Base Language - Bug #2056

null value for an annotation while persisting the AST

02/26/2013 08:05 AM - Ovidiu Maxiniuc

Status:	Closed	**Start date:**	02/26/2013
Priority:	Normal	**Due date:**	
Assignee:	Greg Shah	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	Conversion Support for Server Features		
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

History

#1 - 02/26/2013 08:17 AM - Ovidiu Maxiniuc

This is a task that tracks the issues reported by Eric in #1985, note 19.
I tried to eliminate the issue by pinpointing where the null getter was created. During this dirty fix, I observed some strange thinks in the syntax, some widgets were not explicitly defined.

Working with Constantin we realized that the mesg1, mesg2 and mesg3 were defined *inline*, during the display statements that directly uses them.

The minimal test-case is:

```
DEFINE VARIABLE a AS CHARACTER NO-UNDO INIT "a".
DISPLAY a @ b AS CHARACTER.
b = "b".
```

which is converted into something like (simplified code):

```
character a = new character("a");
frame0.openScope();
FrameElement[] elementList0 = new FrameElement[]
{
   new Element(a, "x(8)")
};
frame0.display(elementList0);
b.assign("b");
```

#2 - 02/26/2013 08:25 AM - Constantin Asofiei

Note that the following case is converted correctly by P2J:

```
display msg as char.
msg = "".
```

It seems to be a problem related to the @base-field specification.

#3 - 02/26/2013 10:12 AM - Ovidiu Maxiniuc

I looked into manuals about the base-field specification, but they only describe how data is displayed into the respective place, I did not found any useful information into @4GL Reference and Progress Language Tutorial about these defined variables. However, some examples, exists, and I found even another form:

```
DEFINE VARIABLE c AS CHARACTER NO-UNDO INIT "C".
DEFINE VARIABLE a AS CHARACTER NO-UNDO INIT "A".
DISPLAY a @ b LIKE c.
```

The newly created variable can be supplied with some other attributes like LABEL:

```
DISPLAY a @ b LIKE c LABEL "blabel".
MESSAGE b:LABEL.
```

#4 - 02/26/2013 10:13 AM - Constantin Asofiei

Ovidiu, for now try and fix the "display c @ b as datatype" case, leave that for later.

#5 - 02/26/2013 02:36 PM - Ovidiu Maxiniuc

- File om_upd20130226a.zip added

- File om_upd20130226b.zip added

The NPE is thrown because the function "copy_as_widget" attempts to duplicate an incomplete (missing "getter" annotation) widget node. However, the true problem lies elsewhere: gen_var_getter is not called when needed or not doing the right things.

As the fist step, the progress.g must be updated so as the ast to be generated:

```
<ast col="0" hidden="true" id="532575944724" line="0" text="format phrase" type="FORMAT_PHRASE">
```

```
  <ast col="12" id="532575944725" line="2" text="@" type="AT">
    <ast col="14" id="532575944728" line="2" text="b" type="SYMBOL"/>
    <ast col="16" id="532575944731" line="2" text="as" type="KW_AS">
      <ast col="19" id="532575944733" line="2" text="char" type="KW_CHAR"/>
    </ast>
  </ast>
</ast>
```

I noted that the big difference between DISPLAY a b AS CHARACTER. and @display msg as char. is that the reference to the variable to be created is kept as subnode of AT instead of FORMAT_PHRASE, but the annotations added are always at the FORMAT_PHRASE level.

I tried to adjust a few functions like possible_var_def, var_def, var_definition but at the end of the day I could not find the solution.

I attached the (slightly) modified progress.g (there is room for more: add like clause, disable warnings) and my sample test cases.

#6 - 02/27/2013 09:36 AM - Greg Shah

I can help with this.

Constantin: do you have any other findings to report?

We do support @-base fields already to a great degree. Majic uses them heavily. This feature of the 4GL is not well documented. Actually, our documentation in the P2J Conversion Reference is probably more useful but I suspect it is still incomplete. It is important to note that the conversion processing of these fields is fragile. For example, there is a strange precedence to how format strings for the base field are calculated. It is not straightforward and it must take into account multiple format strings across the entire file... My point here is that we already have a great deal of TRPL code written to try to deal with these fields (and some of the runtime is dedicated to this too). We must be very cautious in changing the parser because the current TRPL code will most likely break if the structure changes. I prefer to avoid this.

#7 - 02/27/2013 09:47 AM - Constantin Asofiei

- File ca_test20130227b.zip added

I was trying to force the rules to see the var definition hidden in the format_phrase node, and emit widget + var for it. But at this point, looks like the frame still sees and treats the left-side expression for the @ clause as the widget, and not the right-side expression.

At this point, I was searching for the code which creates the widget in frame_scoping, but as I'm not the author of this file, is a little difficult to piece everything together and see how it works.

