User Interface - Bug #4498

Embedded assignment of extent in shared frame fails

01/09/2020 09:39 AM - 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 - 01/09/2020 09:55 AM - Roger Borrello

Description

An array of extent 2 or more does not get the "hardcoded" array widgets created in the shared frame.

Testcase

uast/sharedframes/shared_frame_broken_update.p

```
def shared frame sf1.
def shared var fieldx as char no-undo init "fxval".
def shared var myx as dec extent 2.

form fieldx with frame sf1.
display fieldx with frame sf1.

loopy:
repeat with frame sf1:
   update myx = 0.
   leave loopy.
end.
```

Generated Business Logic

```
            FrameElement[] elementList1 = new FrameElement[]
            {
               new Element(subscript(myx, 1), sf1Frame.widgetMyxArray0()),
               new EmbeddedAssignment(subscript(myx, 2), 0)
            };
```

Generated Shared Frame

```
public interface FrameSf1_1
extends CommonFrame
{
   public character getFieldx();

   public void setFieldx(character parm);

   public void setFieldx(String parm);
   public void setFieldx(BaseDataType parm);

   public FillInWidget widgetFieldx();

}
```

If you bump up the extent size, you just get more use of missing widgets in the business logic.

#2 - 01/09/2020 10:12 AM - Greg Shah

Questions:

- Does this failure only occur when this is an imported shared frame?
- What does the 4GL do here? It is my understanding that one could NOT add widgets to an imported shared frame in the 4GL. If that is correct, what does this code actually do in the 4GL?

#3 - 01/09/2020 10:34 AM - Roger Borrello

> - Does this failure only occur when this is an imported shared frame?

If imported means non-shared (or local), the business logic is:

```
            FrameElement[] elementList1 = new FrameElement[]
            {
               new Element(subscript(myx, 1), sf1Frame.widgetMyxArray0()),
               new EmbeddedAssignment(subscript(myx, 2), 0)
            };
```

The frame class has FillInWidgets: widgetMyxArray0(), and widgetMyxArray()

Note: I'm not sure that's correct, since it is creating an element, but not assigning it to 0 for Array0.

> - What does the 4GL do here? It is my understanding that one could NOT add widgets to an imported shared frame in the 4GL. If that is correct, what does this code actually do in the 4GL?

4GL doesn't complain for the non-shared frame or shared frame cases.

#4 - 01/09/2020 10:42 AM - Greg Shah

> > Does this failure only occur when this is an imported shared frame?
>
> If imported means non-shared (or local)

No, this is not correct.

There are 3 types of frame:

- non-shared (DEF FRAME, no use of the SHARED keyword)
- shared
  - DEF NEW SHARED FRAME - this creates the resource and in FWD, registers it with the SharedVariableManager at runtime
  - DEF SHARED FRAME - this IMPORTS the resource, in FWD it is looked up in the SharedVariableManager at runtime

So, "imported shared frame@ **IS** a shared frame, but it uses an existing frame that is created further up the call stack. As far as I understand, you cannot add widgets to such a frame. The def frame or form statement that you use to define the local structure of the imported frame must be a structural match to the one created externally.

> 4GL doesn't complain for the non-shared frame or shared frame cases.

Are you saying that the displayed frame has additional widgets?

#5 - 01/09/2020 10:59 AM - Roger Borrello

> There are 3 types of frame:
>
> - non-shared (DEF FRAME, no use of the SHARED keyword)
> - shared
>   - DEF NEW SHARED FRAME - this creates the resource and in FWD, registers it with the SharedVariableManager at runtime
>   - DEF SHARED FRAME - this IMPORTS the resource, in FWD it is looked up in the SharedVariableManager at runtime

> So, "imported shared frame@ **IS** a shared frame, but it uses an existing frame that is created further up the call stack. As far as I understand, you cannot add widgets to such a frame. The def frame or form statement that you use to define the local structure of the imported frame must be a structural match to the one created externally.

The frame definitions are exactly the same in the parent procedure as in child procedure. There is this situation in the child procedure where a widget not in the frame definition has an embedded assignment.

> > 4GL doesn't complain for the non-shared frame or shared frame cases.

> Are you saying that the displayed frame has additional widgets?

I don't see any additional widgets. But there are no errors.

#6 - 01/09/2020 02:31 PM - Roger Borrello

These testcases both result in 4GL errors:
cannot_add_widget_to_imported_frame.p

```
def shared frame sf1.
def shared var fieldx as char no-undo init "fxval".
def shared var myx as dec extent 2.

form fieldx with frame sf1.
display fieldx with frame sf1.

/*
 * A widget is not added to a shared frame by the frame importer.
 * 4GL gives "Shared frame sf1 field myx must first appear in a FORM statement (890)
 */
update myx with frame sf1.

message "done.".
```

cannot_add_widget_to_shared_frame.p

```
def shared frame sf1.
def shared var fieldx as char no-undo init "fxval".
def shared var myx as dec extent 2.

form fieldx with frame sf1.
display fieldx with frame sf1.
```

```
/*
 * A widget is not added to a shared frame by the frame owner unless frame importer also adds.
 * 4GL gives "Shared frame sf1 field myx must first appear in a FORM statement (890)
form myx with frame sf1.
 */
update myx with frame sf1.

message "done.".
```

cannot_add_widget_to_shared_frame-run.p

```
def new shared frame sf1.
def new shared var fieldx as char no-undo init "fxval".
def new shared var myx as dec extent 2.

form fieldx with frame sf1.

/*
 * A widget is not added to a shared frame by the frame owner unless frame importer also adds.
 * 4GL gives "Shared frame sf1 field myx must first appear in a FORM statement (890)
 */
form myx with frame sf1.

def var path as char init "testcases/uast/sharedframes/".
def var pname as char init "cannot_add_widget_to_shared_frame.p".
def var progname as char.
progname = "z:/" + path + pname.
if opsys = "unix" then
   progname = "~~" + path + progname.
if opsys <> "win32" then
   message opsys "is an unsupported OS".
else
   run value(progname).
message "Done.".
```

In other words, unless both the creator and importer of the frame have the adde widget, you get an error.

#7 - 01/09/2020 02:36 PM - Roger Borrello

In this testcase, you can add a widget to non-shared frame, and update the variables.

nonshared_frame_embedded_update.p

```
def frame f1.
def var fieldx as char no-undo init "fxval".
def var myx as int extent 2 init -1.
def var i as int init -1.

form fieldx with frame f1.

/*
 * i is added as a widget, and retains what you enter.
 * When done hits, i is what you entered, and myx[1] and myx[2] are 1.
 */

update i myx = 1 with frame f1.
view frame f1.
message "Done." i myx[1] myx[2].
```

#8 - 01/09/2020 02:55 PM - Roger Borrello

Shared Frame

The only difference between these 2 testcases is the embedded assignment. However, the one with the embedded assignment does not complain about "Shared frame sf1 field myx must first appear in a FORM statement (890)"

cannot_add_widget_to_shared_frame.p

```
def shared frame sf1.
def shared var fieldx as char no-undo init "fxval".
def shared var myx as dec extent 2 init -1.

form fieldx with frame sf1.

/*
 * A widget is not added to a shared frame by the frame owner unless frame importer also adds.
 * 4GL gives "Shared frame sf1 field myx must first appear in a FORM statement (890)
form myx with frame sf1.
 */
update myx with frame sf1.

message "done." myx[1] myx[2].
```

shared_frame_no_widget_embedded_update.p

```
def shared frame sf1.
def shared var fieldx as char no-undo init "fxval".
def shared var myx as dec extent 2 init -1.

form fieldx with frame sf1.

/*
 * A widget is not added to a shared frame. But, no error is given
 * and the myx[1] and myx[2] values are modified to 1.
 */
update myx = 1 with frame sf1.

message "done." myx[1] myx[2].
```

#9 - 01/09/2020 03:37 PM - Roger Borrello

Non-Shared Frame

Again, the only difference between these 2 testcases is the embedded assignment. However, unlike the Shared Frame, you can add the widget to the frame with an UPDATE. However, the UPDATE myx = 1 does not add a widget to the frame, but the variable is modified.

shared_frame_no_widget_embedded_update.p

```
def shared frame sf1.
def shared var fieldx as char no-undo init "fxval".
def shared var myx as int extent 2 init -1.

form fieldx with frame sf1.

/*
 * A widget is not added to a shared frame. But, no error is given
 * and the myx[1] and myx[2] values are modified to 1.
 */
update myx = 1 with frame sf1.

message "done." myx[1] myx[2].
```

can_add_widget_to_nonshared_frame.p

```
def frame f1.
def var fieldx as char no-undo init "fxval".
def var myx as int extent 2 init -1.

form fieldx with frame f1.

/*
 * A widget is added to a shared frame.
 * When done hits, myx[1] and myx[2] are what you entered.
 */
update myx with frame f1.

message "Done." myx[1] myx[2].
```

#10 - 01/09/2020 05:45 PM - Roger Borrello

Testcase Results

Shared Frames (Standalone)

Both testcases are very similar, with the exception of the embedded assignment. They define a shared frame and act upon it.

