User Interface - Bug #6038

ComboBox setScreenValue improvements

02/03/2022 06:19 AM - Marian Edu

Status:	Feedback	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Marian Edu	**% Done:**	80%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

Related issues:

Related to User Interface - Bug #6016: Selection list setScreenValue improvem...	**Closed**
Related to User Interface - Bug #6057: Attributes access causing widget to be...	**WIP**

History

#1 - 02/03/2022 08:40 AM - Greg Shah

- Related to Bug #6016: Selection list setScreenValue improvements added

#2 - 02/04/2022 07:19 AM - Marian Edu

- File 6038-20220204.patch added

- % Done changed from 0 to 80

Fixes for ComboBox/Radioset that re-uses the basic value validation from base class ControlSetEntity and extend as needed.

Combo:
- single/drop everything works even if not valid
- if valid first item that matches is being selected (might have different case)
- if empty and not a valid option is set to unknown
- unknown value is supported, reset selection and is shown as empty string not question mark
- no multiple values allowed (delimiter)

RadioSet:
- unknown not supported
- empty is ignored if not valid option
- if valid first item that matches is being selected (might have different case)
- no multiple values allowed (delimiter)

Remains to be done:
- test in CHUI
- write unit tests if possible without user interaction

Attached the patch made against rev.13479 (xref).

#3 - 02/04/2022 07:25 AM - Greg Shah

- Start date deleted (02/03/2022)

Hynek: Please review.

Marian: Let us know if the ChUI testing shows any issues.

#4 - 02/04/2022 11:46 AM - Hynek Cihlar

Code review 6038-20220204.patch.

validateScreenValue, the method javadoc is a bit misleading. It suggests SCREEN-VALUE is updated by the method itself, while it just validates the passed in value. Also please make it more explicit in the javadoc that the passed in argument may be reassigned. And also make sure to always pass a new instance of the value in validateScreenValue, currently the argument to setScreenValue is passed in. This behavior may not be always obvious for the callers of setScreenValue and may cause issues later.

In CBW.validateScreenValue there is a comment // if value empty and not valid value reset to unknown. At that point the value is already valid, so the comment is misleading.

In RSW the methods isValid* and validateScreen* should be moved in the protected section of the class file.

The original behavior of RSW.setScreenValue was to silently return false when empty string passed in. Now error message is displayed. Was this intentional?

#5 - 02/04/2022 12:05 PM - Marian Edu

Hynek Cihlar wrote:

> Code review 6038-20220204.patch.
>
> validateScreenValue, the method javadoc is a bit misleading. It suggests SCREEN-VALUE is updated by the method itself, while it just validates the passed in value. Also please make it more explicit in the javadoc that the passed in argument may be reassigned.

Right, will do so.

> And also make sure to always pass a new instance of the value in validateScreenValue, currently the argument to setScreenValue is passed in. This behavior may not be always obvious for the callers of setScreenValue and may cause issues later.

Well, that was the whole idea behind re-assigning it's value - the actual screen-value in the end might need to be different from what was sent as input and we need to update that before it gets set in the frame's screen-buffer. Alternatively I can change validateScreenValue to return a character with updated value and the change the setScreenValue in ControlSetEntity to handle the case when different from the original value so we avoid updating the reference we get.

> In CBW.validateScreenValue there is a comment // if value empty and not valid value reset to unknown. At that point the value is already valid, so the comment is misleading.

Will take that out, just a left over from something I've tried.

> In RSW the methods isValid* and validateScreen* should be moved in the protected section of the class file.

I'm usually adding things at the end most of the time, with the outline view I hardly need to have visibility based sections but will move them for the vim lovers :)

> The original behavior of RSW.setScreenValue was to silently return false when empty string passed in. Now error message is displayed. Was this intentional?

It shouldn't be any error message when trying to set empty string as screen-value on the RWS, maybe I'm missed something from the patch but I don't see how the error message could by raised :(

#6 - 02/04/2022 12:23 PM - Hynek Cihlar

Marian Edu wrote:

> Hynek Cihlar wrote:
>
> > Code review 6038-20220204.patch.
> >
> > And also make sure to always pass a new instance of the value in validateScreenValue, currently the argument to setScreenValue is passed in. This behavior may not be always obvious for the callers of setScreenValue and may cause issues later.
>
> Well, that was the whole idea behind re-assigning it's value - the actual screen-value in the end might need to be different from what was sent as input and we need to update that before it gets set in the frame's screen-buffer. Alternatively I can change validateScreenValue to return a character with updated value and the change the setScreenValue in ControlSetEntity to handle the case when different from the original value so we avoid updating the reference we get.

It is OK to mutate the argument in validateScreenValue I think. My point was not to leak the change out of setScreenValue. It should be enough to create new instance in setScreenValue and pass that in validateScreenValue, super methods, etc.

> In RSW the methods isValid* and validateScreen* should be moved in the protected section of the class file.
>
> I'm usually adding things at the end most of the time, with the outline view I hardly need to have visibility based sections but will move them for the vim lovers :)

Well, this rule is carved in stone (aka code standards), no way around :-).

> The original behavior of RSW.setScreenValue was to silently return false when empty string passed in. Now error message is displayed.

> Was this intentional?

