression error [jname = sprintf("object<? extends %s>",
cdef.getQualifiedName())]
    at com.goldencode.expr.Expression.getCompiledInstance(Expression.java:681)
    at com.goldencode.expr.Expression.execute(Expression.java:380)
    at com.goldencode.p2j.pattern.Rule.apply(Rule.java:500)
    at com.goldencode.p2j.pattern.Rule.executeActions(Rule.java:751)
    at com.goldencode.p2j.pattern.Rule.coreProcessing(Rule.java:717)
    at com.goldencode.p2j.pattern.Rule.apply(Rule.java:537)
    at com.goldencode.p2j.pattern.Rule.executeActions(Rule.java:751)
    at com.goldencode.p2j.pattern.Rule.coreProcessing(Rule.java:717)
    at com.goldencode.p2j.pattern.Rule.apply(Rule.java:537)
    at com.goldencode.p2j.pattern.NamedFunction.execute(NamedFunction.java:454)
    at com.goldencode.p2j.pattern.AstSymbolResolver.execute(AstSymbolResolver.java:795)
    at com.goldencode.p2j.pattern.CommonAstSupport$Library.execLib(CommonAstSupport.java:1862)
    at com.goldencode.expr.CE13437.execute(Unknown Source)
    at com.goldencode.expr.Expression.execute(Expression.java:391)
    ... 22 more
Caused by: com.goldencode.expr.CompilerException: Error parsing expression
    at com.goldencode.expr.Compiler.process(Compiler.java:369)
    at com.goldencode.expr.Compiler.compile(Compiler.java:289)
    at com.goldencode.expr.Expression.getCompiledInstance(Expression.java:673)
    ... 35 more
Caused by: com.goldencode.expr.UnresolvedSymbolException: No function resolved for instance (com.goldencode.p2
j.uast.ClassDefinition getVar(int) @class com.goldencode.expr.CompiledExpression) and method (name == getQuali
fiedName, signature == ([]))
```

```
    at com.goldencode.expr.ExpressionParser.method(ExpressionParser.java:2077)
    at com.goldencode.expr.ExpressionParser.primary_expr(ExpressionParser.java:851)
    at com.goldencode.expr.ExpressionParser.un_expr(ExpressionParser.java:1625)
    at com.goldencode.expr.ExpressionParser.prod_expr(ExpressionParser.java:1477)
    at com.goldencode.expr.ExpressionParser.sum_expr(ExpressionParser.java:1405)
    at com.goldencode.expr.ExpressionParser.shift_expr(ExpressionParser.java:1326)
    at com.goldencode.expr.ExpressionParser.compare_expr(ExpressionParser.java:1238)
    at com.goldencode.expr.ExpressionParser.equality_expr(ExpressionParser.java:1165)
    at com.goldencode.expr.ExpressionParser.bit_and_expr(ExpressionParser.java:1114)
    at com.goldencode.expr.ExpressionParser.bit_xor_expr(ExpressionParser.java:1069)
    at com.goldencode.expr.ExpressionParser.bit_or_expr(ExpressionParser.java:1024)
    at com.goldencode.expr.ExpressionParser.log_and_expr(ExpressionParser.java:979)
    at com.goldencode.expr.ExpressionParser.expr(ExpressionParser.java:927)
    at com.goldencode.expr.ExpressionParser.method(ExpressionParser.java:1997)
    at com.goldencode.expr.ExpressionParser.primary_expr(ExpressionParser.java:843)
    at com.goldencode.expr.ExpressionParser.un_expr(ExpressionParser.java:1625)
    at com.goldencode.expr.ExpressionParser.prod_expr(ExpressionParser.java:1477)
    at com.goldencode.expr.ExpressionParser.sum_expr(ExpressionParser.java:1405)
    at com.goldencode.expr.ExpressionParser.shift_expr(ExpressionParser.java:1326)
    at com.goldencode.expr.ExpressionParser.compare_expr(ExpressionParser.java:1238)
    at com.goldencode.expr.ExpressionParser.equality_expr(ExpressionParser.java:1165)
    at com.goldencode.expr.ExpressionParser.bit_and_expr(ExpressionParser.java:1114)
    at com.goldencode.expr.ExpressionParser.bit_xor_expr(ExpressionParser.java:1069)
    at com.goldencode.expr.ExpressionParser.bit_or_expr(ExpressionParser.java:1024)
    at com.goldencode.expr.ExpressionParser.log_and_expr(ExpressionParser.java:979)
    at com.goldencode.expr.ExpressionParser.expr(ExpressionParser.java:927)
    at com.goldencode.expr.ExpressionParser.expression(ExpressionParser.java:676)
    at com.goldencode.expr.Compiler.parse(Compiler.java:475)
    at com.goldencode.expr.Compiler.process(Compiler.java:339)
```

#45 - 07/12/2023 07:48 AM - Constantin Asofiei

I think that was meant to be getQualifiedJavaName, not getQualifiedName.