The results were the same whether it was a scalar int or an EXTENT.

shared_frame_cannot_add_widget-standalone.p

```
def new shared frame sf1.
def new shared var fieldx as char no-undo init "fxval".
def new shared var myx as int extent 2 init -1.

form fieldx with frame sf1.

/*
 * A widget cannot be added to a shared frame.
 * 4GL gives "Shared frame sf1 field myx must first appear in a FORM statement (890)
 */
update myx with frame sf1.

message "Done." myx[1] myx[2].
```

shared_frame_no_widget_embedded_assign-standalone.p

```
def new shared frame sf1.
def new shared var fieldx as char no-undo init "fxval".
def new shared var myx as int extent 2 init -1.

form fieldx with frame sf1.

/*
 * A widget is not added to a shared frame. But, no error is given
 * and the myx[1] and myx[2] values are modified to 1.
 */
update myx = 1 with frame sf1.

message "Done." myx[1] myx[2].
```

Non-Shared Frame

Again, the first 2 testcases are very similar, with the exception of the embedded assignment. The results were the same whether it was a scalar int or an EXTENT.

The last testcase just illustrates what occurs when an additional widget is on the line, without an embedded assignment.

nonshared_frame_can_add_widget.p

```
def frame f1.
def var fieldx as char no-undo init "fxval".
def var myx as int extent 2 init -1.

form fieldx with frame f1.

/*
 * A widget for myx is added to the frame
 * and the myx[1] and myx[2] values are modified to 1.
 */
update myx with frame f1.

message "Done." myx[1] myx[2].
```

nonshared_frame_no_widget_embedded_assign.p

```
def frame f1.
def var fieldx as char no-undo init "fxval".
def var myx as int extent 2 init -1.

form fieldx with frame f1.

/*
 * No widget is added to the frame, but myx[1] and myx[2] are 1.
 */
update myx = 1 with frame f1.

message "Done." myx[1] myx[2].
```

nonshared_frame_extra_widget_embedded_assign.p

```
def frame f1.
def var fieldx as char no-undo init "fxval".
def var myx as int extent 2 init -1.
def var i as int init -1.

form fieldx with frame f1.

/*
 * i is added as a widget, and the frame is displayed.
 * When done hits, i is what you entered, and myx[1] and myx[2] are 1.
 */

update i myx = 1 with frame f1.

message "Done." i myx[1] myx[2].
```

Shared Imported Frames

These testcases involve the use of a parent procedure that runs the child procedure, which is importing the shared frame. They are not included here for brevity. They are named the same as the child procedure, but with -run added to the name. They define the frame the same as the child.

Again, there was no difference when a scalar was used instead of an EXTENT.

imported_frame_cannot_add_widget_to_shared_frame.p

```
def shared frame sf1.
def shared var fieldx as char no-undo init "fxval".
def shared var myx as int extent 2 init -1.

form fieldx with frame sf1.

/*
 * A widget cannot be added to a shared frame by the frame importer.
 * 4GL gives "Shared frame sf1 field myx must first appear in a FORM statement (890)
 */
update myx with frame sf1.

message "done." myx[1] myx[2].
```

imported_frame_no_widget_embedded_assign.p

```
def shared frame sf1.
def shared var fieldx as char no-undo init "fxval".
def shared var myx as int extent 2 init -1.

form fieldx with frame sf1.

/*
 * A widget is not added to the shared frame. But, no error is given
 * and the myx[1] and myx[2] values are modified to 1.
 */
update myx = 1 with frame sf1.

message "done." myx[1] myx[2].
```

#11 - 01/10/2020 10:11 AM - Roger Borrello

Received the following when I was adding scalar testcases...

```
     [java] ./uast/sharedframes/imported_frame_no_widget_embedded_assign-scalar.p
     [java] Elapsed job time:  00:00:00.701
     [java] EXPRESSION EXECUTION ERROR:
     [java] ---------------------------
     [java] throwException(errmsg)
     [java] ^  { Missing frame-id for node id 12884901983 ( [FRAME_SCOPE]:12884901983 @0:0
     [java] ) [FRAME_SCOPE id <12884901983> 0:0] }
     [java] ---------------------------
     [java] EXPRESSION EXECUTION ERROR:
     [java] ---------------------------
     [java] javaname = execLib("get_framename", this)
     [java]            ^  { Expression execution error @1:1 }
     [java] ---------------------------
     [java] ERROR:
     [java] com.goldencode.p2j.pattern.TreeWalkException: ERROR!  Active Rule:
     [java] -----------------------
     [java]       RULE REPORT
     [java] -----------------------
     [java] Rule Type :   WALK
     [java] Source AST:  [  ] BLOCK/FRAME_SCOPE/ @0:0 {12884901983}
     [java] Copy AST  :  [  ] BLOCK/FRAME_SCOPE/ @0:0 {12884901983}
     [java] Condition :  throwException(errmsg)
     [java] Loop      :  false
     [java] --- END RULE REPORT ---
```

Testcase is:

```
def shared frame sf1.
def shared var fieldx as char no-undo init "fxval".
def shared var myx as int init -1.
form fieldx with frame sf1.
update myx = 1 with frame sf1.
message "done." myx.
```

I tested in trunk, and it fails there. I'll try to track this down.

#12 - 01/10/2020 10:25 AM - Roger Borrello

Roger Borrello wrote:

> Received the following when I was adding scalar testcases...
>
> [...]
>
> Testcase is:
> [...]
>
> I tested in trunk, and it fails there. I'll try to track this down.

I will put this on the back burner, but wanted to make a note that it occurs with this testcase, as well:

```
def frame f1.
def var fieldx as char no-undo init "fxval".
def var myx as int init -1.
form fieldx with frame f1.
update myx = 1 with frame f1.
message "Done." myx.
```

Below are the procedures I could not convert:

- ./uast/sharedframes/shared_frame_no_widget_embedded_assign-scalar-standalone.p
- ./uast/sharedframes/imported_frame_no_widget_embedded_assign-scalar.p
- ./uast/sharedframes/imported_frame_no_widget_embedded_assign-scalar-run.p
- ./uast/sharedframes/nonshared_frame_no_widget_embedded_assign-scalar.p

#13 - 01/10/2020 11:55 AM - Roger Borrello

Shared Frame (Standalone)

Note that I could not get embedded assignment scalar versions to convert, per #4498#Note-12.