It shouldn't be any error message when trying to set empty string as screen-value on the RWS, maybe I'm missed something from the patch but I don't see how the error message could by raised :(

Perhaps I'm not interpreting the code directly. This is the flow I see. Empty value is passed in RSW.setScreenValue, CSE.setScreenValue is executed, RSW.validateScreenValue is executed, CSE.validateScreenValue is executed, RSW.isValidScreenValue returns false, and voilà the error 4058 is shown.

#7 - 02/04/2022 12:29 PM - Marian Edu

Hynek Cihlar wrote:

> It is OK to mutate the argument in validateScreenValue I think. My point was not to leak the change out of setScreenValue. It should be enough to create new instance in setScreenValue and pass that in validateScreenValue, super methods, etc.

Realised that when back to the code and implemented the change already :)

> > In RSW the methods isValid* and validateScreen* should be moved in the protected section of the class file.
>
> Well, this rule is carved in stone (aka code standards), no way around :-).

Understood :)

> Perhaps I'm not interpreting the code directly. This is the flow I see. Empty value is passed in RSW.setScreenValue, CSE.setScreenValue is executed, RSW.validateScreenValue is executed, CSE.validateScreenValue is executed, RSW.isValidScreenValue returns false, and voilà the error 4058 is shown.

Why would RSW.isValidScreenValue return false there, it's false only if unknown or not empty and not valid (super).. are you mentally debugging it? :)

#8 - 02/04/2022 12:34 PM - Hynek Cihlar

Marian Edu wrote:

> Hynek Cihlar wrote:
>
> > Perhaps I'm not interpreting the code directly. This is the flow I see. Empty value is passed in RSW.setScreenValue, CSE.setScreenValue is executed, RSW.validateScreenValue is executed, CSE.validateScreenValue is executed, RSW.isValidScreenValue returns false, and voilà the error 4058 is shown.
>
> Why would RSW.isValidScreenValue return false there, it's false only if unknown or not empty and not valid (super)..

True. If empty is valid for Radio set, then you can disregard this.

> are you mentally debugging it? :)

Yes, apparently all my cores are overutilized. :-)

#9 - 02/04/2022 12:55 PM - Marian Edu

- File 6038-20220204.02.patch added

Attached the patch with changes as per Hynek's review.

Hynek Cihlar wrote:

> In CBW.validateScreenValue there is a comment // if value empty and not valid value reset to unknown. At that point the value is already valid, so the comment is misleading.

Fixed the comment here, the value can be valid but not in the list of items - the thing is for unknown value 4GL looks for any item with an empty string value and if found selects that, otherwise it leave it as unknown.

#10 - 02/04/2022 01:08 PM - Greg Shah

Hynek: Merge this to 3821c if you are OK with the final version.

Marian: Is there anything remaining in this task?

#13 - 02/05/2022 11:32 AM - Hynek Cihlar

The changes in 6038-20220204.02.patch are good. Checked them in 3821c revision 13481.

#14 - 02/06/2022 08:29 AM - Eugenie Lyzenko

Hynek Cihlar wrote:

> The changes in 6038-20220204.02.patch are good. Checked them in 3821c revision 13481.

It has GUI regression. Please see respective entry for 3821c testing task.

#15 - 02/07/2022 09:07 AM - Greg Shah

Eugenie Lyzenko wrote:

> Hynek Cihlar wrote:
>
> > The changes in 6038-20220204.02.patch are good. Checked them in 3821c revision 13481.
>
> It has GUI regression. Please see respective entry for 3821c testing task.

As noted in #5034, this is resolved now, right?

Marian: Is there anything remaining to do in this task?

#16 - 02/07/2022 09:11 AM - Marian Edu

Greg Shah wrote:

> Marian: Is there anything remaining to do in this task?

I've wrote 4GL tests (static/dynamic) and there are some deviations, will make a list when complete and discuss what needs to be implemented. Since multiple widgets share the same base class ControlSetEntity I think we need to make sure we have some tests for all: combo, radio, selection-list so when we change something we don't add new regressions :(

#17 - 02/07/2022 09:17 AM - Greg Shah

Agreed.

#18 - 02/07/2022 09:36 AM - Eugenie Lyzenko

Greg Shah wrote:

> Eugenie Lyzenko wrote:
>
> > Hynek Cihlar wrote:
> >
> > > The changes in 6038-20220204.02.patch are good. Checked them in 3821c revision 13481.
> >
> > It has GUI regression. Please see respective entry for 3821c testing task.
>
> As noted in #5034, this is resolved now, right?

Yes, since 13484 it is gone.

#19 - 02/08/2022 05:09 AM - Marian Edu

OK, as per our tests so far it looks like the validation doesn't have anything to do with the fact that the widget's frame is(was) 'realised', it actually depends wether or not the widget itself was realised.

If the frame containing the widget became visible - display/enable/view - all its widget are being realised, unless hidden. If the widget has set as hidden then realising the frame doesn't realise the widget so setting an invalid value won't throw, once the widget is realised then validation start working and can't be turned back off by making the widget hidden or setting visible to true.

Another, less expected, case when the validation starts to be triggered is by simply accessing the screen-value (the getter) - it doesn't matter that the widget is hidden so is not really visible at all on the frame, it becomes 'realised' although remains hidden - from that point on setting invalid screen value throws.