#46 - 07/12/2023 08:55 AM - Marian Edu

Constantin Asofiei wrote:

> I think that was meant to be getQualifiedJavaName, not getQualifiedName.

That did the trick, thanks Constantin.

#47 - 07/18/2023 09:07 AM - Marian Edu

One question here, I'm looking at the 'invoke' method and that was implemented straight into the class (LegacyClass) although it might make more sense to have that logic moved in method/ctor... should I do that change or leave it where it is and callback from method into the class by passing the method's name (like I don't already have the method)?

Thanks

#48 - 07/18/2023 09:14 AM - Constantin Asofiei

Marian Edu wrote:

> One question here, I'm looking at the 'invoke' method and that was implemented straight into the class (LegacyClass) although it might make more sense to have that logic moved in method/ctor... should I do that change or leave it where it is and callback from method into the class by passing the method's name (like I don't already have the method)?

The problem here is that LegacyClass.invoke resolves the target at each invocation, depending on the arguments. With the Legacy4GLMethod, this is already resolved (FWD must call the converted Java method associated with this exact legacy method). There is ControlFlowOps.invokeLegacyMethod(InternalEntry, which I think needs to be called in case of Legacy4GLMethod.invoke - and the InternalEntry needs to be resolved using the parameter types defined at the Legacy4GLMethod.

#49 - 07/18/2023 11:33 AM - Marian Edu

Constantin Asofiei wrote:

> The problem here is that LegacyClass.invoke resolves the target at each invocation, depending on the arguments.

Not sure about this but I expect some (more like the same) method resolution to be performed when using GetMethod on the Class.

> With the Legacy4GLMethod, this is already resolved (FWD must call the converted Java method associated with this exact legacy method). There is ControlFlowOps.invokeLegacyMethod(InternalEntry, which I think needs to be called in case of Legacy4GLMethod.invoke - and the InternalEntry needs to be resolved using the parameter types defined at the Legacy4GLMethod.

Yes, that was my concern as well... if we just pass the parameters to the class's invoke it might be a different method that is being executed :(

#50 - 07/18/2023 11:40 AM - Constantin Asofiei

Marian Edu wrote:

> Constantin Asofiei wrote:
>
> > The problem here is that LegacyClass.invoke resolves the target at each invocation, depending on the arguments.
>
> Not sure about this but I expect some (more like the same) method resolution to be performed when using GetMethod on the Class.

If I understand correctly, leave the resolution logic at LegacyClass.getMethod, and you can cache the InternalEntry at Legacy4GLMethod. So we don't resolve the target on each invoke.

#51 - 07/18/2023 11:47 AM - Marian Edu

Constantin Asofiei wrote:

> Marian Edu wrote:
>
> > Constantin Asofiei wrote:
> >
> > > The problem here is that LegacyClass.invoke resolves the target at each invocation, depending on the arguments.
> >
> > Not sure about this but I expect some (more like the same) method resolution to be performed when using GetMethod on the Class.
>
> If I understand correctly, leave the resolution logic at LegacyClass.getMethod, and you can cache the InternalEntry at Legacy4GLMethod. So we don't resolve the target on each invoke.

Sounds like a plan.

#52 - 07/31/2023 11:30 AM - Constantin Asofiei

Greg, I have this patch for an issue with Progress.Lang.Class:invoke - we need to re-throw the structured exception. Should I include it in 6410a or in a different branch? As this needs to be in 7156b.

```
--- old/src/com/goldencode/p2j/oo/lang/LegacyClass.java
+++ new/src/com/goldencode/p2j/oo/lang/LegacyClass.java
@@ -967,6 +967,10 @@
              }
           }
        }
+       catch (LegacyErrorException lex)
+       {
+          throw lex;
+       }
        catch (ErrorConditionException ece)
        {
           ErrorManager.setSilent(silent);
```

#53 - 07/31/2023 12:47 PM - Greg Shah

Do you think we are ready to merge 6410a to trunk?

