User Interface - Bug #4631

Widget scoped to wrong frame on hide

04/30/2020 08:31 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 - 04/30/2020 08:33 AM - Roger Borrello

- Assignee set to Roger Borrello

Testcase: ./uast/frames/nested_scoping_frame.p

```
def var mymy   as char init "Felix".
def var mymess as char init "Oscar".

def frame y mymess.
repeat with frame y:
   display mymy.
   repeat with frame x:
      hide mymess.
      leave.
   end.
   leave.
end.
message "Done.".
```

Results in:

```
    [javac] /home/rfb/projects/VirtualBox-VMs/shared/projects/testcases/src/com/goldencode/testcases/frames/Ne
stedScopingFrame.java:50: error: cannot find symbol
    [javac]                 xFrame.widgetMymess().hide(false);
    [javac]                       ^
    [javac]   symbol:   method widgetMymess()
    [javac]   location: variable xFrame of type NestedScopingFrameX
```

#2 - 04/30/2020 10:51 AM - Roger Borrello

This is related revision 11497 of rules/annotations/frame_scoping.rules. Researching.

#3 - 04/30/2020 12:32 PM - Roger Borrello

The change to restrict using the resolved frame when there isn't an explicit frame specified was too strict. It excluded any frame when we are in an override frame situation. It should have included when there isn't an at-base field in the format phrase.

Updates in 4231b-11505.

#4 - 04/30/2020 12:32 PM - Roger Borrello

- % Done changed from 0 to 100

#6 - 04/30/2020 05:12 PM - Roger Borrello

Still working on this, as that fix caused breakage elsewhere.

Greg, in this testcase:

```
def var mymy   as char init "Felix".
def var mymess as char init "Oscar".
def frame y mymess.

repeat with frame y:
   display mymy.
   repeat with frame x:
      hide mymess.
      leave.
   end.
   leave.
end.
```

please confirm that mymess should be scoped to frame x. If so, this is more of a case of mymess not being added as a widget to frame x.

#7 - 04/30/2020 05:59 PM - Greg Shah

> please confirm that mymess should be scoped to frame x. If so, this is more of a case of mymess not being added as a widget to frame x.

You need to answer this by testing in the 4GL.

What I recall happens:

- repeat with frame x: does not create a frame x if one does not already exist
- hide, view and apply do not define new frames

So my SWAG is that there is no spoon (frame x). To prove this, add a view frame x to see if mymess appears. I suspect it will not even work.

#8 - 05/01/2020 12:48 AM - Roger Borrello

There was a missing def frame x I added, after taking a closer look at the code that broke.

```
def var mymy   as char init "Felix".
def var mymess as char init "Oscar".
def frame y mymess.
def frame x.

repeat with frame y:
   display mymy.
   pause.
   repeat with frame x:
      hide mymess.
      pause.
      view frame x.
      leave.
   end.
   leave.
end.
```

4GL shows that the widget is on frame y.

There are too many testcases failing for various reasons. For example, this testcase fails to convert due to null annotations.

```
def var junk as char.
do with frame bogus:
  display junk.
  pause.
  down.
end.
```

The implementation approach I took to maintain the legacy_name for the original frame must be flawed. Everywhere the frame name was being stored/retrieved from the frameName map in lowercase, I was doing the same with a frameLName map in the original case. Those maps are stored in the currScope object dictionary.

I am thinking it might be easier to store the legacy_name in a string dictionary, much like we do for the override_frame. Key item is to store the annotation in the FRAME_ALLOC, which is written out as one of the last items frame_scoping performs.

#9 - 05/01/2020 12:48 AM - Roger Borrello

- % Done changed from 100 to 50

#10 - 05/01/2020 09:04 AM - Greg Shah

> Everywhere the frame name was being stored/retrieved from the frameName map in lowercase, I was doing the same with a frameLName map in the original case.

You are not storing maps, you are storing strings in a map (the frame map that is passed around everywhere) and you are storing strings in the currScope dictionary.

Still, it seems complicated and I would hope for a simpler solution.

A key point here is that the legacy name MUST NOT be used for any kind of lookup. We MUST use the normalized lowercase name for that. The legacy name is ONLY needed to be stored so that it can be passed to the frame creation code.

#11 - 05/01/2020 11:01 AM - Roger Borrello

I was able to find a few critical places where frameLName was not being stored off, resulting in the downstream breakage.

- name_scope where the given frame name's scope cannot be located, it is added
- process_frame when a default frame is found and a frame name is calculated.
  - Also added here was a validation check that there is a valid frameLName, otherwise we fall back on the given frame name

I also removed the previous update, which was not related to the root cause.

All the failed procedures are processing, and I am running through all the testcases related to frames and browses.

#12 - 05/01/2020 11:01 AM - Roger Borrello

- % Done changed from 50 to 90

#13 - 05/01/2020 11:18 AM - Roger Borrello

- % Done changed from 90 to 100

- Status changed from New to WIP

#14 - 05/01/2020 11:19 AM - Roger Borrello

Committed revision 4231b-11509.

#15 - 05/04/2020 05:54 PM - Roger Borrello

Task ready for test.

#16 - 05/20/2020 09:37 AM - Roger Borrello

- % Done changed from 100 to 70

Added uast/frames/frame_broken_hide.p testcase, which also exhibits runtime issues with a widget being passed in when it isn't supposed to be.

#17 - 06/15/2020 08:12 AM - Roger Borrello

#4208-447 starts discussion of this same bug. This should be addressed before merging 4231b to trunk.

