User Interface - Bug #4488

Wrong frame used after display includes widgets from other frame

01/02/2020 03:34 PM - Roger Borrello

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

Description

History

#1 - 01/02/2020 04:17 PM - Roger Borrello

Testcase

Updates are in 4207a-11372.

The below has been checked in as uast/sharedframes/shared_frame_mixup.p:

```
def shared frame sf1.
def shared frame sf2.
def shared var fieldx as char no-undo.
def shared var mychar as char no-undo.

form fieldx with frame sf1.
form mychar with frame sf2.

loopy:sf2:
   display mychar @ fieldx with frame sf1.
   if input mychar = "" then do:
      message "Done.".
      leave loopy.
   end.
end.
```

There is a corresponding run procedure uast/sharedframes/shared_frame_mixup-run.p:

```
def new shared frame sf1.
def new shared frame sf2.
def new shared var fieldx as char no-undo.
def new shared var mychar as char no-undo.

form fieldx with frame sf1.
form mychar with frame sf2.

run "~testcases/uast/sharedframes/shared_frame_var.p".

message "Done.".
```

It is not critical that frame sf2 is shared. It can be local to the procedure, and will still fail.

Failure

This results in:

```
    [javac] ~testcases/src/com/goldencode/testcases/sharedframes/SharedFrameMixup.java:49: error: cannot find
symbol
    [javac]             if (_isEqual(sf1Frame.getMychar(), ""))
    [javac]                                  ^
    [javac]   symbol:   method getMychar()
    [javac]   location: variable sf1Frame of type FrameSf1_1
    [javac] 1 error
```

Generated Code

The Java generated:

Java	Description
sharedframes/SharedFrameMixup.java	Business logic for SharedFrameMixup class
sharedframes/SharedFrameMixupRun.java	Business logic for SharedFrameMixupRun class
ui/shared_frames/FrameSf1_1.java	interface FrameSf1_1 for mychar
ui/shared_frames/FrameSf2_1.java	interface FrameSf2_1 for fieldx
ui/sharedframes/SharedFrameMixupRunSf1.java	interface SharedFrameMixupRunSf1 extends FrameSf1_1
ui/sharedframes/SharedFrameMixupRunSf2.java	interface SharedFrameMixupRunSf2 extends FrameSf2_1

SharedFrameMixup

This code is below:

```
public class SharedFrameMixup
{
   FrameSf2_1 sf2Frame = GenericFrame.importSharedFrame(FrameSf2_1.class, "sf2");

   FrameSf1_1 sf1Frame = GenericFrame.importSharedFrame(FrameSf1_1.class, "sf1");

   character fieldx = SharedVariableManager.lookupVariable("fieldx", character.class, true);

   character mychar = SharedVariableManager.lookupVariable("mychar", character.class, true);

   /**
    * External procedure (converted to Java from the 4GL source code
    * in sharedframes/shared_frame_mixup.p).
    */
   public void execute()
   {
      externalProcedure(SharedFrameMixup.this, new Block((Body) () ->
      {
         repeat("loopy", new Block((Body) () ->
         {
            FrameElement[] elementList0 = new FrameElement[]
            {
               new Element(mychar, sf1Frame.widgetFieldx())
            };

            sf1Frame.display(elementList0);

            if (_isEqual(sf1Frame.getMychar(), ""))
            {
               message("Done.");
               leave("loopy");
            }
         }));
      }));
   }
}
```

The wrong class is being used for getMychar() method.

#2 - 01/02/2020 04:18 PM - Roger Borrello

Roger Borrello wrote:

> ...

Hold on... there's a typo in the testcase I need to correct.

#3 - 01/02/2020 04:31 PM - Roger Borrello

Roger Borrello wrote:

> Roger Borrello wrote:
>
> > ...
>
> Hold on... there's a typo in the testcase I need to correct.

Corrected #note-1 text.

#5 - 01/02/2020 05:56 PM - Roger Borrello

When the display mychar @ fieldx with frame sf1 is used with a non-shared sf1, the code SharedFrameMixupSf1 class contains both getFieldx() and getMychar() methods, and the code is good. Only when sf1 is shared (regardless of sf2 being shared or not) do we get crossed-up.

I added some printfln() to annotations/frame_scoping.rules and the only place the testcase hits a check for prog.at is in this walk-rule when processing the fieldx on line 12:

```
      <!-- the variables used in @ base clauses in format_phrase are also part of the frame -->
      <rule>inFormat and parent.type == prog.at
         <action>evalLib("add_widget")</action>
         <action>printfln("frame_scoping:\n%s", this.dumpTree())</action>
      </rule>
```

#6 - 01/03/2020 11:23 AM - Roger Borrello

I am struggling to understand what exactly I should expect that code to do:
display mychar @ fieldx with frame sf1.

Why would you do such a thing right after you start a repeat with frame sf2.? The code in question has over 800 I tried to do something simpler in the testcase:
display mychar with frame sf1. but the 4GL came to its senses, and reported:

```
** Shared frame sf1 field mychar must first appear in a FORM statement. (890)
**  Could not understand line 12. (196)
```

The DEFINE FRAME documentation gives the reason why...

```
All frame fields and Frame phrase options in a shared frame must first be defined in the initial DEFINE NEW SH
ARED FRAME statement or an additional FORM statement in the same procedure. Procedures that share this frame o
nly have to define fields that correspond to the fields in the initial definition plus any specified ACCUM opt
ion. Other Frame phrase options for the SHARED frames are allowed, but are ignored except for the ACCUM option
. This allows you to make use of the same FORM statement in an include file for both the NEW SHARED and matchi
ng SHARED frames. See the FORM statement reference entry for more information.
```

Our AST shows:
BLOCK/STATEMENT/CONTENT_ARRAY/FRAME_ELEMENT/EXPRESSION/VAR_CHAR and the VAR_CHAR is mychar. Then there's a peer to EXPRESSION child of BLOCK/STATEMENT/CONTENT_ARRAY/FRAME_ELEMENT/VAR_CHAR which is for fieldx.

BLOCK/STATEMENT/CONTENT_ARRAY/KW_DISP/EXPRESSION/VAR_CHAR for mychar
BLOCK/STATEMENT/CONTENT_ARRAY/KW_DISP/FORMAT_PHRASE/AT/VAR_CHAR for fieldx.
BLOCK/STATEMENT/CONTENT_ARRAY/KW_DISP/FRAME_PHRASE/KW_FRAME/WID_FRAME for sf1.

#7 - 01/03/2020 11:25 AM - Roger Borrello

Roger Borrello wrote:

> I am struggling to understand what exactly I should expect that code to do:

display mychar @ fieldx with frame sf1.