#54 - 07/31/2023 01:10 PM - Constantin Asofiei

Greg Shah wrote:

> Do you think we are ready to merge 6410a to trunk?

No. Method:invoke and Class:getMethod(s) are not implemented.

#55 - 07/31/2023 01:32 PM - Greg Shah

OK, then let's get this patch into trunk using a different branch.

#57 - 08/29/2023 07:19 AM - Constantin Asofiei

From Marian Edu:

> think I've stumbled upon the same problem before but can't remember exactly the use case nor the solution, or it might be just a wrong recollection... I need to have some 'internal' instances created for relection - method, parameter - to which the ctor is private and I need only to instantiate internally and pass some java objects as input so probably a simple Java ctor like for LegacyMethod is to be used, problem is how to make those internal instance to be recognised as 'valid object'. I this ObjectOps's 'newInstanceInternal' does the magic needed for this but not sure, it feels wrong to have 'getMethodInstance' and 'getParameterInstance' in ObjectOps... is there another (better) way?

#58 - 08/29/2023 07:25 AM - Constantin Asofiei

Marian, can these be created outside of FWD runtime (i.e. by the application?) If these can only be created by FWD runtime, then you can implement _BaseObject_.isTracked to return false, and they will always be valid, without any reference tracking. I assume there are no destructors to be executed anyway, correct? The point here is to carefully manage these instances as singleton (maybe cache them), as I assume the state can't be changed for these.

And if the above is true, to create these you can add a Legacy4GLMethod.newInstance(), bypass all legacy execute/constructor (or just invoke them manually). The fields most likely are no-undo, so use TypeFactory instead of UndoableFactory.

The other part we need to check is how Progress.Lang.Object:next-siblign works for a Progress.Reflect.Method instance (same for enums, I don't recall testing these). But this may be another issue which can be solved via ObjectOps.registerObject().

#59 - 08/29/2023 07:37 AM - Marian Edu

Constantin Asofiei wrote:

> Marian, can these be created outside of FWD runtime (i.e. by the application?) If these can only be created by FWD runtime, then you can implement _BaseObject_.isTracked to return false, and they will always be valid, without any reference tracking. I assume there are no destructors to be executed anyway, correct? The point here is to carefully manage these instances as singleton (maybe cache them), as I assume the state can't be changed for these.

Thanks, the 4GL ctor is private so this should do.

> The other part we need to check is how Progress.Lang.Object:next-siblign works for a Progress.Reflect.Method instance (same for enums, I don't recall testing these). But this may be another issue which can be solved via ObjectOps.registerObject().

Did a quick check and no 'reflection' instances shows in the session objects although those are 'valid-objects' - in fact not even the class instance shows in the list so I guess we do not have this problem, 4GL treats those as 'internal' as well.

#60 - 08/29/2023 07:39 AM - Constantin Asofiei

Please check also what Progress.Reflect.Method:next-sibling returns - is it always unknown? If yes, maybe this (and others in BaseObject) need to return unknown if isTracked returns false.

#61 - 08/29/2023 07:43 AM - Marian Edu

Constantin Asofiei wrote:

> Please check also what Progress.Reflect.Method:next-sibling returns - is it always unknown? If yes, maybe this (and others in BaseObject) need to return unknown if isTracked returns false.

For all 'internal' objects the valid-object on the instance itself returns true, both Next-Sibling/Prev-Sibling returns unknown for those and none of them shows in the list of objects in the current session.

#62 - 08/29/2023 07:44 AM - Constantin Asofiei

Marian Edu wrote:

> Constantin Asofiei wrote:
>
> > Please check also what Progress.Reflect.Method:next-sibling returns - is it always unknown? If yes, maybe this (and others in BaseObject) need to return unknown if isTracked returns false.
>
> For all 'internal' objects the valid-object on the instance itself returns true, both Next-Sibling/Prev-Sibling returns unknown for those and none of them shows in the list of objects in the current session.

Thanks, so we need what I mentioned before, we need to fix BaseObject.getNextSibling and similar to return unknown for untracked ~~widget~~ instance.