Constantin indicated: Roger, the problem is the editing block. This is a case of a 'statement in a statement' - as the fename gets set to null on ascent-rules, for the prog.statement node.

We need to use a scoped dictionary for this (and scopes are pushed on descent and popped on ascent, depending on who sets the fename). Look into the ascent-rules section of ui_statements.rules and see if anything else should be in a dictionary.

Yes, this needs to be fixed. If you don't want to reconvert now, just fix the Java code manually (remove the argument for hide) and see what gets you.

As an example for dictionary usage, see the "function" scope in base_structure.xml - how the addScope, deleteScope, lookup is done. The condition for add/delete will be different for this case.

The dictionary name is used for the scope name in addScope and deleteScope. And you use "fename" string in the dictionary to save/restore the fename variable, on descent/ascent.

#18 - 06/16/2020 07:46 PM - Roger Borrello

- File ui_statements.rules added

Should the descent-rules both add the scope at the appropriate time, and include the rules that had been in place in the walk-rules? I took the rules that filtered down when the fename was retrieved. My first attempt was a big failure, because all the display() methods are missing the frame element names. The hide() method is, as well, but that's just an anomaly.

```
      <rule>type == prog.content_array and parent.type == prog.statement
         <action>addScope("frame_element_name")</action>

         <action>fename = null</action>
         <rule>numImmediateChildren > 0
            <rule>evalLib("non_aggregate")
               <action>lastid = closestPeerId</action>
               <!-- Dealing with an aggregate, so reach to the peer node -->
               <action on="false">
                  scoperef = getAst(#(long) copy.getAnnotation("scope-id"))
               </action>

               <rule on="false">
                  scoperef.getAnnotation("aggr-id") == null
                  <action>
                     lastid = getAst(#(long) scoperef.getAnnotation("peerid")).parent.id
                  </action>

                  <action on="false">
                     lastid = null
                  </action>
               </rule>
            </rule>
            <rule>lastid != null
               <!-- determine the instance name -->
               <action>fename = getNoteString("javaname")</action>
               <action>addDictionaryObject("frame_element_name", "fename", fename)</action>
            </rule>
         </rule>
      </rule>
```

#19 - 06/17/2020 11:30 AM - Roger Borrello

- File ui_statements.rules added

I have an update that I'd like reviewed. It correctly converted the testcase, as well as the customer procedure. Are there any regression testing planned I could slipstream into? Or should I run my own?

#20 - 06/18/2020 07:08 PM - Greg Shah

The overall approach seems close. The one thing that looks wrong is line 1884 (<action>addDictionaryObject("frame_element_name", "fename", fename)</action>) where **during a walk rule** the current version of the fename in the current dictionary scope is overwritten. On line 711 you set that value when descending from the prog.statement node. The code in line 1884 is inside a type prog.content_array and parent.type prog.statement which means that it will definitely overwrite the just saved fename value.

Why is that code there?

> Are there any regression testing planned I could slipstream into?

No.

> Or should I run my own?

Yes. Use the ChUI application testing for this. It is a HEAVY user of EDITING blocks. We should have a new branch tomorrow for you to commit into. But go ahead and get it tested first.

#21 - 06/19/2020 09:19 AM - Roger Borrello

Greg Shah wrote:

> The overall approach seems close. The one thing that looks wrong is line 1884 (<action>addDictionaryObject("frame_element_name", "fename", fename)</action>) where **during a walk rule** the current version of the fename in the current dictionary scope is overwritten. On line 711 you set that value when descending from the prog.statement node. The code in line 1884 is inside a type prog.content_array and parent.type prog.statement which means that it will definitely overwrite the just saved fename value.
>
> Why is that code there?

It is most likely isn't needed.

> Yes. Use the ChUI application testing for this. It is a HEAVY user of EDITING blocks. We should have a new branch tomorrow for you to commit into. But go ahead and get it tested first.

I am undergoing some disk space challenges, and am backing up a couple of projects to my new thumb drive.

#22 - 06/19/2020 09:33 AM - Greg Shah

The ChUI testing is on devsrv01.

#23 - 06/19/2020 05:55 PM - Roger Borrello

Greg Shah wrote:

> The ChUI testing is on devsrv01.

After the complete conversion testing, there were no artifact differences in the 4231b-11611 and the conversion and the conversion with my convert/ui_statement.rules update applied.

#24 - 06/20/2020 10:18 AM - Greg Shah

Was this with the version that had removed the addDictionaryObject() which I asked about?

If so, does that same version fix all your standalone testcases and also the customer ChUI application for which it was intended?

#25 - 06/21/2020 09:26 PM - Roger Borrello

Greg Shah wrote:

> Was this with the version that had removed the addDictionaryObject() which I asked about?
>
> If so, does that same version fix all your standalone testcases and also the customer ChUI application for which it was intended?

Affirmative to both questions. Committed in 3821c-11353.

#26 - 06/22/2020 06:39 AM - Greg Shah

- Status changed from WIP to Test

- % Done changed from 70 to 100

Code Review Task Branch 3821c Revision 11353

The changes are good.

#28 - 12/04/2020 10:42 AM - Greg Shah

Can I close this?

#29 - 12/06/2020 10:52 PM - Roger Borrello

Greg Shah wrote:

> Can I close this?

Yes!

#30 - 12/07/2020 07:00 AM - Greg Shah

- Status changed from Test to Closed

Files

ui_statements.rules	153 KB	06/16/2020	Roger Borrello
ui_statements.rules	152 KB	06/17/2020	Roger Borrello