#22 - 02/08/2022 10:19 AM - Greg Shah

Marian: We are backing out the latest 3821c in the customer test environment for both #5881 and #6044. Please let us know when you have the fix(es) that resolve those regressions so that we can move ahead in that test environment.

#23 - 02/08/2022 11:46 AM - Marian Edu

Greg Shah wrote:

> Marian: We are backing out the latest 3821c in the customer test environment for both #5881 and #6044. Please let us know when you have the fix(es) that resolve those regressions so that we can move ahead in that test environment.

Greg, not clear to me what the issues are there or if there were solved or not by changes in rev 13481.

#6044 was reposted as solved, then looks like it's not actually the case or there is something else?
#5881 what is the actual issue there? It was reported there is some focus issue introduced in rev 13481, then when the patch is applied (?) and the code looks now exactly like in rev 13481 everything seems to work again and the only confusion there is how the value changed from " " (space) to "" (empty, not really null)?

Right now I don't see how the validation can occur if the widget is not realised and adding an extra check as it was before on it's parent frame 'wasRealized' doesn't make much of a difference.

I'll probably send a patch tomorrow for that widget realisation clean-up, dropping the 'temporary' wasRealized configuration attribute - there is something that bothers me there, way to many places where that 'realized' attribute is check (using the string constant) while there is a method that does exactly that (most probably added later on, for whatever reason it does start with underscore - _isRealized - sounds like a private member to me).

#24 - 02/08/2022 12:16 PM - Eugenie Lyzenko

Marian,

Please see #5991.

We have severe regression from recent combo-box changes for (set/get)ScreenValue. Probably need to get a step back to restore applications functionality.

#25 - 02/08/2022 12:22 PM - Roger Borrello

With respect to #6044, the fixes in 13481 corrected the problem which exist with 13481. However, the fixes in 13481 regress the focus issue #5881.

I spent some time updating 13480 revision with the changes in 13481, and narrowed it down to the #5881-17 comments, where the focus is affected by the inclusion of the value setting in validateScreenValue.

#26 - 02/08/2022 01:26 PM - Marian Edu

Eugenie Lyzenko wrote:

> Marian,
>
> Please see #5991.
>
> We have severe regression from recent combo-box changes for (set/get)ScreenValue. Probably need to get a step back to restore applications functionality.

Thanks for that Eugenie, I can see how that could be a problem if the items have trailing spaces, have to check that in a quick 4GL test and will post a patch first thing tomorrow morning.

#27 - 02/08/2022 01:45 PM - Eugenie Lyzenko

Marian Edu wrote:

> Eugenie Lyzenko wrote:
>
> > Marian,
> >
> > Please see #5991.
> >
> > We have severe regression from recent combo-box changes for (set/get)ScreenValue. Probably need to get a step back to restore applications functionality.
>
> Thanks for that Eugenie, I can see how that could be a problem if the items have trailing spaces, have to check that in a quick 4GL test and will post a patch first thing tomorrow morning.

How about this change in GenericFrame.java?

```
   public void assignScreenValue(FrameElement data)
   {
      GenericWidget<?> widget = data.getWidget();
      int              wid    = getWidgetId(widget);
      Class<?>         type   = ((ControlEntity<?>) widget).getDataClass();

      // get the value out of the screen buffer (and make sure it is
      // converted to the right type if needed)
      BaseDataType value = (BaseDataType) getter(wid, type, false, false);

      if (value != null && value.isUnknown() && widget instanceof BrowseColumnWidget)
      {
         BrowseWidget browse = ((BrowseColumnWidget) widget).getBrowse();
         if (browse.getCurrentRow() == -1)
         {
            value = null;
         }
      }

      // only assign if there is a value in the screen buffer
```

```
      if (value != null)
      {
         if (value.isUnknown() && widget instanceof ComboBoxWidget)
         {
            data.set(value.instantiateDefault());
         }
+         // EVL***
+         else if (widget instanceof ComboBoxWidget && !value.isUnknown() &&
+                  value instanceof com.goldencode.p2j.util.Text)
+         {
+            data.set(TextOps.rightTrim((com.goldencode.p2j.util.Text)value));
+         }
+         // EVL***
         else
         {
            data.set(value);
         }
      }
   }
```

This fixes the regression I currently see.

#28 - 02/08/2022 01:51 PM - Marian Edu

Eugenie Lyzenko wrote:

> How about this change in GenericFrame.java?

I've started with GenericFrame before but turns out it's better to stop the problem for happening than trying to fix it later down the stream... why do they have spaces in the values in list-items is the question. I need to check what 4GL does in that case, it might be caused by how the list-items value is composed, maybe there should not be any spaces or it's also possible 4GL just trims those values and we need to know if it does that for list/radio as well or just combo.

#29 - 02/08/2022 02:14 PM - Eugenie Lyzenko

Marian Edu wrote:

> Eugenie Lyzenko wrote:
>
> > How about this change in GenericFrame.java?
>
> I've started with GenericFrame before but turns out it's better to stop the problem for happening than trying to fix it later down the stream... why do they have spaces in the values in list-items is the question.

The current format value is considering. If the text is less than format length the missing chars are auto filled with space.