Why would you do such a thing right after you start a repeat with frame sf2.? The code in question has over 800 I tried to do something simpler in the testcase:
display mychar with frame sf1. but the 4GL came to its senses, and reported:
[...]

The DEFINE FRAME documentation gives the reason why...
[...]

Our AST shows:
BLOCK/STATEMENT/CONTENT_ARRAY/FRAME_ELEMENT/EXPRESSION/VAR_CHAR and the VAR_CHAR is mychar. Then there's a peer to EXPRESSION child of BLOCK/STATEMENT/CONTENT_ARRAY/FRAME_ELEMENT/VAR_CHAR which is for fieldx.

BLOCK/STATEMENT/CONTENT_ARRAY/KW_DISP/EXPRESSION/VAR_CHAR for mychar
BLOCK/STATEMENT/CONTENT_ARRAY/KW_DISP/FORMAT_PHRASE/AT/VAR_CHAR for fieldx.
BLOCK/STATEMENT/CONTENT_ARRAY/KW_DISP/FRAME_PHRASE/KW_FRAME/WID_FRAME for sf1.

```
    <ast col="0" id="17179869281" line="0" text="block" type="BLOCK">
      <annotation datatype="java.lang.Boolean" key="recordscoping" value="true"/>
      <ast col="0" id="17179869282" line="0" text="statement" type="STATEMENT">
        <ast col="0" id="17179869345" line="0" text="" type="CONTENT_ARRAY">
          <annotation datatype="java.lang.String" key="javaname" value="elementList0"/>
          <annotation datatype="java.lang.Long" key="peerid" value="180388626530"/>
          <ast col="0" id="17179869346" line="0" text="" type="FRAME_ELEMENT">
            <annotation datatype="java.lang.Long" key="peerid" value="180388626531"/>
            <ast col="0" id="17179869347" line="0" text="expression" type="EXPRESSION">
              <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
              <annotation datatype="java.lang.String" key="javaname" value="fieldx"/>
              <annotation datatype="java.lang.String" key="setter" value="setFieldx"/>
              <annotation datatype="java.lang.String" key="getter" value="getFieldx"/>
              <annotation datatype="java.lang.String" key="widgettype" value="FillInWidget"/>
              <annotation datatype="java.lang.String" key="accessor" value="widgetFieldx"/>
              <annotation datatype="java.lang.Boolean" key="next2lastref" value="true"/>
              <annotation datatype="java.lang.Long" key="peerid" value="180388626532"/>
              <ast col="12" id="17179869348" line="12" text="mychar" type="VAR_CHAR">
                <annotation datatype="java.lang.Long" key="oldtype" value="2782"/>
                <annotation datatype="java.lang.Long" key="refid" value="17179869223"/>
                <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
                <annotation datatype="java.lang.Long" key="peerid" value="180388626533"/>
              </ast>
            </ast>
            <ast col="21" id="17179869349" line="12" text="fieldx" type="VAR_CHAR">
              <annotation datatype="java.lang.Long" key="oldtype" value="2782"/>
              <annotation datatype="java.lang.Long" key="refid" value="17179869207"/>
              <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
              <annotation datatype="java.lang.String" key="getter" value="getFieldx"/>
              <annotation datatype="java.lang.String" key="setter" value="setFieldx"/>
              <annotation datatype="java.lang.String" key="widgettype" value="FillInWidget"/>
              <annotation datatype="java.lang.String" key="javaname" value="fieldx"/>
              <annotation datatype="java.lang.String" key="accessor" value="widgetFieldx"/>
              <annotation datatype="java.lang.Boolean" key="lastref" value="true"/>
              <annotation datatype="java.lang.Long" key="frame-id" value="17179869343"/>
              <annotation datatype="java.lang.Long" key="peerid" value="180388626534"/>
            </ast>
          </ast>
        </ast>
        <ast col="4" id="17179869283" line="12" text="display" type="KW_DISP">
          <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
          <annotation datatype="java.lang.Long" key="scope-id" value="17179869342"/>
          <annotation datatype="java.lang.Long" key="frame-id" value="17179869343"/>
          <annotation datatype="java.lang.Long" key="block_par_id" value="180388626519"/>
          <annotation datatype="java.lang.Long" key="peerid" value="180388626537"/>
          <ast col="0" hidden="true" id="17179869285" line="0" text="expression" type="EXPRESSION">
            <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
            <annotation datatype="java.lang.String" key="javaname" value="fieldx"/>
            <annotation datatype="java.lang.String" key="setter" value="setFieldx"/>
            <annotation datatype="java.lang.String" key="getter" value="getFieldx"/>
            <annotation datatype="java.lang.String" key="widgettype" value="FillInWidget"/>
            <annotation datatype="java.lang.String" key="accessor" value="widgetFieldx"/>
            <annotation datatype="java.lang.Boolean" key="next2lastref" value="true"/>
            <annotation datatype="java.lang.Long" key="frame-id" value="17179869343"/>
```

```
            <ast col="12" id="17179869286" line="12" text="mychar" type="VAR_CHAR">
              <annotation datatype="java.lang.Long" key="oldtype" value="2782"/>
              <annotation datatype="java.lang.Long" key="refid" value="17179869223"/>
              <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
              <annotation datatype="java.lang.Long" key="frame-id" value="17179869343"/>
            </ast>
          </ast>
          <ast col="0" hidden="true" id="17179869288" line="0" text="format phrase" type="FORMAT_PHRASE">
            <annotation datatype="java.lang.Long" key="frame-id" value="17179869343"/>
            <ast col="19" id="17179869289" line="12" text="@" type="AT">
              <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
              <ast col="21" id="17179869291" line="12" text="fieldx" type="VAR_CHAR">
                <annotation datatype="java.lang.Long" key="oldtype" value="2782"/>
                <annotation datatype="java.lang.Long" key="refid" value="17179869207"/>
                <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
                <annotation datatype="java.lang.String" key="getter" value="getFieldx"/>
                <annotation datatype="java.lang.String" key="setter" value="setFieldx"/>
                <annotation datatype="java.lang.String" key="widgettype" value="FillInWidget"/>
                <annotation datatype="java.lang.String" key="javaname" value="fieldx"/>
                <annotation datatype="java.lang.String" key="accessor" value="widgetFieldx"/>
                <annotation datatype="java.lang.Boolean" key="lastref" value="true"/>
                <annotation datatype="java.lang.Long" key="frame-id" value="17179869343"/>
              </ast>
            </ast>
          </ast>
          <ast col="28" hidden="true" id="17179869293" line="12" text="with" type="FRAME_PHRASE">
            <annotation datatype="java.lang.Long" key="frame-id" value="17179869343"/>
            <ast col="33" hidden="true" id="17179869295" line="12" text="frame" type="KW_FRAME">
              <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
              <ast col="39" id="17179869297" line="12" text="sf1" type="WID_FRAME"/>
            </ast>
          </ast>
        </ast>
      </ast>
```