No Embedded Assignment

shared_frame_cannot_add_widget-standalone.p and shared_frame_cannot_add_widget-scalar-standalone.p

```
/*
```

```
 * A widget cannot be added to a shared frame.
 * 4GL gives "Shared frame sf1 field myx must first appear in a FORM statement (890)
 */
update myx1 with frame sf1.
```

Generated:

Extent Version

```
         FrameElement[] elementList0 = new FrameElement[]
         {
            new Element(subscript(myx1, 1), sf1Frame.widgetMyx1Array0()),
            new Element(subscript(myx1, 2), sf1Frame.widgetMyx1Array1())
         };

         /*
          * A widget cannot be added to a shared frame.
          * 4GL gives "Shared frame sf1 field myx must first appear in a FORM statement (890)
          */
         sf1Frame.update(elementList0);
```

FrameSf1_1 contains widget widgetMyx1Array0

Scalar Version

```
         FrameElement[] elementList0 = new FrameElement[]
         {
            new Element(myx1s, sf1sFrame.widgetMyx1s())
         };

         /*
          * A widget cannot be added to a shared frame.
          * 4GL gives "Shared frame sf1 field myx must first appear in a FORM statement (890)
          */
         sf1sFrame.update(elementList0);
```

FrameSf1s_1 contains widget widgetMyx1s

Embedded Assignment

shared_frame_no_widget_embedded_assign-standalone.p

```
/*
 * A widget is not added to a shared frame. But, no error is given
 * and the myx[1] and myx[2] values are modified to 1.
 */
update myx2 = 1 with frame sf2.
```

Generated:

```
         FrameElement[] elementList0 = new FrameElement[]
         {
            new Element(subscript(myx2, 1), sf2Frame.widgetMyx2Array0()),
            new EmbeddedAssignment(subscript(myx2, 2), 1)
         };

         /*
          * A widget is not added to a shared frame. But, no error is given
          * and the myx[1] and myx[2] values are modified to 1.
          */
         sf2Frame.update(elementList0);
```

FrameSf2_1 contains widget widgetMyx2Array0

Shared Frame (Imports)

No Embedded Assignment

imported_frame_cannot_add_widget_to_shared_frame.p and imported_frame_cannot_add_widget_to_shared_frame-scalar.p

```
/*
 * A widget cannot be added to a shared frame by the frame importer.
 * 4GL gives "Shared frame sf1 field myx must first appear in a FORM statement (890)
 */
update myx4 with frame sf4.
```

Generated:

Extent Version

```
         FrameElement[] elementList0 = new FrameElement[]
         {
            new Element(subscript(myx4, 1), sf4Frame.widgetMyx4Array0()),
            new Element(subscript(myx4, 2), sf4Frame.widgetMyx4Array1())
         };

         /*
          * A widget cannot be added to a shared frame by the frame importer.
          * 4GL gives "Shared frame sf1 field myx must first appear in a FORM statement (890)
          */
         sf4Frame.update(elementList0);
```

FrameSf4_2 contains widget widgetMyx4Array0

Scalar Version

```
         FrameElement[] elementList0 = new FrameElement[]
         {
            new Element(myx4s, sf4sFrame.widgetMyx4s())
         };

         /*
          * A widget cannot be added to a shared frame by the frame importer.
          * 4GL gives "Shared frame sf1 field myx must first appear in a FORM statement (890)
          */
         sf4sFrame.update(elementList0);
```

FrameSf4s_1 does **not** contain a widget for myx4s

Embedded Assignment

shared_frame_no_widget_embedded_assign.p

```
/*
 * A widget is not added to the shared frame. But, no error is given
 * and the myx[1] and myx[2] values are modified to 1.
 */
update myx3 = 1 with frame sf3.
```

Generated:

```
         FrameElement[] elementList0 = new FrameElement[]
         {
            new Element(subscript(myx3, 1), sf3Frame.widgetMyx3Array0()),
            new EmbeddedAssignment(subscript(myx3, 2), 1)
         };

         /*
          * A widget is not added to the shared frame. But, no error is given
          * and the myx[1] and myx[2] values are modified to 1.
          */
         sf3Frame.update(elementList0);
```

FrameSf3_1 does **not** contain a widget for myx3.

Non-Shared Frame

No Embedded Assignment

nonshared_frame_can_add_widget.p and nonshared_frame_can_add_widget-scalar.p

```
/*
 * A widget for myx is added to the frame
 * and the myx[1] and myx[2] values are modified to 1.
 */
update myx5 with frame f5.
```

Generated:

Extent Version

```
         FrameElement[] elementList0 = new FrameElement[]
         {
            new Element(subscript(myx5, 1), f5Frame.widgetMyx5Array0()),
            new Element(subscript(myx5, 2), f5Frame.widgetMyx5Array1())
         };

         /*
          * A widget for myx is added to the frame
          * and the myx[1] and myx[2] values are modified to 1.
          */
         f5Frame.update(elementList0);
```

NonsharedFrameCanAddWidgetF5 contains widget widgetMyx5Array0

Scalar Version

```
         FrameElement[] elementList0 = new FrameElement[]
         {
            new Element(myx5s, f5sFrame.widgetMyx5s())
         };

         /*
          * A widget for myx is added to the frame
          * and the myx[1] and myx[2] values are modified to 1.
          */
         f5sFrame.update(elementList0);
```

NonsharedFrameCanAddWidgetScalarF5s contains widget widgetMyx5s

Embedded Assignment

nonshared_frame_no_widget_embedded_assign.p

```
/*
 * No widget is added to the frame, but myx[1] and myx[2] are 1.
 */
update myx6 = 1 with frame f6.
```

Generated:

```
         FrameElement[] elementList0 = new FrameElement[]
         {
            new Element(subscript(myx6, 1), f6Frame.widgetMyx6Array0()),
            new EmbeddedAssignment(subscript(myx6, 2), 1)
         };

         /*
          * No widget is added to the frame, but myx[1] and myx[2] are 1.
          */
         f6Frame.update(elementList0);
```

NonsharedFrameNoWidgetEmbeddedAssignF6 contains widget widgetMyx6Array0

#14 - 01/10/2020 01:08 PM - Roger Borrello

In running the "extra widget" testcase, there is something strange generated in the AST.

Line 13 of the testcase is update i myx = 1 with frame f1. where myx is int extent 2.