> I need to check what 4GL does in that case, it might be caused by how the list-items value is composed, maybe there should not be any spaces or it's also possible 4GL just trims those values and we need to know if it does that for list/radio as well or just combo.

OK.

#30 - 02/08/2022 02:34 PM - Marian Edu

Testing this in 4GL only for combo-box the values are trimmed but only at the end, the list-items/list-item-pairs does show the trailing spaces as set though. Selection-list and radio-set keep the value as set, no leading/trailing spaces are removed.

#31 - 02/09/2022 07:16 AM - Marian Edu

- File 6038-20220209.patch added

Marian Edu wrote:

> Testing this in 4GL only for combo-box the values are trimmed but only at the end, the list-items/list-item-pairs does show the trailing spaces as set though. Selection-list and radio-set keep the value as set, no leading/trailing spaces are removed.

Attached the patch for combo-box so the trailing spaces from value are removed just like in 4GL and replace item's getCharacterValue with getValue because the formatting is applied when the item list is set and right-trim on formatted value - aka, if value is longer than format it will be cropped.

I've also dropped the temporary wasRealized attribute and always use realized instead, in fact replaced the getAttr call with the existing isRealized method.

#32 - 02/09/2022 08:44 AM - Hynek Cihlar

Marian Edu wrote:

> Marian Edu wrote:
> Attached the patch for combo-box so the trailing spaces from value are removed just like in 4GL and replace item's getCharacterValue with getValue because the formatting is applied when the item list is set and right-trim on formatted value - aka, if value is longer than format it will be cropped.

In ComboBoxWidget setLabel was called only when value type differed from the combo-box type. With your change setLabel is called even if the types are the same. Is this expected?

#33 - 02/09/2022 08:53 AM - Marian Edu

Hynek Cihlar wrote:

> In ComboBoxWidget setLabel was called only when value type differed from the combo-box type. With your change setLabel is called even if the types are the same. Is this expected?

Actually setLabel was only called if there are no pairs defined and that didn't change. In fact the only data type allowed in 4GL is character but then the duplicate through newValue was always done previously because of case sensitive compare between "character" (config.dataType) and "CHARACTER" returned by getTypeName method (as per it's javadoc that returns an upper case value).

#34 - 02/09/2022 08:56 AM - Hynek Cihlar

Hynek Cihlar wrote:

> Marian Edu wrote:
>
> > Marian Edu wrote:
> > Attached the patch for combo-box so the trailing spaces from value are removed just like in 4GL and replace item's getCharacterValue with getValue because the formatting is applied when the item list is set and right-trim on formatted value - aka, if value is longer than format it will be cropped.
>
> In ComboBoxWidget setLabel was called only when value type differed from the combo-box type. With your change setLabel is called even if the types are the same. Is this expected?.

I'm going to merge the patch on 3821c. The removal of wasRealized will require regression testing, to rule out cases where we distinguish widget realization by making the widget visible and by assigning certain attributes.

#35 - 02/09/2022 09:01 AM - Hynek Cihlar

I did a couple of cosmetic changes to 6038-20220209.patch and checked it in 3821c revision 13497.

#36 - 02/09/2022 09:40 AM - Roger Borrello

Using 13497 on my customer's application, and now receiving 4053 error: Unable to set SCROLLBAR-HORIZONTAL because the selection list xxx has been realized. (4053)

I am trying to get more details, but wanted to post this.

#37 - 02/09/2022 10:09 AM - Roger Borrello

Roger Borrello wrote:

> Using 13497 on my customer's application, and now receiving 4053 error: Unable to set SCROLLBAR-HORIZONTAL because the selection list xxx has been realized. (4053)
>
> I am trying to get more details, but wanted to post this.

Likewise I am getting ** Attribute SCREEN-VALUE for the COMBO-BOX xxx has an invalid value of SP. (4058)

This occurs from ControlSetEntity.validateScreenValue:

```
      if (!isValidScreenValue(value))
      {
         if (_isRealized())         //<--- have not debugged, but this is TRUE
         {
            ErrorManager.recordOrShowWarning(4058, String.format(
                     "Attribute SCREEN-VALUE for the %s %s has an invalid value of %s", type(),
                     widgetName(), (value.isUnknown() ? "UNKNOWN" : value.getValue())), true,
                     false, false, true);
         }

         return false;
      }
```

#38 - 02/09/2022 10:11 AM - Marian Edu

Roger Borrello wrote:

> Using 13497 on my customer's application, and now receiving 4053 error: Unable to set SCROLLBAR-HORIZONTAL because the selection list xxx has been realized. (4053)
>
> I am trying to get more details, but wanted to post this.

Yes, SCROLLBAR-HORIZONTAL can't be set once realized - this is also the 4GL behaviour.

Can you isolate that code in a simple test case?

#39 - 02/09/2022 10:12 AM - Roger Borrello

Roger Borrello wrote:

> Roger Borrello wrote:
>
> > Using 13497 on my customer's application, and now receiving 4053 error: Unable to set SCROLLBAR-HORIZONTAL because the selection list xxx has been realized. (4053)
> >
> > I am trying to get more details, but wanted to post this.
>
> Likewise I am getting ** Attribute SCREEN-VALUE for the COMBO-BOX xxx has an invalid value of SP. (4058)
>
> This occurs from ControlSetEntity.validateScreenValue:
> [...]