#63 - 08/29/2023 07:45 AM - Marian Edu

Marian Edu wrote:

> Constantin Asofiei wrote:
>
> > Please check also what Progress.Reflect.Method:next-sibling returns - is it always unknown? If yes, maybe this (and others in BaseObject) need to return unknown if isTracked returns false.
>
> For all 'internal' objects the valid-object on the instance itself returns true, both Next-Sibling/Prev-Sibling returns unknown for those and none of them shows in the list of objects in the current session.

So, I guess getNextSibling/getPrevSibling methods in BaseObject needs to be updated to return unknown if object isn't tracked.

#64 - 08/29/2023 07:45 AM - Marian Edu

Constantin Asofiei wrote:

> Thanks, so we need what I mentioned before, we need to fix BaseObject.getNextSibling and similar to return unknown for untracked widget.

Right, will do that in 6410 ok?

#65 - 08/29/2023 07:46 AM - Constantin Asofiei

Marian Edu wrote:

> Constantin Asofiei wrote:
>
> > Thanks, so we need what I mentioned before, we need to fix BaseObject.getNextSibling and similar to return unknown for untracked widget.
>
> Right, will do that in 6410 ok?

Great, thanks.

#66 - 09/05/2023 08:09 AM - Marian Edu

- % Done changed from 0 to 90

- Status changed from WIP to Review

Completed most of getMethod/getMethods implementation, please review.

There are a few things remaining:

- package-private/package-protected access modes, those are both mapped to 'package' (default) mode in java, can't infer back the 4GL mode out of it unless we add something in LegacySignature
- method invoke is now done in LegacyClass, the logic should probably be moved in Legacy4GLMethod
- some errors (validation parameters) does not pop-out of the invokeImpl of LegacyClass - does not get into the error-status when called with silent either
- ctors are not implemented, let me know if that is needed or not

Had to update some rules to keep the bind/append flags for parameters around in LegacyParameter and make public a couple of private/package methods in ControlFlowOps and SourceNameMapper so we can use them in reflection.

#67 - 09/06/2023 05:24 AM - Greg Shah

Constantin: Please review.

We need to get our OO support to the 100% mark sooner rather than later. Otherwise we will be constantly scrambling to deal with things that present as bugs but are really just missing features. So my initial idea is that we should go ahead and finish all of pending features for getMethods.

#68 - 09/21/2023 09:32 AM - Constantin Asofiei

6410a was rebased from trunk rev 14747 - new rev 14777.

I'm putting this into conversion testing of some projects.

#69 - 09/22/2023 11:12 AM - Constantin Asofiei

Branch 6410a was merged to trunk rev 14751 and archived.

Marian: please let us know when we can create branch 6410b.

#70 - 09/26/2023 05:46 AM - Marian Edu

Constantin Asofiei wrote:

> Branch 6410a was merged to trunk rev 14751 and archived.
>
> Marian: please let us know when we can create branch 6410b.

Constantin, if you want me to close the remaining issues then please go ahead and make the new branch and will pick it up from there. I will go over the unit tests and update the ones related to methods.

Thanks

#71 - 09/26/2023 09:28 AM - Greg Shah

In addition to the remaining items in getMethods(), what work is left from #6410-1?

#72 - 09/29/2023 04:04 AM - Marian Edu

Greg Shah wrote:

> In addition to the remaining items in getMethods(), what work is left from #6410-1?

As far as I can see (6410a) this is the status as of today:

- openedge.core.collections.list (stubs)
- openedge.core.collections.stringcollection (stubs)
- openedge.core.memptr (full) - resource support level needs to be updated
- openedge.core.string (full) - resource support level needs to be updated
- openedge.core.system.argumenterror (full)
- openedge.core.timestamp (stubs)

- openedge.net.http.credentials (full) - resource support level needs to be updated
- openedge.net.http.ihttpclientlibrary (full) - resource support level needs to be updated
- openedge.net.serverconnection.clientsocket (stubs)
- openedge.net.uri (full) - resource support level needs to be updated
- progress.json.objectmodel.jsonarray (full) - resource support level needs to be updated
- progress.lang.class (partial) - missing ctor's, properties, variables, events, invoke needs some attention
- progress.lang.parameterlist (full) - resource support level needs to be updated
- progress.lang.proerror (full) - the callstack is only needed if enabled for session (SESSION:ERROR-STACK-TRACE), think there was a discussion about it in ABLUnit thread
- progress.reflect.method (partial) - imho the invoke logic should be moved here from LegacyClass, some issues with dataset-handle, table-handle parameters, those are annotated as handle during conversion - we can either fix it there or need to check the java method parameter type.

