Base Language - Bug #4754

DYNAMIC-FUNCTION parsing fails when first parameter is not an expression

07/12/2020 05:24 PM - Roger Borrello

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

Description

History

#1 - 07/12/2020 05:31 PM - Roger Borrello

The second DYNAMIC-FUNCTION call fails to parse:

```
DEFINE VARIABLE phTarget AS HANDLE     NO-UNDO.
DEFINE VARIABLE hFunc    AS HANDLE     NO-UNDO.

hFunc:BUFFER-VALUE = DYNAMIC-FUNCTION('hFunc-NAME' IN phTarget) NO-ERROR.
hFunc:BUFFER-VALUE = DYNAMIC-FUNCTION(hFunc:NAME IN phTarget) NO-ERROR.
```

```
./uast/handle/handle_attr_dyn_func.p:5:53: unexpected token: phTarget
    at com.goldencode.p2j.uast.ProgressParser.widget_qualifier(ProgressParser.java:58653)
    at com.goldencode.p2j.uast.ProgressParser.chained_object_members(ProgressParser.java:22234)
    at com.goldencode.p2j.uast.ProgressParser.un_type(ProgressParser.java:57875)
    at com.goldencode.p2j.uast.ProgressParser.prod_expr(ProgressParser.java:57742)
    at com.goldencode.p2j.uast.ProgressParser.sum_expr(ProgressParser.java:41663)
    at com.goldencode.p2j.uast.ProgressParser.compare_expr(ProgressParser.java:57328)
    at com.goldencode.p2j.uast.ProgressParser.log_not_expr(ProgressParser.java:57180)
    at com.goldencode.p2j.uast.ProgressParser.bitwise_xor_expr(ProgressParser.java:57111)
    at com.goldencode.p2j.uast.ProgressParser.log_and_expr(ProgressParser.java:57050)
    at com.goldencode.p2j.uast.ProgressParser.expr(ProgressParser.java:12178)
    at com.goldencode.p2j.uast.ProgressParser.func_call_parm(ProgressParser.java:61588)
    at com.goldencode.p2j.uast.ProgressParser.dynamic_function_func(ProgressParser.java:59597)
    at com.goldencode.p2j.uast.ProgressParser.primary_expr(ProgressParser.java:58168)
    at com.goldencode.p2j.uast.ProgressParser.chained_object_members(ProgressParser.java:22170)
    at com.goldencode.p2j.uast.ProgressParser.un_type(ProgressParser.java:57875)
    at com.goldencode.p2j.uast.ProgressParser.prod_expr(ProgressParser.java:57742)
    at com.goldencode.p2j.uast.ProgressParser.sum_expr(ProgressParser.java:41663)
    at com.goldencode.p2j.uast.ProgressParser.compare_expr(ProgressParser.java:57328)
    at com.goldencode.p2j.uast.ProgressParser.log_not_expr(ProgressParser.java:57180)
    at com.goldencode.p2j.uast.ProgressParser.bitwise_xor_expr(ProgressParser.java:57111)
    at com.goldencode.p2j.uast.ProgressParser.log_and_expr(ProgressParser.java:57050)
    at com.goldencode.p2j.uast.ProgressParser.expr(ProgressParser.java:12178)
    at com.goldencode.p2j.uast.ProgressParser.un_type(ProgressParser.java:57898)
    at com.goldencode.p2j.uast.ProgressParser.prod_expr(ProgressParser.java:57742)
    at com.goldencode.p2j.uast.ProgressParser.sum_expr(ProgressParser.java:41663)
    at com.goldencode.p2j.uast.ProgressParser.compare_expr(ProgressParser.java:57328)
    at com.goldencode.p2j.uast.ProgressParser.log_not_expr(ProgressParser.java:57180)
    at com.goldencode.p2j.uast.ProgressParser.bitwise_xor_expr(ProgressParser.java:57111)
    at com.goldencode.p2j.uast.ProgressParser.log_and_expr(ProgressParser.java:57050)
    at com.goldencode.p2j.uast.ProgressParser.expr(ProgressParser.java:12178)
    at com.goldencode.p2j.uast.ProgressParser.assignment(ProgressParser.java:8691)
    at com.goldencode.p2j.uast.ProgressParser.single_block(ProgressParser.java:7334)
    at com.goldencode.p2j.uast.ProgressParser.block(ProgressParser.java:7017)
    at com.goldencode.p2j.uast.ProgressParser.external_proc(ProgressParser.java:6944)
    at com.goldencode.p2j.uast.AstGenerator.parse(AstGenerator.java:1571)
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:996)
```

```
    at com.goldencode.p2j.uast.ScanDriver.lambda$scan$0(ScanDriver.java:375)
    at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:410)
    at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:248)
    at com.goldencode.p2j.convert.TransformDriver.runScanDriver(TransformDriver.java:369)
    at com.goldencode.p2j.convert.TransformDriver.front(TransformDriver.java:234)
    at com.goldencode.p2j.convert.TransformDriver.executeJob(TransformDriver.java:944)
    at com.goldencode.p2j.convert.ConversionDriver.main(ConversionDriver.java:1025)
Failure in file '/home/rfb/projects/VirtualBox-VMs/shared/projects/testcases/uast/handle/handle_attr_dyn_func.
p':
com.goldencode.ast.AstException: Error processing ./uast/handle/handle_attr_dyn_func.p
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:1008)
    at com.goldencode.p2j.uast.ScanDriver.lambda$scan$0(ScanDriver.java:375)
    at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:410)
    at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:248)
    at com.goldencode.p2j.convert.TransformDriver.runScanDriver(TransformDriver.java:369)
    at com.goldencode.p2j.convert.TransformDriver.front(TransformDriver.java:234)
    at com.goldencode.p2j.convert.TransformDriver.executeJob(TransformDriver.java:944)
    at com.goldencode.p2j.convert.ConversionDriver.main(ConversionDriver.java:1025)
Caused by: java.lang.RuntimeException: Parser encountered 1 errors
    at com.goldencode.p2j.uast.AstGenerator.parse(AstGenerator.java:1640)
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:996)
    ... 7 more
```