I notice the ComboBoxConfig.format is 2 exclamations marks: !!. I am not familiar with this format. Is it valid?

#40 - 02/09/2022 10:14 AM - Marian Edu

Roger Borrello wrote:

> Likewise I am getting ** Attribute SCREEN-VALUE for the COMBO-BOX xxx has an invalid value of SP. (4058)
>
> This occurs from ControlSetEntity.validateScreenValue:
> [...]

Indeed, once 'realized' the validation occurs and for DROP-DOWN-LIST combo-box if not a valid value the error is thrown... we need to find why those widgets gets realized before those attributes are being set. That might have been the whole idea behind the temporary solution with wasRealized.

#41 - 02/09/2022 10:21 AM - Marian Edu

Roger Borrello wrote:

> I notice the ComboBoxConfig.format is 2 exclamations marks: !!. I am not familiar with this format. Is it valid?

Yes, it is valid and it means it can be only letter and it will be turned in caps.

#42 - 02/09/2022 11:40 AM - Roger Borrello

Marian Edu wrote:

> Roger Borrello wrote:
>
> > Likewise I am getting ** Attribute SCREEN-VALUE for the COMBO-BOX xxx has an invalid value of SP. (4058)
> >
> > This occurs from ControlSetEntity.validateScreenValue:
> > [...]
>
> Indeed, once 'realized' the validation occurs and for DROP-DOWN-LIST combo-box if not a valid value the error is thrown... we need to find why those widgets gets realized before those attributes are being set. That might have been the whole idea behind the temporary solution with wasRealized.

I apologize... I must have incorrect data in my database. I am making a choice which results in SP, but that value is not in the combo-list, therefore this is a valid error to receive for this case.

As for the error in #6038-36 (Unable to set SCROLLBAR-HORIZONTAL because the selection list xxx has been realized. (4053)) I will post more information.

#43 - 02/09/2022 12:27 PM - Roger Borrello

Regarding what might be going on with setting the SCROLLBAR-HORIZONTAL attribute of a SELECTION-LIST after the frame is realized...

This is an ADM2 application, so it is a challenge to replicate. Some of the 4GL items:

```
DEFINE VARIABLE slChoices AS CHARACTER
     VIEW-AS SELECTION-LIST SINGLE
     SCROLLBAR-HORIZONTAL SCROLLBAR-VERTICAL
     SIZE 39.2 BY 14.33 NO-UNDO.
...
DEFINE FRAME frttSSeen
     slChoices AT ROW 3.33 COL 76 NO-LABEL NO-TAB-STOP
...
    WITH 1 DOWN NO-BOX KEEP-TAB-ORDER OVERLAY
         SIDE-LABELS NO-UNDERLINE THREE-D
         AT COL 3 ROW 1.71
         SIZE 116 BY 21.29.
```

The frame itself is set as a property:

```
ghProp:BUFFER-FIELD('WindowFrameHandle':U):BUFFER-VALUE = FRAME frttSSeen:handle
```

So there are probably some callbacks that are displaying the frame, which I haven't captured yet.

#44 - 02/09/2022 12:30 PM - Roger Borrello

There is this:

```
PROCEDURE enable_UI :
/*------------------------------------------------------------------------------
  Purpose:     ENABLE the User Interface
  Parameters:  <none>
  Notes:       Here we display/view/enable the widgets in the
               user-interface.  In addition, OPEN all queries
               associated with each FRAME and BROWSE.
               These statements here are based on the "Other
               Settings" section of the widget Property Sheets.
------------------------------------------------------------------------------*/
```

```
  DISPLAY ... slChoices ...
      WITH FRAME frttSSeen IN WINDOW wWin.
  ENABLE ... slChoices ... RECT-6 RECT-7 RECT-8
      WITH FRAME frttSSeen IN WINDOW wWin.

  VIEW wWin.
END PROCEDURE.
```

#45 - 02/09/2022 12:32 PM - Roger Borrello

The error message comes right after I make the menu choice to run the program, and before I hit:

```
frttsseenFrame.display(elementList0, wWin);
```

So something else is setting the frame to realized earlier.

#46 - 02/09/2022 12:40 PM - Roger Borrello

Roger Borrello wrote:

> The error message comes right after I make the menu choice to run the program, and before I hit:
> [...]
>
> So something else is setting the frame to realized earlier.

The error message comes even before the initializeObject procedure is called.

#47 - 02/09/2022 12:49 PM - Roger Borrello

Roger Borrello wrote:

> Roger Borrello wrote:
>
> > The error message comes right after I make the menu choice to run the program, and before I hit:
> > [...]
> >
> > So something else is setting the frame to realized earlier.

The error message comes even before the initializeObject procedure is called.