The AST has the below section. Strange that there are duplicated VAR_INT for i in the FRAME_ELEMENT, and the only difference in the annotations is the second one has a frame-id (and a different peer-id).

```
    <ast col="0" id="17179869264" line="0" text="statement" type="STATEMENT">
      <ast col="0" id="17179869323" line="0" text="" type="CONTENT_ARRAY">
        <annotation datatype="java.lang.String" key="javaname" value="elementList0"/>
        <annotation datatype="java.lang.Long" key="peerid" value="167503724616"/>
        <ast col="0" id="17179869324" line="0" text="" type="FRAME_ELEMENT">
          <annotation datatype="java.lang.Long" key="peerid" value="167503724617"/>
          <ast col="8" id="17179869325" line="13" text="i" type="VAR_INT">
            <annotation datatype="java.lang.Long" key="oldtype" value="2782"/>
            <annotation datatype="java.lang.Long" key="refid" value="17179869233"/>
            <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
            <annotation datatype="java.lang.String" key="getter" value="getI"/>
            <annotation datatype="java.lang.String" key="setter" value="setI"/>
          <annotation datatype="java.lang.String" key="widgettype" value="FillInWidget"/>
          <annotation datatype="java.lang.String" key="javaname" value="i"/>
          <annotation datatype="java.lang.String" key="accessor" value="widgeti"/>
          <annotation datatype="java.lang.Boolean" key="firstref" value="true"/>
          <annotation datatype="java.lang.Boolean" key="lastref" value="true"/>
          <annotation datatype="java.lang.Long" key="peerid" value="167503724618"/>
        </ast>
        <ast col="8" id="17179869326" line="13" text="i" type="VAR_INT">
          <annotation datatype="java.lang.Long" key="oldtype" value="2782"/>
          <annotation datatype="java.lang.Long" key="refid" value="17179869233"/>
          <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
          <annotation datatype="java.lang.String" key="getter" value="getI"/>
          <annotation datatype="java.lang.String" key="setter" value="setI"/>
          <annotation datatype="java.lang.String" key="widgettype" value="FillInWidget"/>
          <annotation datatype="java.lang.String" key="javaname" value="i"/>
          <annotation datatype="java.lang.String" key="accessor" value="widgeti"/>
          <annotation datatype="java.lang.Boolean" key="firstref" value="true"/>
          <annotation datatype="java.lang.Boolean" key="lastref" value="true"/>
          <annotation datatype="java.lang.Long" key="frame-id" value="17179869322"/>
          <annotation datatype="java.lang.Long" key="peerid" value="167503724619"/>
        </ast>
      </ast>
      <ast col="0" id="17179869327" line="0" text="" type="FRAME_ELEMENT">
        <annotation datatype="java.lang.Long" key="peerid" value="167503724621"/>
        <ast col="10" id="17179869328" line="13" text="myx" type="VAR_INT">
          <annotation datatype="java.lang.Long" key="oldtype" value="2782"/>
          <annotation datatype="java.lang.Long" key="extent" value="2"/>
          <annotation datatype="java.lang.Long" key="refid" value="17179869213"/>
          <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
          <annotation datatype="java.lang.String" key="getter" value="getMyxArray0"/>
          <annotation datatype="java.lang.String" key="setter" value="setMyxArray0"/>
          <annotation datatype="java.lang.String" key="widgettype" value="FillInWidget"/>
          <annotation datatype="java.lang.String" key="javaname" value="myxArray0"/>
          <annotation datatype="java.lang.String" key="accessor" value="widgetMyxArray0"/>
          <annotation datatype="java.lang.Boolean" key="firstref" value="true"/>
          <annotation datatype="java.lang.Boolean" key="lastref" value="true"/>
          <annotation datatype="java.lang.Long" key="peerid" value="167503724623"/>
          <ast col="0" id="17179869329" line="0" text="[" type="LBRACKET">
            <annotation datatype="java.lang.Boolean" key="is-expanded" value="true"/>
            <annotation datatype="java.lang.Long" key="peerid" value="167503724622"/>
            <ast col="0" id="17179869330" line="0" text="" type="EXPRESSION">
              <annotation datatype="java.lang.Long" key="peerid" value="167503724624"/>
              <ast col="0" id="17179869331" line="0" text="1" type="NUM_LITERAL">
                <annotation datatype="java.lang.Boolean" key="use64bit" value="false"/>
                <annotation datatype="java.lang.Long" key="peerid" value="167503724625"/>
              </ast>
            </ast>
          </ast>
        </ast>
        <ast col="10" id="17179869332" line="13" text="myx" type="VAR_INT">
          <annotation datatype="java.lang.Long" key="oldtype" value="2782"/>
          <annotation datatype="java.lang.Long" key="extent" value="2"/>
          <annotation datatype="java.lang.Long" key="refid" value="17179869213"/>
          <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
          <annotation datatype="java.lang.String" key="getter" value="getMyxArray0"/>
          <annotation datatype="java.lang.String" key="setter" value="setMyxArray0"/>
          <annotation datatype="java.lang.String" key="widgettype" value="FillInWidget"/>
          <annotation datatype="java.lang.String" key="javaname" value="myxArray0"/>
          <annotation datatype="java.lang.String" key="accessor" value="widgetMyxArray0"/>
          <annotation datatype="java.lang.Boolean" key="firstref" value="true"/>
          <annotation datatype="java.lang.Boolean" key="lastref" value="true"/>
          <annotation datatype="java.lang.Long" key="frame-id" value="17179869322"/>
          <annotation datatype="java.lang.Long" key="peerid" value="167503724626"/>
        </ast>
      </ast>
      <ast col="14" id="17179869271" line="13" text="=" type="EMBEDDED_ASSIGNMENT">
        <annotation datatype="java.lang.Boolean" key="direct_emit" value="true"/>
        <annotation datatype="java.lang.Long" key="peerid" value="167503724628"/>
        <ast col="10" id="17179869309" line="13" text="myx" type="VAR_INT">
          <annotation datatype="java.lang.Long" key="oldtype" value="2782"/>
          <annotation datatype="java.lang.Long" key="extent" value="2"/>
          <annotation datatype="java.lang.Long" key="refid" value="17179869213"/>
          <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
          <annotation datatype="java.lang.Long" key="peerid" value="167503724630"/>
          <ast col="0" id="17179869310" line="0" text="[" type="LBRACKET">
          <annotation datatype="java.lang.Boolean" key="is-expanded" value="true"/>
          <annotation datatype="java.lang.Long" key="peerid" value="167503724629"/>
          <ast col="0" id="17179869311" line="0" text="" type="EXPRESSION">
            <annotation datatype="java.lang.Long" key="peerid" value="167503724631"/>
            <ast col="0" id="17179869312" line="0" text="2" type="NUM_LITERAL">
              <annotation datatype="java.lang.Boolean" key="use64bit" value="false"/>
              <annotation datatype="java.lang.Long" key="peerid" value="167503724632"/>
            </ast>
          </ast>
        </ast>
      </ast>
      <ast col="0" id="17179869273" line="0" text="expression" type="EXPRESSION">
        <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
        <annotation datatype="java.lang.Long" key="peerid" value="167503724633"/>
        <ast col="16" id="17179869274" line="13" text="1" type="NUM_LITERAL">
          <annotation datatype="java.lang.Boolean" key="is-literal" value="true"/>
          <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
          <annotation datatype="java.lang.Boolean" key="use64bit" value="false"/>
          <annotation datatype="java.lang.Long" key="peerid" value="167503724634"/>
        </ast>
      </ast>
    </ast>
  </ast>
  <ast col="1" id="17179869265" line="13" text="update" type="KW_UPDATE">
    <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
    <annotation datatype="java.lang.Long" key="scope-id" value="17179869321"/>
    <annotation datatype="java.lang.Long" key="frame-id" value="17179869322"/>
    <annotation datatype="java.lang.Long" key="block_par_id" value="167503724580"/>
    <annotation datatype="java.lang.Long" key="peerid" value="167503724636"/>
    <ast col="8" hidden="true" id="17179869267" line="13" text="i" type="VAR_INT">
      <annotation datatype="java.lang.Long" key="oldtype" value="2782"/>
      <annotation datatype="java.lang.Long" key="refid" value="17179869233"/>
      <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
      <annotation datatype="java.lang.String" key="getter" value="getI"/>
      <annotation datatype="java.lang.String" key="setter" value="setI"/>
      <annotation datatype="java.lang.String" key="widgettype" value="FillInWidget"/>
      <annotation datatype="java.lang.String" key="javaname" value="i"/>
      <annotation datatype="java.lang.String" key="accessor" value="widgeti"/>
      <annotation datatype="java.lang.Boolean" key="firstref" value="true"/>
      <annotation datatype="java.lang.Boolean" key="lastref" value="true"/>
      <annotation datatype="java.lang.Long" key="frame-id" value="17179869322"/>
    </ast>
    <ast col="10" hidden="true" id="17179869316" line="13" text="myx" type="VAR_INT">
      <annotation datatype="java.lang.Long" key="oldtype" value="2782"/>
      <annotation datatype="java.lang.Long" key="extent" value="2"/>
      <annotation datatype="java.lang.Long" key="refid" value="17179869213"/>
      <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
      <annotation datatype="java.lang.String" key="getter" value="getMyxArray0"/>
      <annotation datatype="java.lang.String" key="setter" value="setMyxArray0"/>
      <annotation datatype="java.lang.String" key="widgettype" value="FillInWidget"/>
      <annotation datatype="java.lang.String" key="javaname" value="myxArray0"/>
      <annotation datatype="java.lang.String" key="accessor" value="widgetMyxArray0"/>
      <annotation datatype="java.lang.Boolean" key="firstref" value="true"/>
      <annotation datatype="java.lang.Boolean" key="lastref" value="true"/>
      <annotation datatype="java.lang.Long" key="frame-id" value="17179869322"/>
      <ast col="0" hidden="true" id="17179869317" line="0" text="[" type="LBRACKET">
        <annotation datatype="java.lang.Boolean" key="is-expanded" value="true"/>
        <ast col="0" id="17179869318" line="0" text="" type="EXPRESSION">
          <ast col="0" id="17179869319" line="0" text="1" type="NUM_LITERAL">
            <annotation datatype="java.lang.Boolean" key="use64bit" value="false"/>
          </ast>
        </ast>
      </ast>
    </ast>
    <ast col="18" hidden="true" id="17179869276" line="13" text="with" type="FRAME_PHRASE">
      <annotation datatype="java.lang.Long" key="frame-id" value="17179869322"/>
      <ast col="23" hidden="true" id="17179869278" line="13" text="frame" type="KW_FRAME">
        <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
        <ast col="29" id="17179869280" line="13" text="f1" type="WID_FRAME"/>
      </ast>
    </ast>
  </ast>
</ast>
```

#15 - 01/10/2020 01:16 PM - Greg Shah

> Strange that there are duplicated VAR_INT for i in the FRAME_ELEMENT, and the only difference in the annotations is the second one has a frame-id (and a different peer-id).

Nothing wrong there. It is how it is supposed to be. Remember that we have to emit 2 items, the var and the widget, both patterned off the original widget reference. The first one has frame-id removed which we know will cause it to NOT emit as a widget. This is what we want.