The fact that the second instance has a handle attribute, instead of an expressions seems to mess up func_call_param:

```
func_call_parm [boolean builtin]
   {
      boolean isInputFunc = (LA(1) == KW_INPUT &&
                             LA(2) == KW_FRAME &&
                             resolveLvalueCoreType(4, false) != -1);
      boolean classLookup = sym.isClassLookup();
      sym.setClassLookup(false);
   }
   :
      // this should be merged with the parameter rule (this one doesn't create a root
      // node and here we match KW_INPUT as a builtin based on the flag passed in)
      (
         options { generateAmbigWarnings = false; }
         :
           { !builtin && !isInputFunc }?
           KW_INPUT
         | KW_OUTPUT
         | KW_IN_OUT
      )?
      expr
      (KW_APPEND | KW_BY_VALUE | KW_BY_REF | KW_BIND)*
      {
         sym.setClassLookup(classLookup);
      }
   ;
```

In debugging, the call to func_call_param for the first DYNAMIC-FUNCTION determines it is an expression. But on the second, does not.

#3 - 07/12/2020 05:37 PM - Roger Borrello

Added testcase uast/handle/handle_attr_dyn_func.p

#4 - 07/12/2020 06:06 PM - Greg Shah

Roger Borrello wrote:

> It is like the IN is never parsed.

No, the IN is seen but the following token is unrecognized.

```
widget_qualifier
   :
      KW_IN^
      (
         options { generateAmbigWarnings = false; }
         :
           frame_reference[false]
         | menu_reference

           // used to be browse_reference but that rule was removed and this
           // was consolidated as a recursive call to lvalue
         | { LA(1) == KW_BROWSE }? lvalue
      )
   ;
```

There is no provision to match a "bare" expression of type handle. It only will match FRAME framename or MENU menuname or SUBMENU submenuname or BROWSE browsename. The lvalue is only used is KW_BROWSE is the next token after KW_IN. We could try removing the contraint (officially this is a "semantic predicate") of { LA(1) == KW_BROWSE }?. This would allow any kind of handle expression (and much more). But this may be wrong if any kind of handle sub-expression can be written here. In that case you could leave the semantic predicate and instead add an alternative for the handle rule after it.

```
handle
   :
      h:chained_object_members
      {
         #h.getType() == VAR_HANDLE         ||
         #h.getType() == FIELD_HANDLE       ||
         #h.getType() == FUNC_HANDLE        ||
         #h.getType() == COLON              ||
         #h.getType() == OBJECT_INVOCATION  ||
         #h.getType() == OO_METH_HANDLE     ||
         #h.getType() == SYS_HANDLE         ||
         #h.getType() == VAR_COM_HANDLE     ||
         #h.getType() == FIELD_COM_HANDLE   ||
         #h.getType() == FUNC_COM_HANDLE    ||
         #h.getType() == OO_METH_COM_HANDLE
      }?
   ;
```