The scope is opened right away. Do we mark it realized then?

```
frttsseenFrame.openScope();
```

#48 - 02/09/2022 03:55 PM - Roger Borrello

I was able to debug the client, breaking in AbstractWidget._setVisible:

```
   public void _setVisible(boolean visible)
   {
      if (this.visible != visible)
      {
         widgetStateChanged();
      }

      this.visible = visible;
      WidgetConfig config = config();
      if (config != null)
      {
         config.visible = visible;
         if (visible && !config.realized)
         {
            config.realized = true;          //<--- Realized here
            Frame container = parent(Frame.class);
            if (container != null)
            {
               container.getContentPane().normalizeZOrder();
            }
         }
      }
   }
```

On the client stack:

```
Thread [main] (Suspended (breakpoint at line 2656 in AbstractWidget))
    ScrollPaneGuiImpl(AbstractWidget<O>)._setVisible(boolean) line: 2656
    ScrollPaneGuiImpl.<init>(ScrollableWidget<GuiOutputManager>, Supplier<TopLevelWindow<GuiOutputManager>>) l
ine: 247
    ScrollPaneGuiImpl.<init>(ScrollableWidget<GuiOutputManager>) line: 221
    GuiWidgetFactory.createScrollPane(ScrollableWidget<GuiOutputManager>) line: 333
    GuiWidgetFactory.createScrollPane(ScrollableWidget) line: 151
    FrameGuiImpl(Frame<O>).initialize(WidgetId, FrameConfig) line: 1273
    FrameGuiImpl.initialize(WidgetId, FrameConfig) line: 471
    FrameGuiImpl.initialize(WidgetId, WidgetConfig) line: 239
    WidgetRegistry<O>.reconstructWidget(WidgetConfig) line: 333
    WidgetRegistry<O>.pushDefinition(ScreenDefinition[]) line: 743
    ThinClient.lambda$pushOneDef$41(Frame, WidgetConfig, ScreenDefinition) line: 10394
    1181545730.run() line: not available
...
```

On the server, we are in the frttsseenFrame.openScope(); in the business logic. On the stack:

```
...
    $Proxy22.pushScreenDefinition(ScreenDefinition[]) line: not available
    LogicalTerminal.pushScreenDefInt(ScreenDefinition[]) line: 14826
    LogicalTerminal.processDeferredPush(boolean) line: 10184
    LogicalTerminal.processDeferredPush() line: 10159
    $__Proxy238(GenericFrame).openScope() line: 7465
...
```

#49 - 02/10/2022 02:53 AM - Marian Edu

Roger, thanks for debugging this - the fact that setting the widget visibility to true marks it as 'realized' is ok.

Why is the ScrollPaneGuiImpl container set itself as 'visible' on initialization I don't know, there might be a reason for it. But then, even if the selection-list is placed in that container and hence is made itself visible (unless it had hidden set to true), where is the error coming from - if SCROLLBAR-HORIZONTAL is only set in variable definition that will be converted in the 'frame definition' class in setup method.

A quick test in testcases - ui/frame/scrollable_init.w doesn't raise any error so making the pane container visible on initialization does not seem to affect the selection list widget realization and setting it's properties in setup method does not cause any error since the widget was not yet realised.

#50 - 02/10/2022 08:18 AM - Roger Borrello

One more thing to note. I had a breakpoint in GenericWidget.realize:

```
   public void realize()
   {
      if (_isRealized())
      {
         return;
      }
      if (frame == null)
      {
         ErrorManager.recordOrShowError(4040, String.format(
               "Unable to realise %s widget because it is not in a frame",
               type()),
               false);

      }
      /**
       * Set the auxiliary flag to indicate whether the widget's has been realized.
       *  A temporary solution. Most likely should be revised
       *  after a proper realization of the realize() method,
       */
      config.realized = true;         //<--- Breakpoint
      // TODO: implement
   }
```

I hit this breakpoint on the server when this 4GL code is executing:

```
    /* find the longest selection choice */
    DO i = 1 TO giMaxTT:
```

```
      IF ttSScue[i] = "" THEN
        NEXT.
      DO iTwo = 1 TO NUM-ENTRIES(ttSSChoices[i], CHR(1)):
        IF LENGTH(ENTRY(iTwo,ttSSChoices[i],CHR(1))) GT iLongestSize THEN
          iLongestSize = LENGTH(ENTRY(iTwo,ttSSChoices[i],CHR(1))).
      END. /* look up the list of selection options for this cue number */
    END. /* 1 to giMaxTT */
```

Which converts to:

```
            if (_isGreaterThan(iLongestSize, frttsseenFrame.widgetSlChoices().getWidthChars()))
// <--- start of stack shown below
            {
               frttsseenFrame.widgetSlChoices().setScrollbarHorizontal(new logical(true));
            }
            else
            {
               frttsseenFrame.widgetSlChoices().setScrollbarHorizontal(new logical(false));
            }
```

```
Daemon Thread [Conversation [00000013:bogus]] (Suspended (breakpoint at line 6011 in GenericWidget))
    SelectionListWidget(GenericWidget<T>).realize() line: 6011
    SelectionListWidget(GenericWidget<T>).canAccess(String) line: 6113
    SelectionListWidget.getWidthChars() line: 336
```

Note that just resetting the config.realized back to false at this point creates this error:

```
   protected boolean canAccess(String attr)
   {
      if (realizeOnAttributeAccess)
      {
         realize();
      }
      else
      {
         return true;
      }
      if (!_isRealized())               //<--- Not realized here
      {
         ErrorManager.recordOrShowError(4104,
               String.format(
               "Unknown error code 1 for attribute ? on the %s %s",
               type(), widgetName()),
               false);
```

#51 - 02/10/2022 08:20 AM - Roger Borrello

This is in the constructor of GenericWidget:

```
   /** Flag indicating if the widget must be realized on attribute access. */
   protected boolean realizeOnAttributeAccess = true;
```

That takes us through that code path to set realized to true.