#16 - 01/10/2020 03:12 PM - Roger Borrello

In focusing on the incorrect widget for myx, the add_widget function in **annotations/frame_scoping.rules**, the UPDATE node has 2 children processed. Here is the complete node:

```
     [java] update [KW_UPDATE]:17179869255 @11:1
     [java]    myx3 [VAR_INT]:17179869301 @11:8
     [java]       [ [LBRACKET]:17179869302 @0:0
     [java]           [EXPRESSION]:17179869303 @0:0
     [java]             1 [NUM_LITERAL]:17179869304 @0:0
     [java]    myx3 [VAR_INT]:17179869294 @11:8
     [java]       [ [LBRACKET]:17179869295 @0:0
     [java]           [EXPRESSION]:17179869296 @0:0
     [java]             2 [NUM_LITERAL]:17179869297 @0:0
     [java]    = [ASSIGN]:17179869259 @11:13
     [java]       expression [EXPRESSION]:17179869261 @0:0
     [java]          1 [NUM_LITERAL]:17179869262 @11:15
     [java]    with [FRAME_PHRASE]:17179869264 @11:17
     [java]       frame [KW_FRAME]:17179869266 @11:22
     [java]          sf3 [WID_FRAME]:17179869268 @11:28
```

The first time, myx is added. It's the first VAR_INT:

```
     [java] frame_scoping.add_widget: adding widget (jName=expr2)
     [java] myx3 [VAR_INT]:17179869301 @11:8
     [java]    [ [LBRACKET]:17179869302 @0:0
     [java]        [EXPRESSION]:17179869303 @0:0
     [java]          1 [NUM_LITERAL]:17179869304 @0:0
```

The second time, we don't add myx because it's sibling type is prog.assign:

```
     [java] frame_scoping.add_widget: We are NOT adding due to sibling type=prog.assign
     [java] myx3 [VAR_INT]:17179869294 @11:8
     [java]    [ [LBRACKET]:17179869295 @0:0
     [java]        [EXPRESSION]:17179869296 @0:0
     [java]          2 [NUM_LITERAL]:17179869297 @0:0
```

#17 - 01/10/2020 04:08 PM - Roger Borrello

In annotations/array_expansion.rules I updated the below to protect the setting of attachIdx and parentRef:

```
            <!-- set, update, prompt-for, disable, enable and assign all
                 have direct attachment (no intermediate nodes) -->
            <rule>
               (parent.type == prog.kw_set       or
                parent.type == prog.kw_update    or
                parent.type == prog.kw_prmt_for  or
                parent.type == prog.kw_disable   or
                parent.type == prog.kw_enable    or
                parent.type == prog.kw_assign) and
               parent.parent.type == prog.statement

               <!-- We need to prevent expansion if we are going to do an embedded assignment -->
               <rule>!(parent.type == prog.kw_update and copy.nextSibling.type == prog.assign)
                  <action>attachIdx = copy.indexPos</action>
                  <action>parentRef = copy.parent</action>
               </rule>
            </rule>
```

Now we are getting the same code conversion error as the scalar testcases (against trunk) where there is a missing frame-id. Probably someone looking for an annotation we don't have anymore, without error checking.

#18 - 01/10/2020 06:10 PM - Roger Borrello

Roger Borrello wrote:

> In annotations/arrray_expansion.rules I updated the below to protect the setting of attachIdx and parentRef:
> [...]
>
> Now we are getting the same code conversion error as the scalar testcases (against trunk) where there is a missing frame-id. Probably someone looking for an annotation we don't have anymore, without error checking.

The "get_framename" function is being called from convert/frame_construction.rules. The AST has an empty FRAME_SCOPE which is being created in annotations/frame_scoping.rules. There are 3 places that are candidates for creating this:

- process_scope
- process_fake_scope
- scope_frame

My debug removes process_scope so it's one of the others. I added some debug for and our testcase yields:

```
     [java] frame_scoping: (process frame phrase) get_frame=
     [java] {IsRootScope=false, New=false, Disp=false, Down=-1, Ref=[FRAME_PHRASE], Wid={expr1=EXPRESSION}, De
f=false, Expr=0, Index={}, Shared=false, Name=sf3, Processed=false}
     [java] frame_scoping.process_fake_scope:
     [java]  [FRAME_SCOPE]:12884902110 @0:0
     [java] 
     [java] ## INFO: ./uast/sharedframes/imported_frame_no_widget_embedded_assign-run.p: assuming DOWN set to 
1 for sf3
     [java] frame_scoping (in loop)!: get_named_frame=
     [java] {IsRootScope=true, New=true, Disp=true, Down=1, Def=true, Index={}, Shared=true, Name=sf3, Ref=[DE
FINE_FRAME, FRAME_SCOPE, FRAME_PHRASE, KW_FORM], Wid={expr1=EXPRESSION}, Expr=0, Scope=FRAME_SCOPE, Processed=
false}
     [java] ./uast/sharedframes/imported_frame_no_widget_embedded_assign.p
     [java] frame_scoping: (process frame phrase) get_frame=
     [java] {IsRootScope=false, New=false, Disp=false, Down=-1, Ref=[FRAME_PHRASE], Wid={expr1=EXPRESSION}, De
f=false, Expr=0, Index={}, Shared=false, Name=sf3, Processed=false}
     [java] frame_scoping.process_fake_scope:
     [java]  [FRAME_SCOPE]:17179869291 @0:0
     [java] 
     [java] ## INFO: ./uast/sharedframes/imported_frame_no_widget_embedded_assign.p: assuming DOWN set to 1 fo
r sf3
     [java] frame_scoping: (process frame phrase) get_frame=
     [java] {IsRootScope=false, New=false, Disp=false, Down=-1, Ref=[FRAME_PHRASE], Wid={}, Def=false, Expr=0,
 Index={}, Shared=false, Name=, Processed=false}
     [java] frame_scoping.process_fake_scope:
     [java]  [FRAME_SCOPE]:17179869292 @0:0
     [java] 
     [java] frame_scoping (in loop)!: get_named_frame=
     [java] {IsRootScope=true, New=false, Disp=true, Down=1, Def=true, Index={}, Shared=true, Name=sf3, Ref=[D
EFINE_FRAME, FRAME_SCOPE, FRAME_PHRASE, KW_FORM], Wid={expr1=EXPRESSION}, Expr=0, Scope=FRAME_SCOPE, Processed
=false}
```

Still analyzing.

#19 - 01/10/2020 06:16 PM - Roger Borrello

Roger Borrello wrote:

> In annotations/array_expansion.rules I updated the below to protect the setting of attachIdx and parentRef:
> [...]
>
> Now we are getting the same code conversion error as the scalar testcases (against trunk) where there is a missing frame-id. Probably someone looking for an annotation we don't have anymore, without error checking.

Added additional check for prog.kw_set:

```
               <!-- We need to prevent expansion if we are going to do an embedded assignment -->
               <rule>!((parent.type == prog.kw_update or
                        parent.type == prog.kw_set)
                        and copy.nextSibling.type == prog.assign)
                  <action>attachIdx = copy.indexPos</action>
                  <action>parentRef = copy.parent</action>
               </rule>
```

Do we need to validate that there isn't already an LBRACKET child? Perhaps there's display myx[1] = 1?

#20 - 01/11/2020 08:37 AM - Greg Shah

> Do we need to validate that there isn't already an LBRACKET child? Perhaps there's display myx[1] = 1?

We are only in this location if we already know it is an unsubscripted array reference (or a range array reference). See line 233.

#21 - 01/13/2020 08:52 AM - Roger Borrello

By protecting the evalLib("process_fake_scope", tmp, true) in **annotations/frame_scoping.rules** with a check for this.parent.type != prog.kw_assign, the empty FRAME_SCOPE is not longer being created. The Java emitted for the scalar is now a simple sf2sFrame.assign(1) and for the extent is assignMulti(myx3, 1);

There is still something to investigate, as the testcases that are trying to add a widget to the imported shared frame (extent and scalar) without an embedded assignment are failing. They are trying the add myx to the frame as a widget, but they are not in the frame definition. The standalone and non-shared versions don't have an issue (even though the converted code won't run properly) generating code that will compile.