#5 - 07/12/2020 06:07 PM - Greg Shah

From Roger:

Why would widget_qualifier be parsing, instead of dynamic_function_func

```
dynamic_function_func
   :
      d:KW_DYN_FUNC^
      {
         #d.setType(FUNC_POLY);
      }
      lparens
      (
         func_call_parm[false] (in_procedure_clause)?
         (comma func_call_parm[false])*
      )
      rparens
      {
         // save the original token type that rewriting erased
         ##.putAnnotation("oldtype", new Long(KW_DYN_FUNC));

         // write our function-specific annotations
         sym.annotateFunction(#d.getText(), ##, false);
      }
   ;
```

and in_procedure_clause?

```
/**
 * Matches the IN keyword and the required following expression
 * that resolves to the procedure handle construct in which the internal
 * procedure is to be run.
 * <p>
 * Used by {@link #run_stmt}, {@link #dynamic_function_func}, {@link #event_proc_clause},
 * {@link #subscribe_stmt} and {@link #unsubscribe_stmt}, the subtree is rooted by the keyword
 * (which is the reason for using a separate rule for something so simple).
 */
in_procedure_clause
   :
      KW_IN^ expr
   ;
```

#6 - 07/12/2020 06:10 PM - Greg Shah

Because the 2nd case is not supposed to match an in_procedure_clause. The 2nd usage is an attribute that is being qualified by a widget handle. Did you try my original suggestion for changing widget_qualifier?

#7 - 07/13/2020 12:07 AM - Roger Borrello

Greg Shah wrote:

> Because the 2nd case is not supposed to match an in_procedure_clause. The 2nd usage is an attribute that is being qualified by a widget handle. Did you try my original suggestion for changing widget_qualifier?

I tried both ways:

```
widget_qualifier
   :
      KW_IN^
      (
         options { generateAmbigWarnings = false; }
         :
           frame_reference[false]
         | menu_reference

           // used to be browse_reference but that rule was removed and this
           // was consolidated as a recursive call to lvalue
         | lvalue
      )
   ;
```

and

```
widget_qualifier
   :
      KW_IN^
      (
         options { generateAmbigWarnings = false; }
         :
           frame_reference[false]
         | menu_reference

           // used to be browse_reference but that rule was removed and this
           // was consolidated as a recursive call to lvalue
         | { LA(1) == KW_BROWSE }? lvalue
         | handle
      )
   ;
```

Both worked and generated equivalent ast files.

#8 - 07/13/2020 03:16 AM - Constantin Asofiei

What's the generated Java code for this call? The dynamic_function_func rule checks for func_call_parm[false] (in_procedure_clause)?, which has:

```
in_procedure_clause
   :
      KW_IN^ expr
   ;
```

I don't think func_call_parm should resolve the IN clause via the widget_qualifier. widget_qualifier needs to check expressions like ... IN <frame-reference> (or other kind of reference, not handle). There is no syntax AFAIK which can have a widget qualifier with a handle on the right-side of IN.

#9 - 07/13/2020 08:33 AM - Roger Borrello

The 4GL in the ADM2 panel.p is:

```
  DO iFunc = 1 TO hBuffer:NUM-FIELDS.
    hFunc = hBuffer:BUFFER-FIELD(iFunc).
    ghTargetProcedure = TARGET-PROCEDURE.
    hFunc:BUFFER-VALUE = DYNAMIC-FUNCTION(hFunc:NAME IN phTarget) NO-ERROR.
  END.
```

and the converted code is:

```
         doTo("loopLabel10", toClause2, new Block((Body) () ->
         {
            hFunc.assign(hBuffer.unwrapBuffer().bufferField(iFunc));
            ghTargetProcedure.assign(targetProcedure());
            silent(() -> hFunc.unwrapBufferField().changeValue(ControlFlowOps.invokeDynamicFunctionIn(hFunc.un
wrap().name(), phTarget_1)));
         }));
```

I was a little confused that Greg had pointed out the change was in widget_qualifier, but that certainly worked. The large ChUI regression test passed with the 2nd version of his recommended changes, and the first is running now.

#10 - 07/13/2020 08:46 AM - Constantin Asofiei

hFunc:NAME is a character, which in the DYNAMIC-FUNCTION's syntax for the first argument (function-name[IN proc-handle]) is the target function name. IN phTarget is the persistent program handle.

My opinion is that the parser should no go down the widget_qualifier path - if we change widget_qualifier to accept IN handle, then we are adding to the 4GL syntax...