Please let me know when a new branch is available for this on xfer and what needs to be done beside finishing the method/invoke support.

#73 - 09/29/2023 04:59 AM - Constantin Asofiei

Created task branch 6410b from trunk rev 14756.

The branch is available on xfer.

#74 - 09/29/2023 06:15 AM - Marian Edu

Constantin Asofiei wrote:

> Created task branch 6410b from trunk rev 14756.
>
> The branch is available on xfer.

Thanks, just to confirm I was able to pull it down.

Related to the parameter annotation for dataset-handle/table-handle is it ok to leave those annotated as simply handle or it should get fixed in conversion? Right now we only use the annotations for building the parameter 'signature', qualified also seems to be all lowercase for whatever reason :(

#75 - 09/29/2023 09:43 AM - Greg Shah

> what needs to be done beside finishing the method/invoke support.

Please do implement all the remaining items as documented in #6410-72.

#76 - 10/03/2023 08:16 AM - Marian Edu

- % Done changed from 90 to 50

Working on remaining issues on method invoke and beside the known issue with return extent I have a situation that I'm not sure how to fix it so maybe someone has an idea. The 4GL throws an error when trying to assign the result from a method invoke to a variable if the method doesn't have any return - void, inside the invoke method I have no idea the return will be assigned or not and assign doesn't really care where the value to be assigned is coming from :(

The error number is 15304, that sounds like *Invalid use of dynamic invoke of VOID method '<method_name>' in an expression*.

With the current conversion for method invoke I don't see how/where to make that validation.

```
var.assign(meth.ref().invoke(...));
```

Thoughts???

#77 - 10/04/2023 05:44 AM - Constantin Asofiei

Marian, I think we need 'invokeStandalone' to be emitted when the returned value is not read and 'invoke' in all other cases - which will force the method to have something to return, otherwise raise the 15304 error. Note that LegacyClass.invokeStandalone already exists - look in annotations/method_defs.rules at line 960:

```
         <rule>ref.isAnnotation("qualified") and
               getReferenceNoteString(ref.id, "qualified").equalsIgnoreCase("Progress.Lang.Class")
            <action>copy.putAnnotation("javaname", "invokeStandalone")</action>
         </rule>
      </rule>
   </walk-rules>
```

#78 - 10/25/2023 10:13 AM - Constantin Asofiei

Marian, please let me know when you have everything committed to 6410b, so I can get it from xfer, to rebase it.

#79 - 10/25/2023 12:34 PM - Marian Edu

Constantin Asofiei wrote:

> Marian, please let me know when you have everything committed to 6410b, so I can get it from xfer, to rebase it.

Constantin, you can go ahead and rebase it and i will merge afterwards anyway, i have a local git repo so won't lose anything…

#80 - 10/25/2023 12:47 PM - Constantin Asofiei

6410b is now on trunk rev 14790 and was pushed to xfer.

#81 - 11/06/2023 01:22 PM - Constantin Asofiei

- File dataset-handle.diff added

Marian, I have this patch to properly report datasethandle/tablehandle vs dataset/table - please review. I plan to integrate it in 7156b. Do you have work on 6410b? If so, please integrate it in that, otherwise I'll use another branch.

#82 - 11/06/2023 02:23 PM - Greg Shah

I'd like to get the next phase of #6410 support into code review if it is far enough along.

#83 - 11/07/2023 02:50 AM - Marian Edu

Constantin Asofiei wrote:

> Marian, I have this patch to properly report datasethandle/tablehandle vs dataset/table - please review. I plan to integrate it in 7156b. Do you have work on 6410b? If so, please integrate it in that, otherwise I'll use another branch.

Hi Constantin, thought you modified the conversion rules or something to be more specific about the handle in the signature... I've taken almost the exact path as you meanwhile, I will push some of current changes in bazaar today.