#8 - 01/03/2020 01:38 PM - Roger Borrello

Creating a standalone NON-SHARED testcase:

```
def var fieldx as char no-undo init "fxval".
```

```
def var mychar as char no-undo init "myval".

form fieldx with frame sf1.
form mychar with frame sf2.

loopy:
repeat with frame sf2:
   /* When sf1 is shared, it breaks regardless of sf2 being shared or not */
   display mychar @ fieldx with frame sf1.
   if input mychar = "" then do:
      message "Done.".
      leave loopy.
   end.
end.
```

Generated business logic NonsharedFrameMixupStandalone.java:

```
package com.goldencode.testcases.sharedframes;

import com.goldencode.p2j.util.*;
import com.goldencode.testcases.ui.sharedframes.*;
import com.goldencode.p2j.ui.*;

import static com.goldencode.p2j.util.BlockManager.*;
import static com.goldencode.p2j.util.CompareOps.*;
import static com.goldencode.p2j.ui.LogicalTerminal.*;

/**
 * Business logic (converted to Java from the 4GL source code
 * in sharedframes/nonshared_frame_mixup-standalone.p).
 */
public class NonsharedFrameMixupStandalone
{
   NonsharedFrameMixupStandaloneSf2 sf2Frame = GenericFrame.createFrame(NonsharedFrameMixupStandaloneSf2.class
, "sf2");

   NonsharedFrameMixupStandaloneSf1 sf1Frame = GenericFrame.createFrame(NonsharedFrameMixupStandaloneSf1.class
, "sf1");

   /**
    * External procedure (converted to Java from the 4GL source code
    * in sharedframes/nonshared_frame_mixup-standalone.p).
    */
   public void execute()
   {
      character fieldx = TypeFactory.character("fxval");
      character mychar = TypeFactory.character("myval");

      externalProcedure(NonsharedFrameMixupStandalone.this, new Block((Body) () ->
      {
         sf1Frame.openScope();
         sf2Frame.openScope();

         repeat("loopy", new Block((Body) () ->
         {
            FrameElement[] elementList0 = new FrameElement[]
            {
               new Element(mychar, sf1Frame.widgetFieldx())
            };

            /* When sf1 is shared, it breaks regardless of sf2 being shared or not */
            sf1Frame.display(elementList0);

            if (_isEqual(sf1Frame.getMychar(), ""))
            {
               message("Done.");
               leave("loopy");
            }
         }));
      }));
   }
}
```

The ui files...

sf1

```
package com.goldencode.testcases.ui.sharedframes;

import com.goldencode.p2j.util.*;
import com.goldencode.p2j.ui.*;

public interface NonsharedFrameMixupStandaloneSf1
extends CommonFrame
{
   public static final Class configClass = NonsharedFrameMixupStandaloneSf1Def.class;

   public character getFieldx();

   public void setFieldx(character parm);

   public void setFieldx(String parm);

   public void setFieldx(BaseDataType parm);

   public FillInWidget widgetFieldx();

   public character getMychar();

   public void setMychar(character parm);

   public void setMychar(String parm);

   public void setMychar(BaseDataType parm);

   public FillInWidget widgetMychar();

   public static class NonsharedFrameMixupStandaloneSf1Def
   extends WidgetList
   {
      FillInWidget fieldx = new FillInWidget();

      FillInWidget mychar = new FillInWidget();

      public void setup(CommonFrame frame)
      {
         frame.setDown(1);
         fieldx.setDataType("character");
         fieldx.setAt(true);
         fieldx.setAtFormatLength(8);
         mychar.setDataType("character");
         mychar.setLabel("mychar");
         mychar.setFormat("x(8)");
         fieldx.setLabel("fieldx");
      }

      {
         addWidget("fieldx", "fieldx", fieldx);
         addWidget("mychar", "mychar", mychar);
      }
   }
}
```

sf2

```
package com.goldencode.testcases.ui.sharedframes;

import com.goldencode.p2j.util.*;
import com.goldencode.p2j.ui.*;
```

```
public interface NonsharedFrameMixupStandaloneSf2
extends CommonFrame
{
   public static final Class configClass = NonsharedFrameMixupStandaloneSf2Def.class;

   public character getMychar();

   public void setMychar(character parm);

   public void setMychar(String parm);

   public void setMychar(BaseDataType parm);

   public FillInWidget widgetMychar();

   public static class NonsharedFrameMixupStandaloneSf2Def
   extends WidgetList
   {
      FillInWidget mychar = new FillInWidget();

      public void setup(CommonFrame frame)
      {
         frame.setDown(1);
         mychar.setDataType("character");
         mychar.setLabel("mychar");
         mychar.setFormat("x(8)");
      }

      {
         addWidget("mychar", "mychar", mychar);
      }
   }
}
```

#9 - 01/03/2020 02:45 PM - Greg Shah

In both the shared and non-shared cases, the use of @ base-field syntax **does NOT create a new widget** in the 4GL. In FWD, we have multiple problems:

1. In the non-shared case, NonsharedFrameMixupStandaloneSf1 has a mychar widget that should not be there.
2. In both the shared and non-shard cases, the input mychar is matched with sf1/f1 instead of sf2/f2. That leads to an incorrect frame reference which is a compile error for the shard case (because the widget correctly does not exist) and which compiles for the non-shared case (because the incorrect widget is there) but should not be there.

#10 - 01/03/2020 03:26 PM - Roger Borrello

Greg Shah wrote:

> In both the shared and non-shared cases, the use of @ base-field syntax **does NOT create a new widget** in the 4GL. In FWD, we have multiple problems:
>
> 1. In the non-shared case, NonsharedFrameMixupStandaloneSf1 has a mychar widget that should not be there.
>
> - In both the shared and non-shard cases, the input mychar is matched with sf1/f1 instead of sf2/f2. That leads to an incorrect frame reference which is a compile error for the shard case (because the widget correctly does not exist) and which compiles for the non-shared case (because the incorrect widget is there) but should not be there.