#11 - 07/13/2020 09:12 AM - Greg Shah

If you are sure that this is procedure handle syntax instead of widget qualifier syntax, then we may need a different approach. However, the solution is messy.

The widget:attribute IN frame_or_menu_or_browse is valid 4GL syntax. The parser predicts this match in preference to the in_procedure_clause because func_call_parm uses expr and expr will "see" the KW_IN before it exits back into dynamic_function_func where it could match to in_procedure_clause. More specifically, in chained_object_members, we match the COLON followed by attribute_or_method and this is where we optionally match widget_qualifier.

To avoid this, we must know that we are processing the first parameter of the dynamic_function_func somewhere further up the stack.

1. Add a member variable inDynFunc to the parser class.

2. Modify dynamic_function_func like this:

```
      d:KW_DYN_FUNC^
      {
         #d.setType(FUNC_POLY);
         inDynFunc = true;
      }
      lparens
      (
         func_call_parm[false] { inDynFunc = false; } (in_procedure_clause)?
         (comma func_call_parm[false])*
      )
      rparens
```

3. Modify chained_object_members like this:

```
               | am:attribute_or_method  { refnode = #am; }
                 (
                     // this is needed for constructs such as var:ATTR IN FRAME frame
                     { !inDynFunc }?
                     widget_qualifier
                   | // empty alternative
                 )
```

#12 - 07/13/2020 10:12 AM - Roger Borrello

Why is the 1st parameter of KW_DYN_FUNC handled the same as the other parameters, when it is the *function-name*, not a parameter? Wouldn't that allow for parsing separate from func_call_parm? Something more like:

```
dynamic_function_func
   :
      d:KW_DYN_FUNC^
      {
         #d.setType(FUNC_POLY);
      }
      lparens
      (
         func_call_name (in_procedure_clause)?   <-- new method
         (comma func_call_parm[false])*
      )
      rparens
      {
         // save the original token type that rewriting erased
         ##.putAnnotation("oldtype", new Long(KW_DYN_FUNC));

         // write our function-specific annotations
         sym.annotateFunction(#d.getText(), ##, false);
      }
   ;
```

The func_call_name would just parse the handle?

#13 - 07/13/2020 10:21 AM - Greg Shah

The first parameter can be an arbitrarily complex sub-expression like something + func(handle:blah:blah:method(), object-stuff:meth()) + if not trash then yatta else another:stuff:whatever():more-stuff. That matches with expr.

#14 - 07/14/2020 11:57 AM - Roger Borrello

Greg Shah wrote:

> The first parameter can be an arbitrarily complex sub-expression like something + func(handle:blah:blah:method(), object-stuff:meth()) + if not trash then yatta else another:stuff:whatever():more-stuff. That matches with expr.

But why does it need to be handled using func_call_parm? It's not the same as a function call parameter.

#15 - 07/14/2020 02:19 PM - Greg Shah

> It's not the same as a function call parameter.

In what way?

We know that the parameter must be an expr. That is the same as a function call parameter. The only restriction is that the result must be a character expression, but that does not exclude anything of consequence. I don't see what you are trying to achieve.

Did you try my suggestion?

#16 - 07/16/2020 09:25 AM - Roger Borrello

Greg Shah wrote:

> It's not the same as a function call parameter.

> In what way?

Well, my ignorance was showing. It certainly is a function call parameter, but is a very particular one.

> We know that the parameter must be an expr. That is the same as a function call parameter. The only restriction is that the result must be a character expression, but that does not exclude anything of consequence. I don't see what you are trying to achieve.
>
> Did you try my suggestion?

Yes, and it works very well! Can I check it in?

#17 - 07/16/2020 09:42 AM - Greg Shah

> Can I check it in?

Yes.

#18 - 07/16/2020 09:56 AM - Roger Borrello

3821c-11399 ready for review.

#19 - 07/16/2020 10:10 AM - Greg Shah

- Assignee set to Roger Borrello

- Status changed from New to Test

- % Done changed from 0 to 100

I'm fine with the change.

#20 - 04/20/2021 04:09 PM - Roger Borrello

Was this waiting for the large ChUI regression to complete in order to be closed?

#21 - 04/20/2021 04:32 PM - Greg Shah

No, DYNAMIC-FUNCTION() is not used in that code. If you confirm all is well, I will close it.

#22 - 04/20/2021 05:53 PM - Roger Borrello

All is well... at least as far as this task is concerned.

#23 - 04/20/2021 07:20 PM - Greg Shah

- Status changed from Test to Closed