I'm trying to fight a case when invalid parameters are being sent to invoke, the error message contains the caller method/procedure and I think because of the block added to invoke this is 'polluting' the call-stack, in FWD the caller is reported as invoke while in 4GL it is the original caller where invoke was used... I can drop the function block there but then you said there might be issues with error handling, I've couldn't come up with any solution so far :(

#84 - 11/07/2023 04:36 AM - Constantin Asofiei

- File call_mode_patch.diff added

Marian, see another patch - please integrate it also in 6410b.

> I'm trying to fight a case when invalid parameters are being sent to invoke, the error message contains the caller method/procedure and I think because of the block added to invoke this is 'polluting' the call-stack, in FWD the caller is reported as invoke while in 4GL it is the original caller where invoke was used... I can drop the function block there but then you said there might be issues with error handling, I've couldn't come up with any solution so far :(

Hmm... this may mean that OpenEdge does not report the internal classes in the callstack. Can you get some other cases where an internal OpenEdge class call fails, and check the callstack?

#85 - 11/07/2023 05:10 AM - Marian Edu

Constantin Asofiei wrote:

> Hmm... this may mean that OpenEdge does not report the internal classes in the callstack. Can you get some other cases where an internal OpenEdge class call fails, and check the callstack?

Nope, seems to only be something specific to the invoke method - although might well be other 'internal' methods of classes from the Progress namespace.

#86 - 11/07/2023 06:33 AM - Constantin Asofiei

Marian Edu wrote:

> ... because of the block added to invoke this is 'polluting' the call-stack

Was this block added in 6410b? I don't see any BlockManager API in trunk.

#87 - 11/07/2023 07:55 AM - Marian Edu

Constantin Asofiei wrote:

> Marian Edu wrote:
>
> > ... because of the block added to invoke this is 'polluting' the call-stack
>
> Was this block added in 6410b? I don't see any BlockManager API in trunk.

I did added that as per your code review, the invoke method in LegacyClass had/has no block - for LegacyMethod I've tried to add one, it does kinda work but has that nasty side-effect. I understand why is behaving like it does but as said 4GL seems to not add this call to invoke into the call-stack, regular class methods call works fine.

#88 - 11/08/2023 07:08 AM - Marian Edu

Constantin Asofiei wrote:

> Marian, see another patch - please integrate it also in 6410b.

Had to manually apply that, for Call there seems to be a few revisions missing - you've sent `014`, what was in 6410b was up to `011`.

#89 - 11/08/2023 07:14 AM - Constantin Asofiei

I see these commits on xfer/6410b:

```
------------------------------------------------------------
revno: 14792
committer: Marian Edu<marian.edu@acorn.ro>
branch nick: 6410b
timestamp: Wed 2023-11-08 14:09:44 +0200
message:
  add no-error option for create dataset/datasource
------------------------------------------------------------
revno: 14791
committer: Marian Edu<marian.edu@acorn.ro>
branch nick: 6410b
timestamp: Wed 2023-11-08 14:05:34 +0200
message:
  import call params patch
```

If you have nothing else to commit, I'll rebase.

#90 - 11/08/2023 07:18 AM - Marian Edu

Constantin Asofiei wrote:

> If you have nothing else to commit, I'll rebase.

Give me 10-15 mins please to commit some more...

#91 - 11/08/2023 07:21 AM - Constantin Asofiei

Marian Edu wrote:

> Constantin Asofiei wrote:
>
> > If you have nothing else to commit, I'll rebase.
>
> Give me 10-15 mins please to commit some more...

OK, let me know when it's done.

#92 - 11/08/2023 08:55 AM - Marian Edu

Constantin Asofiei wrote:

> OK, let me know when it's done.

All done what I've could get 'proper' enough... rev #14794. Let me know when done with the rebase and will pull it down afterwards. Thanks

#93 - 11/08/2023 09:52 AM - Constantin Asofiei

Starting rebase now.

#94 - 11/08/2023 09:55 AM - Constantin Asofiei

6410b rev 14794 was rebased from trunk rev 14824 - new rev 14828.

#95 - 11/10/2023 03:23 AM - Constantin Asofiei

Marian, please remind me of your issues - I plan to look at them over the next week. Also, point me to any test representative for these issues. And make sure all relevant changes are committed to 6410b before you leave today.