#52 - 02/10/2022 08:39 AM - Marian Edu

Roger Borrello wrote:

> Which converts to:
> [...]

Good catch, the 4GL snippet is probably not the right one but the issue is coming from accessing the width-char property which uses the canAccess 'guard' which will force the widget to be realized hence the error when trying to set the scrollbar-horizontal.

The problem is that property doesn't need to realize the widget unless it needs to - aka, if it's not already specified the AVM will probably try to realize the widget in in's parent container and figure out what it's size will be. However, if the property was already set then no rrealization is needed as can be seen in this snippet:

```
DEF VAR hs AS HANDLE.
CREATE SELECTION-LIST hs
  ASSIGN
     WIDTH = 10 // comment this and error will be thrown if no container set
     HEIGHT = 2.

DEFINE FRAME f.

//hs:FRAME = FRAME f:HANDLE.

MESSAGE hs:WIDTH-CHARS VIEW-AS ALERT-BOX.
```

#53 - 02/10/2022 08:50 AM - Marian Edu

- Related to Bug #6057: Attributes access causing widget to be realized when not need to. added

#54 - 04/11/2022 06:04 AM - Marian Edu

- Status changed from New to Feedback

Guys, I'm trying to close this somehow but although I've tried to keep the current implementation where the screen-value isn't actually the 'character representation' but might be the actual variable value (different data type), apart from the fact that that one must be constantly formatted as we can't just assume it is a character, there is a show-stopper I can't figure out how to overcome given current implementation. If a combo-box uses a integer data type with a format of 999 and the list items of 1,2,3 when we do a display with a value of 1 the screen-value will be 001. In FWD the value sent to setScreenValue is an integer so there is no way for me to update it to 001 hence the value is set in the frame's screen-buffer as 1 (integer not character) which is wrong. I do think the screen-value should be the string representation so a character as opposed to the input-value that is dependent on the data type.

Does anyone know if there is a strong reason or one that I'm missing right now that stops us from making that change?

#55 - 04/26/2022 10:56 AM - Greg Shah

Our current approach can be described like this:

- static widgets (both client-side and server-side) are aware of the specific data type and format string
  - defined at conversion time
  - possibly defined in multiple language statements (FORM, DEFINE FRAME, DISPLAY, UPDATE...)
  - some parts come from the variable or field definition itself, like the data type
  - if multiple different format strings are defined (e.g. explicit FORMAT "" definitions or implicit definitions at the variable def or in the schema), there is a precedence hierarchy to chosing which is actually the correct one
  - the calculated type and format string are stored in the frame definition and put into the widget when the frame is defined at runtime, using the setup() method
- dynamic widgets define the data type and format string explicitly at CREATE <widget> time, usually by an embedded ASSIGN clause but if not there then certainly in explicit attribute assignments
- normal DISPLAY operations cause BDT values to written into the screen buffer and sent to/from the client
- when control flow moves from the server to the client, the screen buffer BDT values are copied to the DisplayFormat instances
- display and editing operations occur on the client and directly edit the internal DisplayFormat buffer used to create a text representation for the screen
- when control flow returns from the client to the server, the BDT values are read out of the DisplayFormat and pushed into the screen buffer
- ASSIGN operations on the server will copy screen buffer BDT values back into the original variable or fields "lvalues"
- FrameElement instances establish a connection between the original variable/field lvalue and the widget, it is used for DISPLAY, ASSIGN and the aggregate statements like PROMPT-FOR, SET, UPDATE which do multiple operations (e.g. UPDATE does both a DISPLAY and ASSIGN)
- the screen-value attribute mostly is implemented at the server side where the converted code sets or reads the attribute, we attempt to convert it into the correct type and then handle everything else internally in that type
- the input-value attribute pretty much directly returns the BDT value for the widget and sometimes we do some casting in the converted code (this is a kind of POLY case)
- the @-base-field case temporarily overrides the data type and format using a sub-class of the FrameElement, but all the rest of the processing is done the same way (on a BDT internally)

I've been worried for some time that the 4GL actually operates internally as text and not as the actual BDT type. If we need to reverse our implementation, we can do that but it will take some careful implementation since it affects a lot of code. It is possibly the right thing to do, to make things naturally match the 4GL behavior. I'd like to know your opinion.

If we do that, it needs to be after creating testcases to cover all widgets, the full range of types they can use, @-base-field support, screen-value, input-value and enough different formats to be confident in our results.

#56 - 04/27/2022 07:16 AM - Marian Edu

Greg Shah wrote:

- if multiple different format strings are defined (e.g. explicit FORMAT "" definitions or implicit definitions at the variable def or in the schema), there is a precedence hierarchy to chosing which is actually the correct one
- the calculated type and format string are stored in the frame definition and put into the widget when the frame is defined at runtime, using the setup() method

Ah I see, it looks like FWD does somehow the same thing as 4GL and use the last format used for a widget in a frame - aka, a DISPLAY with FORMAT will override the FORMAT used in DEFINE VARIABLE but only if the variable actually had a format defined to begin with??? Updated my test by adding the default yes/no format for the logical variable and now the frame's setup does set the fill-in format to good/bad which is the one used in the last DISPLAY. If I remove the FORMAT from variable definition then setFormat is not present anymore.