The add_widget function in annotations/frame_scoping.rules is used to add a widget to the frame when isWidget is true:

```
            node.parent.type == prog.statement and
            (node.type == prog.kw_choose   or
             node.type == prog.kw_color    or
             node.type == prog.kw_disable  or
             node.type == prog.kw_enable   or
             node.type == prog.kw_disp     or
             node.type == prog.kw_form     or
             node.type == prog.kw_insert   or
             node.type == prog.kw_next_pmt or
             node.type == prog.kw_prmt_for or
             node.type == prog.kw_set      or
             node.type == prog.kw_underlin or
             node.type == prog.kw_update   or
             node.type == prog.define_frame)
AND
type == prog.format_phrase
AND
parent.type == prog.at
```

The comment before adding the widget is "the variables used in @ base clauses in format_phrase are also part of the frame" but perhaps there needs to be another filter to adding the widgets?

#11 - 01/03/2020 03:35 PM - Greg Shah

> The comment before adding the widget is "the variables used in @ base clauses in format_phrase are also part of the frame" but perhaps there needs to be another filter to adding the widgets?

No. This code is probably correct. The issue is that the mychar (variable reference of type VAR_CHAR) **which is NOT the child of prog.at** is being added as a widget.

#12 - 01/03/2020 04:03 PM - Roger Borrello

Greg Shah wrote:

> > The comment before adding the widget is "the variables used in @ base clauses in format_phrase are also part of the frame" but perhaps there needs to be another filter to adding the widgets?
>
> No. This code is probably correct. The issue is that the mychar (variable reference of type VAR_CHAR) **which is NOT the child of prog.at** is being added as a widget.

When adding a widget, I added this debug: printfln("frame_scoping.add_widget: Adding '%s' to frame '%s' in stmt '%s' at line %s\n%s", jName, frame.get(frameName), parent.text, line, this.dumpTree())

the nonshared_frame_mixup-standalone.p testcase showed:

```
     [java] ./uast/sharedframes/nonshared_frame_mixup-standalone.p
     [java] frame_scoping.add_widget: Adding 'expr1' to frame '' in stmt 'form item' at line 0
     [java] expression [EXPRESSION]:12884901930 @0:0
     [java]    fieldx [VAR_CHAR]:12884901931 @4:6
     [java]
     [java] ## INFO: ./uast/sharedframes/nonshared_frame_mixup-standalone.p: assuming DOWN set to 1 for f1
     [java] frame_scoping.add_widget: Adding 'expr2' to frame '' in stmt 'form item' at line 0
     [java] expression [EXPRESSION]:12884901944 @0:0
     [java]    mychar [VAR_CHAR]:12884901945 @5:6
     [java]
     [java] ## INFO: ./uast/sharedframes/nonshared_frame_mixup-standalone.p: assuming DOWN set to 1 for f2
     [java] frame_scoping.add_widget: Adding 'expr3' to frame '' in stmt 'display' at line 0
     [java] expression [EXPRESSION]:12884901973 @0:0
     [java]    mychar [VAR_CHAR]:12884901974 @10:12
     [java]
     [java] frame_scoping.add_widget: Adding 'expr4' to frame '' in stmt '@' at line 10
     [java] fieldx [VAR_CHAR]:12884901979 @10:21
     [java]
```

The jName values being expr... is a little unexpected, as that is what was added to the widget list with wlist.put(jName, copy). But I do see mychar being added on the display statement.

#13 - 01/03/2020 04:45 PM - Roger Borrello

Roger Borrello wrote:

> The jName values being expr... is a little unexpected, as that is what was added to the widget list with wlist.put(jName, copy). But I do see mychar being added on the display statement.

Added some more debug, and it looks like we are adding widgets in this case, as well:

```
      <!-- COLOR/DISPLAY/FORM/SET/UNDELINE/UPDATE/INSERT/PROMPT-FOR statements -->
      <rule>isWidget and !inFormat and type != prog.kw_no_error and
         (parent.type == prog.kw_disp     or
          parent.type == prog.kw_disable  or
          parent.type == prog.kw_enable   or
          parent.type == prog.kw_color    or
          parent.type == prog.kw_insert   or
          parent.type == prog.kw_prmt_for or
          parent.type == prog.form_item   or
          parent.type == prog.kw_set      or
          parent.type == prog.kw_update   or
          (parent.parent.type == prog.kw_underlin and
           parent.type == prog.content_array))
```

```
         <rule>type != prog.kw_stream
            <action>printfln("frame_scoping: Calling add_widget COLOR/DISPLAY/FORM/SET/UNDELINE/UPDATE/INSERT/
PROMPT-FOR statements\n%s", parent.parent.dumpTree())</action>
            <action>evalLib("add_widget")</action>
         </rule>
      </rule>
```

Output:

```
     [java] ./uast/sharedframes/nonshared_frame_mixup-standalone.p
     [java] frame_scoping: Calling add_widget COLOR/DISPLAY/FORM/SET/UNDELINE/UPDATE/INSERT/PROMPT-FOR stateme
nts
     [java] form [KW_FORM]:12884901927 @4:1
     [java]    form item [FORM_ITEM]:12884901929 @0:0
     [java]       expression [EXPRESSION]:12884901930 @0:0
     [java]          fieldx [VAR_CHAR]:12884901931 @4:6
     [java]    with [FRAME_PHRASE]:12884901933 @4:13
     [java]       frame [KW_FRAME]:12884901935 @4:18
     [java]          f1 [WID_FRAME]:12884901937 @4:24
     [java]
     [java] frame_scoping.add_widget: Adding jName='expr1' to frame '' in stmt 'form item' parent==prog.at=fal
se
     [java] expression [EXPRESSION]:12884901930 @0:0
     [java]    fieldx [VAR_CHAR]:12884901931 @4:6
     [java]
     [java] frame_scoping: Calling add_widget COLOR/DISPLAY/FORM/SET/UNDELINE/UPDATE/INSERT/PROMPT-FOR stateme
nts
     [java] form [KW_FORM]:12884901941 @5:1
     [java]    form item [FORM_ITEM]:12884901943 @0:0
     [java]       expression [EXPRESSION]:12884901944 @0:0
     [java]          mychar [VAR_CHAR]:12884901945 @5:6
     [java]    with [FRAME_PHRASE]:12884901947 @5:13
     [java]       frame [KW_FRAME]:12884901949 @5:18
     [java]          f2 [WID_FRAME]:12884901951 @5:24
     [java]
     [java] frame_scoping.add_widget: Adding jName='expr2' to frame '' in stmt 'form item' parent==prog.at=fal
se
     [java] expression [EXPRESSION]:12884901944 @0:0
     [java]    mychar [VAR_CHAR]:12884901945 @5:6
     [java]
     [java] frame_scoping: Calling add_widget COLOR/DISPLAY/FORM/SET/UNDELINE/UPDATE/INSERT/PROMPT-FOR stateme
nts
     [java] statement [STATEMENT]:12884901970 @0:0
     [java]    display [KW_DISP]:12884901971 @10:4
```

```
     [java]       expression [EXPRESSION]:12884901973 @0:0
     [java]          mychar [VAR_CHAR]:12884901974 @10:12
     [java]       format phrase [FORMAT_PHRASE]:12884901976 @0:0
     [java]          @ [AT]:12884901977 @10:19
     [java]             fieldx [VAR_CHAR]:12884901979 @10:21
     [java]       with [FRAME_PHRASE]:12884901981 @10:28
     [java]          frame [KW_FRAME]:12884901983 @10:33
     [java]             f1 [WID_FRAME]:12884901985 @10:39
     [java]
     [java] frame_scoping.add_widget: Adding jName='expr3' to frame '' in stmt 'display' parent==prog.at=false
     [java] expression [EXPRESSION]:12884901973 @0:0
     [java]    mychar [VAR_CHAR]:12884901974 @10:12
     [java]
     [java] frame_scoping: Calling add_widget since inFormat=true and parent.type == prog.at
     [java] frame_scoping.add_widget: Adding jName='expr4' to frame '' in stmt '@' parent==prog.at=true
     [java] fieldx [VAR_CHAR]:12884901979 @10:21
```

