Base Language - Feature #2311

rewrite parsing of handle chains and other misc parser changes

05/28/2014 12:09 PM - Constantin Asofiei

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Constantin Asofiei	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	GUI Support for a Complex ADM2 App		
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to Base Language - Feature #1606: implement persistent procedures/sup...	**Closed**	**11/09/2012**

History

#1 - 05/28/2014 12:24 PM - Constantin Asofiei

- File ca_upd20140526f.zip added

- Subject changed from rewrite parsing of handle changes and other misc parser changes to rewrite parsing of handle chains and other misc parser changes

- File ges_upd20140428a.zip added

The ges_upd20140428a.zip (amongs other changes) rewrites the AST tree for a chaining in a handle, com-handle or object invocation expression, to be in polish form.

ca_upd20140526f.zip is built on top of ges_20140428a.zip and:

- rewrites the conversion rules to be compatible with the parser. this includes:
  - convert/methods_attributes.rules was changed to properly walk and emit the methods/attributes
  - fixes the assign-style statements (as the parser now produces a EXPRESSION node above the lvalue)
  - fixes PUT BITS APIs when is used in an assign-style statement
  - fixed BUFFER Book:HANDLE:BUFFER-COPY and loner literals, which need to be hidden.
  - Fixed a case to ignore unneeded LPARENS node, like in ("bar" + ch) (regression by parser changes).
- changes the handle.unwrap calls from static method calls to instance method calls.

There are no unexpected changes in the MAJIC converted code. The server code had some changes in emitting the comments (which do not affect logic), beside the expected unwrap changes.

Still need to commit the testcases to bzr.

#2 - 05/29/2014 05:38 AM - Constantin Asofiei

A remainder issue is an investigation of the annotations/assignment_style_stmt_rewriting.rules file: this will split a KW_ASSIGN node into multiple nodes, if there are multiple assignment style statements in the same batch ASSIGN statement. The testcase which shows this is uast/assign_style_stmts.p (this test covers most of the assign-style statements).

Following is an email thread regarding this issue (read bottom-up):

```
Constantin,

> I'm curious, do you recall why was this needed? Because I don't quite
> understand why 4GL would want to split this batch assignment into
> multiple pieces...

I believe this was the simplest approach to get the job done at the
time.  I'm pretty sure that there was no 4GL behavior being duplicated
here.  Instead, the problem is that I wanted to convert these embedded
language statements using the same code as would be used if they were a
standalone assignment style language statement. So, the easiest approach
was to split these off.

Greg

On 05/24/2014 07:37 AM, Constantin Asofiei wrote:
> Greg,
>
> In assignment_style_stmt_rewriting.rules H004 there is this note:
>
> A new approach was implemented that splits the assignment statement
> into pieces, keeping the order the same as the original code.
>
> For an ASSIGN statement with multiple "assign-style-statements" as
> children, this will result in generating one ore more additional
> ASSIGN, and the result code will not have a single
> RecordBuffer.start/endBatch pair, but multiple pairs.
>
> I'm curious, do you recall why was this needed? Because I don't quite
> understand why 4GL would want to split this batch assignment into
> multiple pieces...
>
> Thanks,
> Constantin
```

#3 - 05/29/2014 05:26 PM - Greg Shah

Code Review 0526f

Generally, I am happy with the changes. The methods_attributes.rules is so much simpler now. Nice.

1. The com_invocation/com_method processing in methods_attributes.rules is removed. Is that by design? If so, how do we handle those cases now? That support is just provisional for now, but it is important to allow conversion of those cases to go forward without causing abends.

2. The progress.g history entry needs to be moved up into the header. I previously had it separated to avoid merge issues.

3. I don't see any changes to the ECW. That code definitely does quite a bit of processing based on handle chaining and it is important to make it work properly. I would expect issues in the server project without changes there.

#4 - 05/30/2014 01:46 PM - Constantin Asofiei

Greg Shah wrote:

> Code Review 0526f
>
> Generally, I am happy with the changes. The methods_attributes.rules is so much simpler now. Nice.
>
> 1. The com_invocation/com_method processing in methods_attributes.rules is removed. Is that by design? If so, how do we handle those cases now? That support is just provisional for now, but it is important to allow conversion of those cases to go forward without causing abends.

OK, I'll add it back, and do some little testing to ensure it is working.

> 3. I don't see any changes to the ECW. That code definitely does quite a bit of processing based on handle chaining and it is important to make it work properly. I would expect issues in the server project without changes there.