#22 - 01/13/2020 10:41 AM - Roger Borrello

Roger Borrello wrote:

> By protecting the evalLib("process_fake_scope", tmp, true) in **annotations/frame_scoping.rules** with a check for this.parent.type != prog.kw_assign, the empty FRAME_SCOPE is not longer being created. The Java emitted for the scalar is now a simple sf2sFrame.assign(1) and for the extent is assignMulti(myx3, 1);
>
> There is still something to investigate, as the testcases that are trying to add a widget to the imported shared frame (extent and scalar) without an embedded assignment are failing. They are trying the add myx to the frame as a widget, but they are not in the frame definition. The standalone and non-shared versions don't have an issue (even though the converted code won't run properly) generating code that will compile.

Investigation closed, as we don't need to create working java code for something that can't compile in 4GL.

#23 - 01/13/2020 10:54 AM - Roger Borrello

There is a scenario which is still broken, and that is when there is a widget that **should be added** to a non-shared frame such as in this case:

```
def frame f7.
def var fieldx7 as char no-undo init "fxval".
def var myx7 as int extent 2 init -1.
def var i as int init -1.

form fieldx7 with frame f7.

/*
 * i is added as a widget, and the frame is displayed.
 * When done hits, i is what you entered, and myx[1] and myx[2] are 1.
 */

update i myx7 = 1 with frame f7.

message "Done." i myx7[1] myx7[2].
```

We are still getting the embedded assignment as a frameElement, along with the new widget.

```
         FrameElement[] elementList0 = new FrameElement[]
         {
            new Element(i, f7sFrame.widgeti()),
            new EmbeddedAssignment(myx7s, 1)
         };
```

#24 - 01/13/2020 11:30 AM - Roger Borrello

Roger Borrello wrote:

> There is a scenario which is still broken, and that is when there is a widget that **should be added** to a non-shared frame such as in this case:
>
> [...]
>
> We are still getting the embedded assignment as a frameElement, along with the new widget.
>
> [...]

The CONTENT_ARRAY contains the FRAME_ELEMENT and EMBEDDED_ASSIGNMENT as siblings under it. The FRAME_ELEMENT only has the VAR_INT (2 copies, one with the *frame-id*, one without). The EMBEDDED_ASSIGNMENT does not have a *frame-id*, and it's children are VAR_INT and EXPRESSION/NUM_LITERAL.

Should the EMBEDDED_ASSIGNMENT just be moved to be a peer of CONTENT_ARRAY under the enclosing STATEMENT?

#25 - 01/13/2020 12:13 PM - Greg Shah

This is not a bug. This is just the normal embedded assignment case. Since i is still an editable widget, this is still a valid UPDATE statement and should not be converted to ASSIGN. Thus, the EmbeddedAssignment node must be passed in as part of the array.

#26 - 01/13/2020 12:56 PM - Roger Borrello

Greg Shah wrote:

> This is not a bug. This is just the normal embedded assignment case. Since i is still an editable widget, this is still a valid UPDATE statement and should not be converted to ASSIGN. Thus, the EmbeddedAssignment node must be passed in as part of the array.

Well, we are getting this error:

```
   [javac] testcases/src/com/goldencode/testcases/sharedframes/NonsharedFrameExtraWidgetEmbeddedAssign.java:36
: error: no suitable constructor found for EmbeddedAssignment(integer[],int)
    [javac]             new EmbeddedAssignment(myx7, 1)
    [javac]             ^
    [javac]     constructor EmbeddedAssignment.EmbeddedAssignment(BaseDataType,int) is not applicable
    [javac]       (argument mismatch; integer[] cannot be converted to BaseDataType)
    [javac]     constructor EmbeddedAssignment.EmbeddedAssignment(Accessor,int) is not applicable
    [javac]       (argument mismatch; integer[] cannot be converted to Accessor)
    [javac]     constructor EmbeddedAssignment.EmbeddedAssignment(BaseDataType,double) is not applicable
    [javac]       (argument mismatch; integer[] cannot be converted to BaseDataType)
    [javac]     constructor EmbeddedAssignment.EmbeddedAssignment(Accessor,double) is not applicable
    [javac]       (argument mismatch; integer[] cannot be converted to Accessor)
    [javac]     constructor EmbeddedAssignment.EmbeddedAssignment(BaseDataType,boolean) is not applicable
    [javac]       (argument mismatch; integer[] cannot be converted to BaseDataType)
    [javac]     constructor EmbeddedAssignment.EmbeddedAssignment(Accessor,boolean) is not applicable
    [javac]       (argument mismatch; integer[] cannot be converted to Accessor)
    [javac]     constructor EmbeddedAssignment.EmbeddedAssignment(BaseDataType,String) is not applicable
    [javac]       (argument mismatch; integer[] cannot be converted to BaseDataType)
    [javac]     constructor EmbeddedAssignment.EmbeddedAssignment(Accessor,String) is not applicable
    [javac]       (argument mismatch; integer[] cannot be converted to Accessor)
    [javac]     constructor EmbeddedAssignment.EmbeddedAssignment(BaseDataType,BaseDataType) is not applicable
    [javac]       (argument mismatch; integer[] cannot be converted to BaseDataType)
    [javac]     constructor EmbeddedAssignment.EmbeddedAssignment(BaseDataType,Resolvable) is not applicable
    [javac]       (argument mismatch; integer[] cannot be converted to BaseDataType)
    [javac]     constructor EmbeddedAssignment.EmbeddedAssignment(BaseDataType,Supplier<BaseDataType>) is not 
applicable
    [javac]       (argument mismatch; integer[] cannot be converted to BaseDataType)
    [javac]     constructor EmbeddedAssignment.EmbeddedAssignment(Accessor,BaseDataType) is not applicable
    [javac]       (argument mismatch; integer[] cannot be converted to Accessor)
    [javac]     constructor EmbeddedAssignment.EmbeddedAssignment(Accessor,Resolvable) is not applicable
    [javac]       (argument mismatch; integer[] cannot be converted to Accessor)
    [javac]     constructor EmbeddedAssignment.EmbeddedAssignment(Accessor,Supplier<BaseDataType>) is not appl
icable
    [javac]       (argument mismatch; integer[] cannot be converted to Accessor)
    [javac] 1 error
```

The extent java code is below:

```
      character fieldx7 = TypeFactory.character("fxval");
      integer[] myx7 = UndoableFactory.integerExtent(2, (long) -1);
      integer i = UndoableFactory.integer((long) -1);

      externalProcedure(NonsharedFrameExtraWidgetEmbeddedAssign.this, new Block((Body) () -> 
      {
         f7Frame.openScope();

         FrameElement[] elementList0 = new FrameElement[]
         {
            new Element(i, f7Frame.widgeti()),
            new EmbeddedAssignment(myx7, 1)
         };
```

And the scalar is here:

```
      character fieldx7s = TypeFactory.character("fxval");
      integer myx7s = UndoableFactory.integer((long) -1);
      integer i = UndoableFactory.integer((long) -1);

      externalProcedure(NonsharedFrameExtraWidgetEmbeddedAssignScalar.this, new Block((Body) () -> 
      {
         f7sFrame.openScope();

         FrameElement[] elementList0 = new FrameElement[]
         {
            new Element(i, f7sFrame.widgeti()),
            new EmbeddedAssignment(myx7s, 1)
         };
```

#27 - 01/13/2020 02:16 PM - Roger Borrello

Built new testcases, which uses fields in a temp table.

nonshared_frame_extra_widget_embedded_assign_field.p

```
def temp-table tt field myx7 as int extent 2.
create tt. tt.myx7 = -1.
def frame f7.
def var fieldx7 as char no-undo init "fxval".
def var i as int init -1.

form fieldx7 with frame f7.

/*
 * i is added as a widget, and the frame is displayed.
 * When done hits, i is what you entered, and myx[1] and myx[2] are 1.
 */

update i tt.myx7 = 1 with frame f7.

message "Done." i tt.myx7[1] tt.myx7[2].
```

nonshared_frame_extra_widget_embedded_assign_field-scalar.p

```
def temp-table tt field myx7 as int.
create tt. tt.myx7 = -1.
def frame f7.
def var fieldx7 as char no-undo init "fxval".
def var i as int init -1.

form fieldx7 with frame f7.

/*
 * i is added as a widget, and the frame is displayed.
 * When done hits, i is what you entered, and myx[1] and myx[2] are 1.
 */

update i tt.myx7 = 1 with frame f7.

message "Done." i tt.myx7.
```

Errors (edited down the list of unmatched constructors):

```
    [javac] /home/rfb/projects/VirtualBox-VMs/shared/projects/testcases/src/com/goldencode/testcases/dmo/_temp
/impl/Tt_1_1Impl.java:12: error: Tt_1_1Impl is not abstract and does not override abstract method setMyx7(int,
NumberType) in Tt_1
    [javac] public class Tt_1_1Impl
    [javac]        ^
    [javac] /home/rfb/projects/VirtualBox-VMs/shared/projects/testcases/src/com/goldencode/testcases/sharedfra
mes/NonsharedFrameExtraWidgetEmbeddedAssignField.java:42: error: no suitable constructor found for EmbeddedAss
ignment(integer[],int)
    [javac]              new EmbeddedAssignment(tt.getMyx7(), 1)
    [javac]              ^
...
    [javac] testcases/sharedframes/NonsharedFrameExtraWidgetEmbeddedAssignFieldScalar.java:42: error: no suita
ble constructor found for EmbeddedAssignment(integer[],int)
    [javac]              new EmbeddedAssignment(tt.getMyx7(), 1)
    [javac]              ^
...
    [javac] 3 errors
```

#28 - 01/13/2020 02:42 PM - Greg Shah

Did you test this in the 4GL?