#14 - 01/03/2020 05:13 PM - Roger Borrello

It looks to me like the next-to-last add_widget is the problem:

```
     [java] frame_scoping: Calling add_widget COLOR/DISPLAY/FORM/SET/UNDELINE/UPDATE/INSERT/PROMPT-FOR stateme
nts
     [java] statement [STATEMENT]:12884901970 @0:0
     [java]    display [KW_DISP]:12884901971 @10:4
     [java]       expression [EXPRESSION]:12884901973 @0:0
     [java]          mychar [VAR_CHAR]:12884901974 @10:12
     [java]       format phrase [FORMAT_PHRASE]:12884901976 @0:0
     [java]          @ [AT]:12884901977 @10:19
     [java]             fieldx [VAR_CHAR]:12884901979 @10:21
     [java]       with [FRAME_PHRASE]:12884901981 @10:28
     [java]          frame [KW_FRAME]:12884901983 @10:33
     [java]             f1 [WID_FRAME]:12884901985 @10:39
     [java]
     [java] frame_scoping.add_widget: Adding jName='expr3' to frame '' in stmt 'display' parent==prog.at=false
     [java] expression [EXPRESSION]:12884901973 @0:0
     [java]    mychar [VAR_CHAR]:12884901974 @10:12
```

#15 - 01/03/2020 05:30 PM - Greg Shah

Yes, that is what I was referencing in #4488-11. To solve it we need to know when this is an @ base-field reference instead of a new widget. This should be done in an earlier step and remembered by setting an annotation. Then we can avoid the add_widget when that annotation is present and set to true.

#16 - 01/03/2020 06:06 PM - Roger Borrello

Greg Shah wrote:

> Yes, that is what I was referencing in #4488-11. To solve it we need to know when this is an @ base-field reference instead of a new widget. This should be done in an earlier step and remembered by setting an annotation. Then we can avoid the add_widget when that annotation is present and set to true.

I'll look at using get_format_phrase in common-progress.rules to find the next FORMAT_PHRASE. At that point, if there is an immediate child that is prog.at (using <node>.getImmediateChild(prog.at, null)) we do **not** have a widget. If it is null, then we **do** have a widget.

#17 - 01/06/2020 10:42 AM - Roger Borrello

Roger Borrello wrote:

> Greg Shah wrote:
>
> > Yes, that is what I was referencing in #4488-11. To solve it we need to know when this is an @ base-field reference instead of a new widget. This should be done in an earlier step and remembered by setting an annotation. Then we can avoid the add_widget when that annotation is present and set to true.
>
> I'll look at using get_format_phrase in common-progress.rules to find the next FORMAT_PHRASE. At that point, if there is an immediate child that is prog.at (using <node>.getImmediateChild(prog.at, null)) we do **not** have a widget. If it is null, then we **do** have a widget.

Update to **add_widget** in annotations/frame_scoping.rules were to prevent addition of the field reference as a widget when it is part of *@ base-field*.

```
                <!-- Look for an @ base-field which should suppress the generation of a widget -->
                <action>ref = #(com.goldencode.ast.Aast) execLib("get_format_phrase", this)</action>
                <rule>ref != null
                   <action>printfln("frame_scoping.add_widget: We have a format phrase.\n%s", ref.dumpTree())<
/action>
                   <action on="false">printfln("frame_scoping.add_widget: No format phrase.")</action>
                   <action>ref = ref.getImmediateChild(prog.at, null)</action>
                   <rule>ref != null
                        <action>printfln("frame_scoping.add_widget: We do *not* have a widget.")</action>
                        <action on="false">printfln("frame_scoping.add_widget: Adding jName='%s' to frame '%s'
 in stmt '%s' parent==prog.at=%b\n%s",
                                                    jName, frame.get(frameName), parent.text, (parent.type == prog.at)
, this.dumpTree())</action>
                        <action on="false">wlist.put(jName, copy)</action>
                   </rule>
                </rule>
```