- normal DISPLAY operations cause BDT values to written into the screen buffer and sent to/from the client

Yes, this is the only case when a BDT is sent to setScreenValue and because of that the value in screen buffer can be the input value (any data type supported) or the actual screen value (character). For an integer variable defined as fill-in after display the value in frame's screen buffer will be an integer, if one set it's screen-value then the value in the buffer changes to a character and then back again...

- when control flow moves from the server to the client, the screen buffer BDT values are copied to the DisplayFormat instances
- display and editing operations occur on the client and directly edit the internal DisplayFormat buffer used to create a text representation for the screen
- when control flow returns from the client to the server, the BDT values are read out of the DisplayFormat and pushed into the screen buffer

Given on the server side the value can be either a string(character) or the actual data type used by the widget the client should already be able to work with the string representation so always using a character should only simplify things.

- ASSIGN operations on the server will copy screen buffer BDT values back into the original variable or fields "lvalues"

True, only that the screen-value there is always a string (character).

- FrameElement instances establish a connection between the original variable/field lvalue and the widget, it is used for DISPLAY, ASSIGN and the aggregate statements like PROMPT-FOR, SET, UPDATE which do multiple operations (e.g. UPDATE does both a DISPLAY and ASSIGN)

I see, but since the widget format is actually set at 'compile/conversion' time (like in 4GL) then what would be the purpose of the format in FrameElement? The only cases when a ctor that takes the format as input is when displaying a static string using the at option:

```
display "" @ fld.
```

in which case the conversion always uses an x(1) format for whatever reason - wether or not an empty string or just a space is used as display value.

- the screen-value attribute mostly is implemented at the server side where the converted code sets or reads the attribute, we attempt to convert it into the correct type and then handle everything else internally in that type
- the input-value attribute pretty much directly returns the BDT value for the widget and sometimes we do some casting in the converted code (this is a kind of POLY case)

This seems to always convert the screen value, if it's already the same data type just returns it but the check it's always there.

- the @-base-field case temporarily overrides the data type and format using a sub-class of the FrameElement, but all the rest of the processing is done the same way (on a BDT internally)

I don't know about the 'temporary override' the widget does not change it's data type in 4GL, however the format it is set to the last value used in display but as you said this is handled during conversion and it is done in frame's setup method.

> I've been worried for some time that the 4GL actually operates internally as text and not as the actual BDT type. If we need to reverse our implementation, we can do that but it will take some careful implementation since it affects a lot of code. It is possibly the right thing to do, to make things naturally match the 4GL behavior. I'd like to know your opinion.

4GL really store that value as string, sometimes when set it does validation against the defined data type and throw errors if not valid (not always, for a logical fill-in you can display a number and it will happily show that as screen value, it will only throw when accessing input-value in this case). Some widgets accept format while others don't, those having a list of valid options might apply formatting of those values while other doesn't but if formatting is accepted then the value is always formatted and this is the main issue I have here because when the value sent to display is a number and the format if say 9999 the screen value is the actual string representation of that number using the 9999 format while in FWD the value saved in the screen buffer and passed around between server/client is the actual number :(

> If we do that, it needs to be after creating testcases to cover all widgets, the full range of types they can use, @-base-field support, screen-value, input-value and enough different formats to be confident in our results.

This is what we're struggling with for some time, we can keep on patching things using current approach but there is this case when the widget supports only a set of values, it's data type isn't character and a format is being used - if the value is correct it must be set as the formatted string representation and we can't do that in FWD when display is being called.

While input-value is read-only and only scarcely used (56 in <large_customer_application> codebase), screen-value is normally heavily used (10k+ in <large_customer_application>) so I would say it probably make sense to keep that as character and avoid any data type conversions on getScreenValue.

#57 - 04/29/2022 05:33 AM - Hynek Cihlar

Marian Edu wrote:

> Guys, I'm trying to close this somehow but although I've tried to keep the current implementation where the screen-value isn't actually the 'character representation' but might be the actual variable value (different data type), apart from the fact that that one must be constantly formatted as we can't just assume it is a character, there is a show-stopper I can't figure out how to overcome given current implementation. If a combo-box uses a integer data type with a format of 999 and the list items of 1,2,3 when we do a display with a value of 1 the screen-value will be 001. In FWD the value sent to setScreenValue is an integer so there is no way for me to update it to 001 hence the value is set in the frame's screen-buffer as 1 (integer not character) which is wrong. I do think the screen-value should be the string representation so a character as opposed to the input-value that is dependent on the data type.

Given the scope of the change making screen value a string, does it make sense to patch the individual cases and deal with the proper solution later? We could perhaps pass the effective format in the screen buffer and using that to format the screen value BDT.

Files

6038-20220204.patch	11.6 KB	02/04/2022	Marian Edu
6038-20220204.02.patch	12.8 KB	02/04/2022	Marian Edu
6038-20220209.patch	27.9 KB	02/09/2022	Marian Edu