#29 - 01/13/2020 02:49 PM - Roger Borrello

Greg Shah wrote:

> Did you test this in the 4GL?

Yes... fields worked just like variables.

#30 - 01/13/2020 02:52 PM - Constantin Asofiei

I think we need to change the constructor for EmbeddedAssignment to receive a lambda expression, which will actually perform the assignment. And not the rvalue and lvalue.

#31 - 01/14/2020 09:17 AM - Roger Borrello

Constantin Asofiei wrote:

> I think we need to change the constructor for EmbeddedAssignment to receive a lambda expression, which will actually perform the assignment. And not the rvalue and lvalue.

Had a discussion with Greg, and we believe the best course of action would be:

1. Create AccessorArrayWrapper (from AccessorWrapper)
2. Add constructors to EmbeddedAssignment for BastDataType []
3. Change conversion to emit bulk parameter as **true** for unsubscripted field references

The last will get us past conversion, but runtime will still need some work.

Any thoughts or comments?

#32 - 01/14/2020 09:23 AM - Greg Shah

> The last will get us past conversion, but runtime will still need some work.

Not exactly. The last will get us past conversion AND the runtime should already be working.

The first two are needed to allow the current conversion to compile and run. The AccessorArrayWrapper will call ArrayAssigner.assignMulti(wrappee, <value>) inside of set() and the get() will raise an exception since it is never needed (and can't work).

#33 - 01/14/2020 09:26 AM - Constantin Asofiei

Hmm... isn't tt.getMyx7() returning a copy of the array?

#34 - 01/14/2020 09:49 AM - Greg Shah

Constantin Asofiei wrote:

> Hmm... isn't tt.getMyx7() returning a copy of the array?

Yes, I didn't look at that carefully. We will have to tweak conversion to emit a FieldReference which is missing today.

#35 - 01/14/2020 10:54 AM - Roger Borrello

Updates for un-subscripted variables in an embedded assignment are committed in 4207a-11376:

```
modified src/com/goldencode/p2j/ui/EmbeddedAssignment.java
added src/com/goldencode/p2j/util/AccessorArrayWrapper.java
```

#36 - 01/14/2020 12:25 PM - Roger Borrello

With respect to updates to convert/database_references.rules for emitting a field reference...

Wouldn't converting:

```
               <rule>upPath("EMBEDDED_ASSIGNMENT/EXPRESSION")
                  <action>accessor = true</action>
               </rule>
```

to:

```
               <rule>upPath("EMBEDDED_ASSIGNMENT/EXPRESSION") or
                     parent.type == prog.embedded_assignment
                  <action>accessor = true</action>
               </rule>
```

be the exact same thing as:

```
               <rule>parent.type == prog.embedded_assignment
                  <action>accessor = true</action>
               </rule>
```

?

That would really mean no matter what follows the EMBEDDED_ASSIGNMENT, we need to deal with a field reference, right?

In any case, we do generate the correct code:

```
         FrameElement[] elementList0 = new FrameElement[]
         {
            new Element(i, f7Frame.widgeti()),
            new EmbeddedAssignment(new FieldReference(tt, "myx7"), 1)
         };
```

#37 - 01/14/2020 01:11 PM - Greg Shah

No. Read CommonAstSupport for upPath() to understand why.

#38 - 01/14/2020 04:01 PM - Roger Borrello

Committed rules/convert/database_references.rules in **4207a-11377** to use field reference for the lvalue of an embedded assignment.

#39 - 01/14/2020 04:02 PM - Roger Borrello

- % Done changed from 0 to 100

- Status changed from New to WIP

#40 - 01/15/2020 10:04 AM - Roger Borrello

There is a regression found where this code line:

```
update i tt.myx7s = caps(tt.myx7s) with frame f7s.
```

emits this difference (top is trunk, bottom is post @database_references.rules @update):

```
<              new EmbeddedAssignment(tt.getMyx7s(), () -> toUpperCase(tt.getMyx7s()))
---
>              new EmbeddedAssignment(new FieldReference(tt, "myx7s"), () -> toUpperCase(tt.getMyx7s()))
```

#41 - 01/15/2020 10:04 AM - Roger Borrello

- % Done changed from 100 to 90

#42 - 01/15/2020 10:16 AM - Greg Shah

That is not a regression. It is more correct. We WANT the FieldReference to be emitted.

#43 - 01/15/2020 10:55 AM - Roger Borrello

- % Done changed from 90 to 100

#45 - 01/15/2020 10:57 AM - Roger Borrello

Greg Shah wrote:

> That is not a regression. It is more correct. We WANT the FieldReference to be emitted.

Thank you for clarifying!

#46 - 01/15/2020 02:04 PM - Roger Borrello

Greg Shah wrote:

> No. Read CommonAstSupport for upPath() to understand why.

I don't see that method in https://proj.goldencode.com/artifacts/javadoc/latest/api/com/goldencode/p2j/pattern/CommonAstSupport.html

Is that the right place?

#47 - 01/15/2020 02:48 PM - Eric Faulhaber

com.goldencode.p2j.pattern.CommonAstSupport$Library.upPath(String)

That's just a convenience function for TRPL. The implementation is in AnnotatedAst.upPath(String).

#48 - 01/16/2020 11:29 AM - Roger Borrello

A question to our database "gurus" with respect to how the "bulk" option is used in FieldReference.java...

Why cannot the FieldReference constructors determine the need for bulk via the fact that there is an extent? It has access to the property right there in the method.

#49 - 01/16/2020 11:56 AM - Greg Shah

Roger Borrello wrote:

> A question to our database "gurus" with respect to how the "bulk" option is used in FieldReference.java...
>
> Why cannot the FieldReference constructors determine the need for bulk via the fact that there is an extent? It has access to the property right there in the method.

It would also have to take into account that there was no index specified. If the field is an extent field and there is no index provided, then wouldn't any assignment be a bulk assign?

#50 - 01/16/2020 03:32 PM - Roger Borrello

FieldReference has two very similar constructors:

- FieldReference(DataModelObject dmo, String property, int index)

- FieldReference(DataModelObject dmo, String property, NumberType index)

The first calls the worker constructor with:
this(dmo, property, false, new integer(index), false);
the second with:
this(dmo, property, false, index, false);

The worker constructor has this signature:
FieldReference(DataModelObject dmo, String property, boolean uppercase, NumberType index, boolean bulk)

Can someone tell me why both are needed? I'm not looking to eliminate one of them, I'm creating another helper for allowing convert/database_references.rules to utilize a new constructor, and wanted to know if one or the other index types is better. In the new case, we'll have to flip the order of the 3rd and 4th parameters, since there is already a constructor with this signature:
FieldReference(DataModelObject dmo, String property, boolean uppercase, int index) (Again, why int versus NumberType for the *index*?)

#51 - 01/16/2020 04:06 PM - Roger Borrello

Greg Shah wrote:

> Roger Borrello wrote:
>
> > A question to our database "gurus" with respect to how the "bulk" option is used in FieldReference.java...
> >
> > Why cannot the FieldReference constructors determine the need for bulk via the fact that there is an extent? It has access to the property right there in the method.
>
> It would also have to take into account that there was no index specified. If the field is an extent field and there is no index provided, then wouldn't any assignment be a bulk assign?