This causes a lot of issues:

```
     [java] ------------------------------------------------------------------------------
     [java] Code Conversion Annotations
     [java] ------------------------------------------------------------------------------
     [java]
     [java] Optional rule set [customer_specific_annotations_prep] not found.
     [java] ./uast/sharedframes/nonshared_frame_mixup-standalone.p
     [java] frame_scoping: Calling add_widget COLOR/DISPLAY/FORM/SET/UNDELINE/UPDATE/INSERT/PROMPT-FOR stateme
nts
     [java] form [KW_FORM]:12884901927 @4:1
     [java]    form item [FORM_ITEM]:12884901929 @0:0
     [java]       expression [EXPRESSION]:12884901930 @0:0
     [java]          fieldx [VAR_CHAR]:12884901931 @4:6
     [java]    with [FRAME_PHRASE]:12884901933 @4:13
     [java]       frame [KW_FRAME]:12884901935 @4:18
     [java]          f1 [WID_FRAME]:12884901937 @4:24
     [java]
     [java] frame_scoping.add_widget: No format phrase.
     [java] ## INFO: ./uast/sharedframes/nonshared_frame_mixup-standalone.p: assuming DOWN set to 1 for f1
     [java] frame_scoping: Calling add_widget COLOR/DISPLAY/FORM/SET/UNDELINE/UPDATE/INSERT/PROMPT-FOR stateme
nts
     [java] form [KW_FORM]:12884901941 @5:1
     [java]    form item [FORM_ITEM]:12884901943 @0:0
     [java]       expression [EXPRESSION]:12884901944 @0:0
     [java]          mychar [VAR_CHAR]:12884901945 @5:6
     [java]    with [FRAME_PHRASE]:12884901947 @5:13
     [java]       frame [KW_FRAME]:12884901949 @5:18
     [java]          f2 [WID_FRAME]:12884901951 @5:24
     [java]
     [java] frame_scoping.add_widget: No format phrase.
     [java] ## INFO: ./uast/sharedframes/nonshared_frame_mixup-standalone.p: assuming DOWN set to 1 for f2
     [java] frame_scoping: Calling add_widget COLOR/DISPLAY/FORM/SET/UNDELINE/UPDATE/INSERT/PROMPT-FOR stateme
nts
     [java] statement [STATEMENT]:12884901970 @0:0
     [java]    display [KW_DISP]:12884901971 @10:4
     [java]       expression [EXPRESSION]:12884901973 @0:0
     [java]          mychar [VAR_CHAR]:12884901974 @10:12
     [java]       format phrase [FORMAT_PHRASE]:12884901976 @0:0
     [java]          @ [AT]:12884901977 @10:19
     [java]             fieldx [VAR_CHAR]:12884901979 @10:21
     [java]       with [FRAME_PHRASE]:12884901981 @10:28
     [java]          frame [KW_FRAME]:12884901983 @10:33
     [java]             f1 [WID_FRAME]:12884901985 @10:39
     [java]
     [java] frame_scoping.add_widget: We have a format phrase.
     [java] format phrase [FORMAT_PHRASE]:12884901976 @0:0
     [java]    @ [AT]:12884901977 @10:19
     [java]       fieldx [VAR_CHAR]:12884901979 @10:21
     [java]
     [java] frame_scoping.add_widget: We do *not* have a widget.
     [java] frame_scoping: Calling add_widget since inFormat=true and parent.type == prog.at
     [java] frame_scoping.add_widget: No format phrase.
     [java] WARNING: Null annotation (frame-id) for fieldx [VAR_CHAR] @10:21
     [java] WARNING: Null annotation (getter) for fieldx [VAR_CHAR] @10:21
     [java] WARNING: Null annotation (setter) for fieldx [VAR_CHAR] @10:21
     [java] WARNING: Null annotation (accessor) for fieldx [VAR_CHAR] @10:21
     [java] WARNING: Null annotation (widgettype) for fieldx [VAR_CHAR] @10:21
...REPEATED...
     [java] WARNING: Null annotation (frame-id) for fieldx [VAR_CHAR] @10:21 (12884902035)
     [java] WARNING: Null annotation (getter) for fieldx [VAR_CHAR] @10:21 (12884902035)
     [java] WARNING: Null annotation (setter) for fieldx [VAR_CHAR] @10:21 (12884902035)
     [java] WARNING: Null annotation (accessor) for fieldx [VAR_CHAR] @10:21 (12884902035)
     [java] WARNING: Null annotation (widgettype) for fieldx [VAR_CHAR] @10:21 (12884902035)
     [java] EXPRESSION EXECUTION ERROR:
     [java] --------------------------
     [java] persist()
     [java] ^  { null value for annotation 'frame-id':fieldx [VAR_CHAR]:12884902035 @10:21
     [java]  }
     [java] --------------------------
     [java] ERROR:
     [java] com.goldencode.p2j.pattern.TreeWalkException: ERROR!  Active Rule:
     [java] ----------------------
     [java]       RULE REPORT
     [java] ----------------------
     [java] Rule Type :   POST
     [java] Source AST:  [ block ] BLOCK/ @0:0 {12884901889}
```

```
     [java] Copy AST  :  [ block ] BLOCK/ @0:0 {12884901889}
     [java] Condition :  persist()
     [java] Loop      :  false
     [java] --- END RULE REPORT ---
     [java]

AND A NPE
     [java] Caused by: java.lang.NullPointerException: null value for annotation 'frame-id':fieldx [VAR_CHAR]:
12884902035 @10:21
```

#18 - 01/06/2020 10:49 AM - Greg Shah

fieldx still needs all those annotations because it is a real widget reference. Either your changes are in the wrong place or this is just a latent bug that is now reachable when it was not before.

#19 - 01/06/2020 02:22 PM - Roger Borrello

Greg Shah wrote:

> fieldx still needs all those annotations because it is a real widget reference. Either your changes are in the wrong place or this is just a latent bug that is now reachable when it was not before.

Most likely I'm in the wrong place. I'm looking at the process_frame loop, which is very encompassing.

It covers all the widgets and does many things, some of which seem to be done more than once.

I added messages throughout the loop:

```
     [java] frame_scoping: process frame 'sf1'
     [java] frame_scoping: in loop jName='expr1'
     [java] form item [FORM_ITEM]:17179869241 @0:0
     [java]    expression [EXPRESSION]:17179869242 @0:0
     [java]       fieldx [VAR_CHAR]:17179869243 @6:6
     [java]
     [java] frame_scoping: expression processing
     [java] frame_scoping: parent is form item
     [java] frame_scoping: have child that is vartype or customer_specific
     [java] frame_scoping: setter='setFieldx', getter='getFieldx', jName='fieldx'
     [java] frame_scoping: putAnnotation jName='fieldx'
     [java] frame_scoping: putAnnotation getter='getFieldx'
     [java] frame_scoping: putAnnotation widgettype='FillInWidget'
     [java] frame_scoping: put_accessor
     [java] frame_scoping: track_widget
     [java] frame_scoping: Adding frame reference to rlist
     [java] frame_scoping: in loop jName='expr3'
     [java] display [KW_DISP]:17179869283 @12:4
     [java]    expression [EXPRESSION]:17179869285 @0:0
     [java]       mychar [VAR_CHAR]:17179869286 @12:12
     [java]    format phrase [FORMAT_PHRASE]:17179869288 @0:0
     [java]       @ [AT]:17179869289 @12:19
     [java]          fieldx [VAR_CHAR]:17179869291 @12:21
     [java]    with [FRAME_PHRASE]:17179869293 @12:28
     [java]       frame [KW_FRAME]:17179869295 @12:33 HIDDEN
     [java]          sf1 [WID_FRAME]:17179869297 @12:39
     [java]
     [java] frame_scoping: expression processing
     [java] frame_scoping: have child that is vartype or customer_specific
     [java] frame_scoping: setter='setFieldx', getter='getFieldx', jName='fieldx'
     [java] frame_scoping: putAnnotation jName='fieldx'
     [java] frame_scoping: putAnnotation getter='getFieldx'
```

```
    [java] frame_scoping: putAnnotation widgettype='FillInWidget'
    [java] frame_scoping: put_accessor
    [java] frame_scoping: track_widget
    [java] frame_scoping: Adding frame reference to rlist
    [java] frame_scoping: in loop jName='expr4'
    [java] @ [AT]:17179869289 @12:19
    [java]    fieldx [VAR_CHAR]:17179869291 @12:21
    [java]
    [java] frame_scoping: process regular variables, setter='setFieldx', getter='getFieldx', jName='fieldx'
    [java] frame_scoping: reg variables: putAnnotation getter='getFieldx'
    [java] frame_scoping: reg variables: putAnnotation setter='setFieldx'
    [java] frame_scoping: reg variables: putAnnotation widgettype='FillInWidget'
    [java] frame_scoping: reg variables: putAnnotation javaname='fieldx'
    [java] frame_scoping: reg variables: put_accessor
    [java] frame_scoping: reg variables: track_widget
    [java] frame_scoping: reg variables: add frame ref to rlist
    [java] frame_scoping: adjust widget type loop, wname='fieldx'
    [java] frame_scoping: process frame 'f2'
    [java] frame_scoping: in loop jName='expr2'
    [java] form item [FORM_ITEM]:17179869255 @0:0
    [java]    expression [EXPRESSION]:17179869256 @0:0
    [java]       mychar [VAR_CHAR]:17179869257 @7:6
    [java]
    [java] frame_scoping: expression processing
    [java] frame_scoping: parent is form item
    [java] frame_scoping: have child that is vartype or customer_specific
    [java] frame_scoping: setter='setMychar', getter='getMychar', jName='mychar'
    [java] frame_scoping: putAnnotation jName='mychar'
    [java] frame_scoping: putAnnotation getter='getMychar'
    [java] frame_scoping: putAnnotation widgettype='FillInWidget'
    [java] frame_scoping: put_accessor
    [java] frame_scoping: track_widget
    [java] frame_scoping: Adding frame reference to rlist
    [java] frame_scoping: adjust widget type loop, wname='mychar'
```