#96 - 12/04/2023 03:24 AM - Constantin Asofiei

6410b rev 14828 was rebased from trunk rev 14856 - new rev 14860

#97 - 01/16/2024 04:01 PM - Greg Shah

What is the status of this work? Are we close to being able to merge it?

#98 - 01/31/2024 04:07 PM - Constantin Asofiei

Marian: we will get #6410-81 and #6410-84 patches to trunk via #8238. We need to rebase 6410b, please let me know when is a good time (all is committed to xfer).

#99 - 01/31/2024 04:13 PM - Marian Edu

Constantin Asofiei wrote:

> Marian: we will get #6410-81 and #6410-84 patches to trunk via #8238. We need to rebase 6410b, please let me know when is a good time (all is committed to xfer).

Constantin, please go ahead and rebase that and will pick it up and merge on our side as needed. Thanks

#100 - 03/26/2024 04:09 AM - Marian Edu

Constantin, are there any changes in trunk pertaining conversion especially for classes extending/implementing skeleton classes? We kept getting conversion issues that we had to manually fix (after ignoring the conversion error). I've added some sample source code in `tests/convert` folder in testcases (rev #1566) that should show the issue we had with it, I do have a local patch for it in case it wasn't already reported/fixed. Basically the changes are in @SymbolResolver for resolving getter/setter for extent properties in builtin classes ('bulk') and in ClassDefinition to avoid NPE (null setter java method if interface/base class only have a getter defined), this is also only for when implementing/extending built-in classes. Please let me know if this is fixed already and you can merge it in 6410 or we keep the changes and push them in 6410.

Thanks.

#101 - 03/27/2024 05:06 PM - Greg Shah

Some notes from Marian:

> We had constantly issues with conversion when classes extended/implemented built-in classes (skeleton), I've made a note for this on the #6410-100 and I will try to see if this was already solved in trunk otherwise we keep the current local patch - otherwise on each conversion we had to manually fix the converted code and that was anything but fun :(
>
> I expect we will close #6410 this week, we are working a bit more on collection/list/iterators tests trying to figure out how the implementation actually looks like - the issue here is that the 4GL implementation is based on temp-table (AbstractTTCollection) and both Collection and List have constructors that can take a temp-table handle as parameter. Normally in FWD implementation we do not use temp-table, there is some support for this on AbstractTTCollection and Collection but we really didn't test that part. It's probably useless to implement that as I don't expect those to be extended in the first place but the possibility is there so god knows what developers can do... my preference would be to simply remove those constructors so we easily find out if used in client's code base and maybe only implement that part when/if needed, Greg?

#102 - 03/27/2024 05:07 PM - Greg Shah

> It's probably useless to implement that as I don't expect those to be extended in the first place but the possibility is there so god knows what developers can do... my preference would be to simply remove those constructors so we easily find out if used in client's code base and maybe only implement that part when/if needed

Agreed. Please add some comments in the class javadoc about this "intentional limitation" where we are avoiding any features that directly map to the temp-table approach used in the 4GL.

#103 - 04/26/2024 06:49 AM - Marian Edu

- Status changed from Review to WIP

- % Done changed from 50 to 90

I have a question on how is the proper way to return a value out of a OO method (really a function block), we've used to use `returnNormal` (or extent) and that conveniently exit the block so normally nothing after `returnNormal` was executed. This seems to be still the case for 'built-in' classes (from `p2j.oo` package) but it fails to do so in case we have a class that extends one of those `built-in` classes in which case `returnNormal` doesn't exit the block and execution continue afterwards.

Trying to figure out what is wrong there come across this comment in `BlockManager`:

```
// WARNING: the p2o.oo code is not refactored to use 'storeReturnValue' or Java return statements
// instead, this code uses and expects for the returnNormal call to exit the block.
```

and it looks like the check is done on the class name, that explains why it's working for built-in classes and it fails if we use a class that extends the built-in one :)

Question is, what should we do here? Should we always use `return` after `returnNormal` or we can change the code in `BlockManager` to take into account inheritance and use `LegacyResource` annotation instead of the package name matching as mentioned in another comment there?

#104 - 04/26/2024 07:04 AM - Marian Edu

The code in `p2j.oo` almost exclusively use `returnNormal` and expect the execution to end there instead of the `storeReturnValue` paired with a Java `return` as generated by conversion lately. Is `returnNormal` even used outside of the built-in classes? Wouldn't just change the `returnWorker` to throw the `ReturnUnwindException` always unless `STORE` option used?