I agree with that logic. There's a lot of checking done in convert/database_references.rules to deal with input and input-output. To allow FieldReference to handle this would be a simpler implementation.

#52 - 01/16/2020 04:29 PM - Greg Shah

> why int versus NumberType for the index?

Because sometimes the 4GL code has a subscript which is a NUM_LITERAL (a constant that will emit as an int in Java) and other times there is an expression of type integer, int64 or decimal (which are all subclasses of NumberType).

We do this (make variants of APIs) in many places in the runtime for the same reason.

#53 - 01/16/2020 05:20 PM - Roger Borrello

Roger Borrello wrote:

> Greg Shah wrote:
>
>> Roger Borrello wrote:
>>
>>> A question to our database "gurus" with respect to how the "bulk" option is used in FieldReference.java...
>>>
>>> Why cannot the FieldReference constructors determine the need for bulk via the fact that there is an extent? It has access to the property right there in the method.
>>
>> It would also have to take into account that there was no index specified. If the field is an extent field and there is no index provided, then wouldn't any assignment be a bulk assign?
>
> I agree with that logic. There's a lot of checking done in convert/database_references.rules to deal with input and input-output. To allow FieldReference to handle this would be a simpler implementation.

<u>Options</u>
Change FieldReference.java worker constructor from current

```
bulk = bulk && (extent != null) && (index == null);
```

to

```
bulk = bulk ? true : (extent != null) && (index == null);
```

Or change:

- Change FieldReference.java to add a helper constructor with bogus 3rd parameter (int) and 4th parameter (boolean) which would call the worker constructor. Does it matter what is passed for uppercase?
- Change convert/database_references.rules to
  - Include 2 flags (is_extent and lval_of_embedded_assignment)
    - is_extent = getNoteLong("extent") > 0
    - lval_of_embedded_assignment = (parent.type == prog.embedded_assignment) when we are checking for EMBEDDED_ASSIGNMENT/EXPRESSION
  - Check for both flags to be true, and add 2 parameters:

    ```
                      <action>createJavaAst(java.num_literal, "0", lastid)</action>
                      <action>createJavaAst(java.bool_true, "", lastid)</action>
    ```

I have the latter coded, which allows:

```
update i tt.myx7 = 1 with frame f7.
```

to emit:

```
            new EmbeddedAssignment(new FieldReference(tt, "myx7", 0, true), 1)
```

In either case, I'd need to learn how to get runtime testing going.

#54 - 01/17/2020 08:59 AM - Greg Shah

> Does it matter what is passed for uppercase?

This would not be used to match your new constructors. But it might be automatically generated which would naturally cause the 5 parm constructor to be used. So you don't need to worry about this.

> <action>createJavaAst(java.num_literal, "0", lastid)</action>

I prefer using "-1" instead of "0". In Java 0 is a valid subscript so it is confusing to use "0" as meaning invalid subscript.

#55 - 01/17/2020 10:30 AM - Roger Borrello

Greg Shah wrote:

> > Does it matter what is passed for uppercase?

This would not be used to match your new constructors. But it might be automatically generated which would naturally cause the 5 parm constructor to be used. So you don't need to worry about this.

I am not using it in the new constructor, but I have to call the 5 parm constructor with something in position 3... here's what I have:

```
this(dmo, property, false, null, bulk);
```

null would be fine for *index*, since this doesn't come into play. But for *uppercase*, are there implications of setting it one way or the other when the FieldReference is used?

```
<action>createJavaAst(java.num_literal, "0", lastid)</action>
```

I prefer using "-1" instead of "0". In Java 0 is a valid subscript so it is confusing to use "0" as meaning invalid subscript.

OK. But it still gives me two solutions. I could implement both.

#56 - 01/17/2020 12:18 PM - Roger Borrello

Looking into what the ramifications of passing uppercase as **false** when constructing the FieldReference for bulk assignment, I see that the flag is used "to force resolved character result to uppercase/right-trim".

I searched for "uppercase" in other classes, and it is used by BufferFieldImpl.java to pass in when instantiating FieldReference, so that was a dead-end.

Since the value of uppercase defaults to **false**, I believe it should be save to pass in **false** when we are creating a FieldReference in this case.

#57 - 01/19/2020 03:41 PM - Roger Borrello

Code ready for review:
modified **rules/convert/database_references.rules**
modified **src/com/goldencode/p2j/persist/FieldReference.java**
Committed revision **11381**.

#58 - 01/20/2020 02:20 PM - Greg Shah

Code Review Task Branch 4207a Revision 11381

1. In database_references.rules, I don't see any protection for the case where there is an lbracket child of the field_* node. The is_extent does not protect against that case.

2. In FieldReference, the new constructor should not always pass null for the index parameter. What if someone calls with a 0 or positive value? You have to assume that others will eventually use this constructor and protect against that case. Only a -1 (or maybe just anything negative) index should pass null.

3. In FieldReference, the javadoc is incorrect. It should explain what happens when you pass a negative value for index. Also the text "An indexed getter method will be used to resolve the field." is definitely incorrect.

#59 - 01/20/2020 03:04 PM - Roger Borrello

Greg Shah wrote:

> Code Review Task Branch 4207a Revision 11381
>
> 1. In database_references.rules, I don't see any protection for the case where there is an lbracket child of the field_* node. The is_extent does not protect against that case.

Thanks for the review!

I am adding and !downPath("LBRACKET") to the condition under which the new parameter set is used:

```
               <rule>is_extent and lval_of_embedded_assignment and !downPath("LBRACKET")
                  <action>createJavaAst(java.num_literal, "-1", lastid)</action>
                  <action>createJavaAst(java.bool_true, "", lastid)</action>
               </rule>
```

Before/after of update i tt.myx7[1] = 1 with frame f7.:

```
<             new EmbeddedAssignment(new FieldReference(tt, "myx7", -1, true, 0), 1)
---
>             new EmbeddedAssignment(new FieldReference(tt, "myx7", 0), 1)
```

> 2. In FieldReference, the new constructor should not always pass null for the index parameter. What if someone calls with a 0 or positive value? You have to assume that others will eventually use this constructor and protect against that case. Only a -1 (or maybe just anything negative) index should pass null.

Updating with:

```
   {
      this(dmo, property, false, index < 0 ? null : index, bulk);
   }
```

3. In FieldReference, the javadoc is incorrect. It should explain what happens when you pass a negative value for index. Also the text "An indexed getter method will be used to resolve the field." is definitely incorrect.

There is also no mention of the fact that uppercase is not supported with this constructor. How does this read?

```
   /**
    * Create a new field reference given a DMO proxy, a property name, an index for the
    * property, and a flag indicating whether or not to use bulk. Allows for explicit use
    * of {@code bulk}. This constructor does not support forcing a {@code character} value
    * returned by {@link #getObject} to be uppercased/right-trimmed;
    *
    * @param   dmo
    *          DMO proxy.
    * @param   property
    *          Target property name.
    * @param   index
    *          index which, when negative, will be passed as null to constructor.
    * @param   bulk
    *          When true the bulk getter/setter will be used.
    *
    * @throws  IllegalArgumentException
    *          if field reference is not recognized.
    */
```

Why am I not calling the main constructor directly with:

```
               <rule>is_extent and lval_of_embedded_assignment and !downPath("LBRACKET")
                  <action>createJavaAst(java.bool_false, "", lastid)</action>
                  <action>createJavaAst(java.num_literal, "-1", lastid)</action>
                  <action>createJavaAst(java.bool_true, "", lastid)</action>
               </rule>
```

#60 - 01/20/2020 04:31 PM - Greg Shah

The changes are good.

> Why am I not calling the main constructor directly with:

Because then you would create more code for every customer and there is no advantage. In fact it is only a disadvantage to add necessary parameters which customers will have to maintain forever.

#61 - 02/21/2020 04:25 PM - Roger Borrello

Should this be moved to Status= **Test**

#62 - 02/21/2020 04:49 PM - Greg Shah

- Status changed from WIP to Test

#63 - 03/03/2020 02:56 PM - Roger Borrello

Task branch 4207a was merged to trunk as revision 11344.

#64 - 03/04/2020 10:32 AM - Greg Shah

- Status changed from Test to Closed