#20 - 01/07/2020 12:11 PM - Roger Borrello

Updates to:
rules/annotations/frame_scoping.rules
add_widget was modified to include another condition under which the widget is not added:

```
        <!-- Look for an @ base-field which should suppress the generation of a widget -->
        <action>ref = #(com.goldencode.ast.Aast) execLib("get_format_phrase", this)</action>
        <rule>ref != null
           <action>ref = ref.getImmediateChild(prog.at, null)</action>
           <rule>ref != null
              <action>do_add = false</action>
           </rule>
        </rule>
```

rules/annotations/screen_buffer.rules
copy_as_widget was modified to use the parent's annotations if no non_aggregate format_phrase is found from which to get the annotations.

Checked into 4207a-11370

#21 - 01/07/2020 12:14 PM - Roger Borrello

Roger Borrello wrote:

> Updates to:
> rules/annotations/frame_scoping.rules
> add_widget was modified to include another condition under which the widget is not added:
> [...]
>
> rules/annotations/screen_buffer.rules
> copy_as_widget was modified to use the parent's annotations if no non_aggregate format_phrase is found from which to get the annotations.
>
> **Checked into 4207a-11370**

This update resulted in a regression. Below is the testcase.

```
def var mymess as character format "x(50)" no-undo init "mymess-v".
display "Missing value" @ mymess with frame bogus down.
message "Done."
```

Results in:

```
    [javac] testcases/src/com/goldencode/testcases/sharedframes/SharedFrameBrokenDisplayStandalone.java:38: er
ror: cannot find symbol
    [javac]             new Element("Missing value", "x(13)", bogusFrame.widgetMymess())
    [javac]                                                   ^
    [javac]   symbol:   method widgetMymess()
    [javac]   location: variable bogusFrame of type SharedFrameBrokenDisplayStandaloneBogus
    [javac] 1 error
```

#22 - 01/07/2020 01:37 PM - Roger Borrello

Should this testcase:

```
def var mymess as character format "x(50)" no-undo init "mymess-v".
display "Missing value" @ mymess with frame bogus down.
message "Done."
```

Result this code?

```
public interface SharedFrameBrokenDisplayStandaloneBogus
extends CommonFrame
{
   public static final Class configClass = SharedFrameBrokenDisplayStandaloneBogusDef.class;

   public character getMymess();

   public void setMymess(character parm);

   public void setMymess(String parm);

   public void setMymess(BaseDataType parm);

   public FillInWidget widgetMymess();

   public static class SharedFrameBrokenDisplayStandaloneBogusDef
   extends WidgetList
   {
      FillInWidget mymess = new FillInWidget();

      public void setup(CommonFrame frame)
      {
         frame.setDown(0);
         mymess.setAt(true);
         mymess.setAtFormatLength(50);
         mymess.setFormat("x(50)");
         mymess.setDataType("character");
         mymess.setLabel("mymess");
      }

      {
         addWidget("mymess", "mymess", mymess);
      }
   }
}
```

Should mymess be a widget on that frame?

#23 - 01/07/2020 01:51 PM - Greg Shah

Yes. That is the entire point of an at-base field. It IS a widget in the frame.

#24 - 01/07/2020 02:36 PM - Roger Borrello

Greg Shah wrote:

> Yes. That is the entire point of an at-base field. It IS a widget in the frame.

Taking this example...

```
loopy:
repeat with frame f2:
   /* When sf1 is shared, it breaks regardless of sf2 being shared or not */
   display mychar @ fieldx with frame sf1.
   if input mychar = "" then do:
```

The mychar value ("myval") should be displayed in the frame sf1's fieldx widget. The code (without rev 11370) generates the code correctly for that. It's just the input is reading the sf1 buffer, when it should be reading from f2, since we are past the display scoped to sf1, and we should now be scoped to f2 frame for the getMychar() getter, due to the repeat with frame f2, correct?

```
         f2Frame.openScope();

         repeat("loopy", new Block((Body) () ->
         {
            FrameElement[] elementList0 = new FrameElement[]
            {
               new Element(mychar, sf1Frame.widgetFieldx())
            };

            /* When sf1 is shared, it breaks regardless of sf2 being shared or not */
            sf1Frame.display(elementList0);

            if (_isEqual(sf1Frame.getMychar(), ""))
            {
```

Just trying to determine if the issue is not that we are creating a widget where we aren't supposed to, but rather scoping the widget to the wrong frame.

#25 - 01/07/2020 03:25 PM - Greg Shah

> The mychar value ("myval") should be displayed in the frame sf1's fieldx widget.

Yes.

> The code (without rev 11370) generates the code correctly for that. It's just the input is reading the sf1 buffer, when it should be reading from f2,

Yes.

> since we are past the display scoped to sf1,

The display is not scoped to sf1, it is hard coded to sf1 using the frame phrase with frame sf1.