#105 - 04/26/2024 07:12 AM - Constantin Asofiei

Marian Edu wrote:

> The code in `p2j.oo` almost exclusively use `returnNormal` and expect the execution to end there instead of the `storeReturnValue` paired with a Java `return` as generated by conversion lately. Is `returnNormal` even used outside of the built-in classes? Wouldn't just change the `returnWorker` to throw the `ReturnUnwindException` always unless `STORE` option used?

FWD in BlockManager now has 'safety' code to allow returnNormal from p2j.oo code to work as before the storeReturnValue changes. But, if you write new Java code in p2j.oo, use storeReturnValue(); return; but **only if this actually exits the function block**. I mean, you can't use this pattern from a REPEAT block.

#106 - 04/29/2024 02:50 AM - Marian Edu

- File 6410.patch added

Constantin Asofiei wrote:

> Marian Edu wrote:
>
> > The code in `p2j.oo` almost exclusively use `returnNormal` and expect the execution to end there instead of the `storeReturnValue` paired with a Java `return` as generated by conversion lately. Is `returnNormal` even used outside of the built-in classes? Wouldn't just change the `returnWorker` to throw the `ReturnUnwindException` always unless `STORE` option used?
>
> FWD in BlockManager now has 'safety' code to allow returnNormal from p2j.oo code to work as before the storeReturnValue changes. But, if you write new Java code in p2j.oo, use storeReturnValue(); return; but **only if this actually exits the function block**. I mean, you can't use this pattern from a REPEAT block.

OK, we will just leave `returnNormal` alone then where it was used, I don't know the reasons behind 'storeReturnValue' but i would rather use one statement to return from a block :)

In the interim we use thi local patch for `BlockManager' the uses the 'executing' information from 'CalleeInfo' (had to make that public in the interface).

#107 - 04/30/2024 03:17 AM - Marian Edu

Constantin, I'm trying to rerun all tests for method invoke and parameter list but those fails to convert now because of the method lookup, there seems to be no real support for 'poly' parameters in there... both 'exact' and 'flex mode' match are using ParameterKey's 'typeEquivalent' and the 'Object' simply does not equal any of the other data types we're using there :)

I see there is a new property 'dynamicPoly' that is not really used, thought to set that to true if type is 'bdt' ('Object') and mode is INPUT but it just feels wrong to let the ParameterKey if it's type equivalent, imho that depends on how it is used - it is from the caller or callee, it has no knowledge of that.

Unless that was already fixed in another branch that we can merge back in 6410 I'm thinking to change the method lookup in ClassDefinition, I could use the 'dynamicPoly' property but I'm not sure I got its meaning right... should that be true for input BDT/Object parameter or what is it used for exactly?

#108 - 05/02/2024 02:53 AM - Marian Edu

We have tests for `ParameterList` that are failing for extent parameters used as output/input-output. As it is now there is no way to make this work since the conversion is passing the array as input to the `setParameter` method so I don't think we can turn that into an `OutputExtentParameter`/`InputOutputExtentParameter`. Imho this can only be solved during conversion like a special case for `setParameter` method when the mode is output/input-output and value is an extent, thoughts?

#109 - 05/03/2024 07:28 AM - Constantin Asofiei

Marian Edu wrote:

> We have tests for `ParameterList` that are failing for extent parameters used as output/input-output. As it is now there is no way to make this work since the conversion is passing the array as input to the `setParameter` method so I don't think we can turn that into an `OutputExtentParameter`/`InputOutputExtentParameter`. Imho this can only be solved during conversion like a special case for `setParameter` method when the mode is output/input-output and value is an extent, thoughts?

This gets tricky if you pass an extent field via ::. But yes, this would need to get solved via conversion, but you don't know which would be used, OutputExtentParameter or InputOutputExtentParameter (as the mode can be a variable, not literal), so we need something different.

Files

6410a.patch	48.4 KB	05/03/2023	Marian Edu
skeleton.6410a.patch	82.3 KB	05/03/2023	Marian Edu
dataset-handle.diff	3.48 KB	11/06/2023	Constantin Asofiei
call_mode_patch.diff	4.31 KB	11/07/2023	Constantin Asofiei
6410.patch	2.6 KB	04/29/2024	Marian Edu