#8 - 02/27/2013 10:32 AM - Greg Shah

This code works in P2J:

```
def var txt as char init "something" format "x(10)".
display bf as character txt @ bf.
```

This is the functional equivalent of this:

```
def var txt as char init "something" format "x(10)".
display txt @ bf as character
```

Here is the Java:

```
        character txt = new character("something");

        character bf = new character("");

        public void init()
        {
           TransactionManager.register(txt, bf);
        }

        public void body()
        {
           frame0.openScope();

           FrameElement[] elementList0 = new FrameElement[]
           {
              new Element(txt, frame0.widgetBf())
           };

           frame0.display(elementList0);
        }
```

I thought this might be useful, because it shows how the tree would need to look to work with the code as it stands today.

I have to step out for a bit, but will pick this up again when I get back.

#9 - 02/27/2013 05:41 PM - Greg Shah

After looking at this in detail, it is clear that this is something that I need to work on. Both of you should work on other open issues.

The core of the problem lies in how the parser treats format phrases and various other syntax (@ based fields, aggregate phrases...) that can be somewhat intermixed with format phrases. More importantly, the more I look at the syntax that is possible, the more I find that our original parsing of @ base field should not be inside the format phrase. There are even some limits of what can appear before an @ base reference and these suggest that we need to treat things differently. We already handle things like LIKE clauses and so forth, but we don't properly handle that stuff after the @ base field, because we are treating the "widget" reference as the expression/lvalue that is before the @, but the 4GL is treating the reference after the @ as the widget. If I clean that up, it should solve most of the problems BUT it will most likely have major downstream impact on our frame processing rules.

I am also considering a quick and dirty fix, which will have much less impact but it will not fix all cases.

Regardless, this is so deep in the parser, that I really need to take the lead on this myself.

#10 - 02/27/2013 05:43 PM - Greg Shah

One other point: I think if we get the parser right, the downstream processing may be much simpler. Today, we have to do some strange "manual walking" of the tree to find the right nodes and process everything (e.g. format phrase processing). With the proper approach, this may be easier, but the cleanup is going to be very painful.

#11 - 02/28/2013 05:31 PM - Greg Shah

- File funky_base_field.p added

- File ges_upd20130228a.zip added

OK, some findings:

1. Generally, @ base-field can only be used in a DISPLAY stmt and in the column_spec of a DEFINE BROWSE ... DISPLAY.

2. This particular behavior where a new variable can be created at the same time as being an @ base-field reference, can only be done in the DISPLAY statement.

3. If an @ base-field clause is present, only a WHEN clause (e.g. WHEN my-bool-var) and/or a FORMAT clause (e.g. FORMAT "x(8)") can precede it. All other format phrase stuff can only appear after the @ base-field clause. Interestingly, the WHEN cannot appear after an base-field clause. And the FORMAT clause can appear before, after or both before and after. But the meaning of the format phrase is different when it is before then when it is after. This tells us that the @ base-field clause is a kind of "splitting" of what our parser creates as the format phrase. Everything before (and including the @ base-field clause) is related to the expression being displayed AT the base field. In other words, these elements of the format phrase are a format phrase for that expression (which is not a widget, but is going to be displayed in the named location referenced by the @ base-field clause). Everything after the @@ base-field clause is about the creation/configuration of a new variable (and widget).

4. Interestingly, the aggregate phrase can never appear in a widget definition that includes the @ base-field clause. It doesn't matter if you try to put it before or after the @ base-field clause, it causes a 4GL compiler error. This simplifies things greatly.

5. Although I could potentially clean up the tree structure we create during parsing, knowing these rules (and now that the meaning is more clear), we just have too much TRPL code written to the current awkward tree structure. So it is best to do something less intrusive if possible.

6. As note 8 above describes, in fact there is an alternate syntax that our current TRPL rules fully supports. This means that we can solve this problem by converting the failing syntax (which can be identified by an AT node with a SYMBOL child) to the alternate tree structure. The attached update does exactly this, during post parse fixups.

I am also attaching a testcase that is a bit more involved, and which is fixed by this update. Please note that all of the alternate syntax (LIKE clauses and other format phrase stuff) is fully supported using this fix. So it is a general purpose fix, even though it may not be the way we could possibly do the tree, it is a very good solution.

I am running conversion testing now.

#12 - 02/28/2013 06:47 PM - Greg Shah

Passed conversion testing and is checked into bzr as revision 10226.

#13 - 02/28/2013 06:48 PM - Greg Shah

- % Done changed from 0 to 100

- Status changed from New to Closed

- Assignee changed from Ovidiu Maxiniuc to Greg Shah

- Target version set to Milestone 4

#14 - 11/16/2016 10:56 AM - Greg Shah

- Target version changed from Milestone 4 to Conversion Support for Server Features

Files

om_upd20130226a.zip	266 KB	02/26/2013	Ovidiu Maxiniuc
om_upd20130226b.zip	999 Bytes	02/26/2013	Ovidiu Maxiniuc
ca_test20130227b.zip	333 KB	02/27/2013	Constantin Asofiei
ges_upd20130228a.zip	5.88 KB	02/28/2013	Greg Shah
funky_base_field.p	250 Bytes	02/28/2013	Greg Shah