> and we should now be scoped to f2 frame for the getMychar() getter, due to the repeat with frame f2, correct?

Yes, as long as the testcase has some other code that adds mychar to f2.

#26 - 01/07/2020 03:43 PM - Roger Borrello

Greg Shah wrote:

> > and we should now be scoped to f2 frame for the getMychar() getter, due to the repeat with frame f2, correct?
>
> Yes, as long as the testcase has some other code that adds mychar to f2.

Exactly, which we do... form mychar with frame f2. is pretty specific. But when I look at the VAR_CHAR of the FUNC_POLY (input), the frame-id value

for both the FUNC_POLY and VAR_CHAR, they map back to the FRAME_ALLOC for sf1.

#27 - 01/07/2020 03:58 PM - Greg Shah

That does seem wrong. But I thought that you said the input mychar code did emit in Java with frame f2.

#28 - 01/07/2020 04:06 PM - Roger Borrello

Greg Shah wrote:

> That does seem wrong. But I thought that you said the input mychar code did emit in Java with frame f2.

It does, after our 11370 check-in. I'm backtracking to see if there's another approach.

#29 - 01/07/2020 04:15 PM - Greg Shah

What does this have to do with the regression in #4488-22?

#30 - 01/07/2020 04:32 PM - Roger Borrello

Greg Shah wrote:

> What does this have to do with the regression in #4488-22?

Just a different approach. With #4488-22, there is an empty frame class. With resetting to pre-11370, I can look at it knowing what happened from those updates. Which do you suggest?

```
package com.goldencode.testcases.ui.sharedframes;

import com.goldencode.p2j.util.*;
import com.goldencode.p2j.ui.*;

public interface SharedFrameBrokenDisplayStandaloneF1
extends CommonFrame
{
   public static final Class configClass = SharedFrameBrokenDisplayStandaloneF1Def.class;

   public static class SharedFrameBrokenDisplayStandaloneF1Def
   extends WidgetList
   {
      public void setup(CommonFrame frame)
      {
         frame.setDown(1);
      }

      {
      }
   }
}
```

#31 - 01/07/2020 07:10 PM - Roger Borrello

Looking at the more enhanced updates to convert/frame_generator.xml where we want to use firstref, lastref, and have parent.type == prog.at:

Do we need to recalculate the node.parent.type each iteration of the while node.parent.type != prog.statement? In other words, should I have these lines:

```
      <!-- Normally we don't add a widget from '@' but it is the only
           place we can get the widget information -->
      <action>firstref = isNote("firstref") and getNoteBoolean("firstref")</action>
      <action>lastref  = isNote("lastref")  and getNoteBoolean("lastref")</action>
      <action>underat  = node.parent.type == prog.at</action>
```

Inside or outside the while? Note, underat needs to change, depending upon the placement.

```
            <while>node.parent.type != prog.statement
               <rule>(node.type == prog.format_phrase and !firstref) or
                     node.type == prog.color_phrase  or
                     node.type == prog.validation    or
                     node.parent.type == prog.validation
                  <action>do_add = false</action>
               </rule>
               <action>node = node.parent</action>
            </while>
```

Also, are we checking for !firstref and !lastref and !underat (or !(firstref or lastref or underat)) where I have !firstref ?

#32 - 01/08/2020 08:24 AM - Greg Shah

> Do we need to recalculate the node.parent.type each iteration of the while node.parent.type != prog.statement?

I want you to answer your own question. Answer this first: What is the copy node when this function is called in the case we are dealing with? (hint: see at_base_field_clause in the parser)

Post your reasoning here.

> Also, are we checking for !firstref and !lastref and !underat (or !(firstref or lastref or underat)) where I have !firstref ?

We are trying to restrict this condition to:

- the only reference to this widget; AND
- the base field for an at-base field reference

So...

#33 - 01/08/2020 10:50 AM - Roger Borrello

Greg Shah wrote:

> Do we need to recalculate the node.parent.type each iteration of the while node.parent.type != prog.statement?

I want you to answer your own question. Answer this first: What is the copy node when this function is called in the case we are dealing with? (hint: see at_base_field_clause in the parser)

The parser is expecting an lvalue after '@'. This ruleset will get to add_widget whenever there is a node with a frame-id, but in the case we are dealing with, it's the VAR_CHAR node, which has the annotations for the widget, which were formerly in the EXPRESSION node (now prevented by 11370). COPY/THIS will be that VAR_CHAR node, so I don't need to worry about the <while> traversing the parents in terms of evaluating the conditions of the VAR_CHAR node under question.

> Also, are we checking for !firstref and !lastref and !underat (or !(firstref or lastref or underat)) where I have !firstref ?

> We are trying to restrict this condition to:
>
> - the only reference to this widget; AND
> - the base field for an at-base field reference
>
> So...

We don't want do_add to be set to false at this VAR_CHAR, so while the loop is moving from **node** to **node.parent**, we'll prevent do_add from becoming false if (firstref and lastref) and underat are all true.

```
            <!-- Normally we don't add a widget from '@' but it is the only
                 place we can get the widget information -->
            <action>firstref = isNote("firstref") and getNoteBoolean("firstref")</action>
            <action>lastref  = isNote("lastref")  and getNoteBoolean("lastref")</action>
            <action>underat  = this.parent.type == prog.at</action>
            <action>override_at = (firstref and lastref and underat)</action>
```

```
            <!-- check if this is a part of widget creation statement -->
            <action>node = this</action>
            <while>node.parent.type != prog.statement
               <rule>(node.type == prog.format_phrase and !override_at) or
```

```
                    node.type == prog.color_phrase  or
                    node.type == prog.validation    or
                    node.parent.type == prog.validation
                 <action>do_add = false</action>
              </rule>
              <action>node = node.parent</action>
           </while>
```

#34 - 01/08/2020 12:50 PM - Greg Shah

Good.

#35 - 01/08/2020 02:10 PM - Roger Borrello

- Status changed from New to WIP

#36 - 01/08/2020 02:10 PM - Roger Borrello

Updates are in 4207a-11372.

#37 - 01/08/2020 04:07 PM - Roger Borrello

Checked in these testcases:

```
added uast/at/at_frame_broken_display-standalone.p
added uast/at/at_frame_mixup-shared-run.p
added uast/at/at_frame_mixup-shared.p
added uast/at/at_frame_mixup-standalone.p
```

#38 - 03/03/2020 02:56 PM - Roger Borrello

- % Done changed from 0 to 100

- Assignee set to Roger Borrello

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

#39 - 03/04/2020 10:31 AM - Greg Shah

- Status changed from WIP to Closed