The server project and the #2050 tests converted OK after the methods_attributes changes, so I think ECW is compatible with the current AST. Maybe is because now the child on index 1 of a root COLON node is always the rightmost one, which actually gives the chain's type? This is somehow consistent with the previous AST structure, where the right-most node of a COLON was the one determining the resulting type; now, the right subtree of the root COLON node has only one node.

What I think ECW needs is just some simplifications of the cases when COLON nodes are involved. BTW, are there other POLY tests, beside the ones at #2050?

#5 - 05/30/2014 02:17 PM - Greg Shah

> BTW, are there other POLY tests, beside the ones at #2050?

Yes, dynamic_function_context_analysis.p is useful too.

The buffer_value_* files and the dynamic_function_context_analysis.p are checked into testcases/uast/.

#6 - 05/30/2014 03:01 PM - Constantin Asofiei

Constantin Asofiei wrote:

> > 1. The com_invocation/com_method processing in methods_attributes.rules is removed. Is that by design? If so, how do we handle those cases now? That support is just provisional for now, but it is important to allow conversion of those cases to go forward without causing abends.
>
> OK, I'll add it back, and do some little testing to ensure it is working.

I'm not sure what you are expecting from the methods_attributes.rules... the following code:

```
def var hc as com-handle.
def var i as int.

hc:mthd1(i):mthd2():attr1:attr2 = i.
i = hc:mthd1(i):mthd2():attr1:attr2.
hc:mthd1(i):mthd2:attr1:attr2 = i = hc:mthd3(i):mthd4():attr5:attr6.
```

converts to (with no abends):

```
hc.assign(i, i);
i.assign(hc, i);
hc.assign(i, isEqual(i, hc, i));
```

for all cases: latest P2J, this update, or if I use this update with a check for the COM_INVOCATION nodes added at the beginning of the descent-rules:

```
   <descent-rules>
      <rule>type == prog.colon or type == prog.com_invocation
```

Thus, is there anything else I should do at this point? Because I doubt the COM_INVOCATION/COM_METHOD/COM_PARAMETER/COM_PROPERTY nodes should be treated by methods_attributes, as they are a completely different breed of handles. Considering that the COM_METHOD/COM_PROPERTY can be anything (so no compile-time checks are done by 4GL) and the referent of a COM-HANDLE var is unknown at conversion time, I think these will end up converted either to dedicated APIs (like OCXComponent.someComMethodName(<com-handle-expr>)) or to common APIs, like comhandle.comMethod(<com-handle-expr>, "<some-com-method-name>").

#7 - 05/30/2014 03:29 PM - Constantin Asofiei

Constantin Asofiei wrote:

> What I think ECW needs is just some simplifications of the cases when COLON nodes are involved. BTW, are there other POLY tests, beside the ones at #2050?

The only test I was missing from the ones you mention at note 5 is the dynamic_function_context_analysis.p test - and this converts properly with this update. Looking into ECW, the COLON nodes are used like this:

1. expressionType:885 has this:

```
         case COLON:
            jcls  = expressionType(source.getChildAt(1), infer, fuzzy);
            break;
```

   which is consistent how the AST looks today (the right child of a COLON node is what it determines its type).
2. analyzeContext:1447 has this:

```
            // quick out for ATTR_POLY that is index 0 with a COLON parent, this
            // by definition must be a handle
            if (source.getType() == ATTR_POLY && ptype == COLON && index == 0)
            {
               return "handle";
            }
```

   this is OK, because if a COLON's first child is ATTR_POLY, then this is acting as the 'lvalue', and is safely assumed as handle.
3. analyzeContext:1459 has this:

```
            // other ATTR_POLY cases must be checked below, but we must walk up
            // to get above the COLON ancestors; any chaining that isn't with us
            // as a handle (see above) must be the terminal node of the chain, so
            // we only need to "look up" the chain

            while (ptype == LPARENS             ||
                   ptype == LBRACKET            ||
                   ptype == COLON               ||
                   ptype == EXPRESSION          ||
                   ptype == PARAMETER           ||
                   ptype == KW_IN               ||
                   ptype == DB_REF_NON_STATIC)
            {
               // note the order of operations here
               index = past.getIndexPos();
               past  = past.getParent();
               ptype = past.getType();
            }
```

   this is OK, as it looks for parent COLON nodes, to get above the chained expression.

Thus, do you see anything else which might need to be changed? Because my conclusion would remain the same: ECW is fully compatible with the ASTs produced by the current parser.

#8 - 05/30/2014 04:05 PM - Greg Shah

> Thus, is there anything else I should do at this point? Because I doubt the COM_INVOCATION/COM_METHOD/COM_PARAMETER/COM_PROPERTY nodes should be treated by methods_attributes, as they are a completely different breed of handles.

No. It was just provisional support to get through conversion so if that code was actually dead then it is fine for now.

I agree we will treat them separately when the time comes.

> Thus, do you see anything else which might need to be changed? Because my conclusion would remain the same: ECW is fully compatible with the ASTs produced by the current parser.

You are right. I should never have doubted it. :)

#9 - 05/30/2014 04:25 PM - Constantin Asofiei

- File ca_upd20140530b.zip added

OK, the attached version contains this:

1. progress.g history fixed
2. methods_attributes.rules - was added a note in the history entry that the com-handle mthds/props need to be treated elsewhere; plus, a check for var_com_handle was removed, as it is not needed here.

#10 - 05/30/2014 04:41 PM - Greg Shah

Code Review 0530b

I am fine with the changes. At this point, is there anything left to do?

#11 - 05/30/2014 04:52 PM - Constantin Asofiei

Greg Shah wrote:

> I am fine with the changes. At this point, is there anything left to do?

Conversion testing passed for both MAJIC and server project, and runtime testing I don't think is needed. So there is nothing else to do.

#12 - 05/30/2014 05:01 PM - Greg Shah

You're confident that the handle unwrapping changes are safe?

#13 - 05/30/2014 05:38 PM - Constantin Asofiei

Greg Shah wrote:

> You're confident that the handle unwrapping changes are safe?

MAJIC uses only unwrapWidget, and all the unwrap calls throughout P2J are accounted for (else P2J wouldn't have compiled). I've tested the unwrapWidget and some other APIs in a simple test and they work - so the other will work too, as they were modified in the same way (I've made sure to do a find/replace scenario when changing handle.java instead of doing it manually, so if one works, all will work).

#14 - 05/30/2014 05:40 PM - Greg Shah

OK, then go ahead, check in and distribute it.

#15 - 05/30/2014 05:56 PM - Constantin Asofiei

Greg Shah wrote:

> OK, then go ahead, check in and distribute it.

Committed to bzr rev 10538.

As a reminder, the issue at note 2 still remains.

#16 - 05/30/2014 07:39 PM - Greg Shah

What do you suggest for the assignment stmts issue? We can defer it, but we may just hit it in future code. If that were to happen, then we would end up spending more time just figuring out that this was the issue.

If we defer it, then we will probably at least want to detect the situation and issue a big conversion time warning so that we aren't surprised.

#17 - 05/31/2014 03:31 PM - Constantin Asofiei

- File ca_upd20140531b.zip added

Greg Shah wrote:

> What do you suggest for the assignment stmts issue?

The fix in the end was pretty simple - I just had to get rid of all the complications related to duplicating the KW_ASSIGN node; attached fix works fine with my tests, need to check it with the MAJIC and server sources too, before releasing it.

#18 - 06/01/2014 08:41 AM - Constantin Asofiei

0531b.passed conversion testing - MAJIC has 1 case and the server project has 2 cases where RecordBuffer.start/EndBatch pairs were emitted incorrectly, and are now fixed.

#19 - 06/02/2014 06:24 AM - Greg Shah

Excellent! Please check it in and distribute it.

#20 - 06/02/2014 06:40 AM - Constantin Asofiei

- Status changed from WIP to Review

Greg Shah wrote:

> Excellent! Please check it in and distribute it.

Committed to bzr rev 10539. The task can be closed.

#21 - 06/02/2014 08:36 AM - Greg Shah

- % Done changed from 0 to 100

- Status changed from Review to Closed

- Target version set to Milestone 8

#22 - 11/16/2016 11:50 AM - Greg Shah

- Target version changed from Milestone 8 to GUI Support for a Complex ADM2 App

Files

ges_upd20140428a.zip	426 KB	05/28/2014	Constantin Asofiei
ca_upd20140526f.zip	650 KB	05/28/2014	Constantin Asofiei
ca_upd20140530b.zip	648 KB	05/30/2014	Constantin Asofiei
ca_upd20140531b.zip	2.82 KB	05/31/2014	Constantin Asofiei