User Interface - Bug #6057

Attributes access causing widget to be realized when not need to.

02/10/2022 08:44 AM - Marian Edu

Status:	WIP	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

Related issues:

Related to User Interface - Bug #6038: ComboBox setScreenValue improvements	**Feedback**

History

#1 - 02/10/2022 08:50 AM - Marian Edu

Some widget attributes require the widget to be realized else an error will be thrown (in fact a set of three). The method canAccess in GenericWidget is being used as a "guard" method to check if the widget is realized (it can force automatic realization based on realizeOnAttributeAccess property which defaults to true).

Changes in widget realization when the wasRealized temporary flag was dropped lead to some regressions - #6038.

The use of the canAccess guard should be reviewed and fix cases where widget realization is not really needed - as for the WIDTH-CHARS property that only needs the widget to be realized if no value was specified for the property.

#2 - 02/10/2022 08:50 AM - Marian Edu

- Related to Bug #6038: ComboBox setScreenValue improvements added

#3 - 02/11/2022 05:51 PM - Roger Borrello

Marian Edu wrote:

> The use of the canAccess guard should be reviewed and fix cases where widget realization is not really needed - as for the WIDTH-CHARS property that only needs the widget to be realized if no value was specified for the property.

What should be done in that case? In SelectionListWidget.getWidthChars just calls canAccess. Should it do a check for if the property doesn't have a value, then set realizeOnAttributeAccess to false before calling canAccess? Or maybe send false as a new parameter to canAccess?

#4 - 02/11/2022 07:08 PM - Roger Borrello

Is this the kind of update you are envisioning? It is successful in my customer's app.

```
   /**
    * Gets the WIDTH-CHARS writable attribute.
    *
    * @return   The current value of the WIDTH-CHARS attribute.
    */
   @Override
   public decimal getWidthChars()
   {
      decimal retval;
      boolean saveRealize = this.realizeOnAttributeAccess;
      this.realizeOnAttributeAccess = !(getAttr("clientWidthChars", () -> config.clientWidthChars, true) ==
                                        BaseConfig.INV_COORD);
      if (!canAccess("WIDHTH"))
      {
         retval = new decimal(0);
      }
      else
      {
         retval = super.getWidthChars();
      }
      this.realizeOnAttributeAccess = saveRealize;

      return retval;
   }
```

#5 - 02/12/2022 07:28 AM - Marian Edu

Roger Borrello wrote:

> Is this the kind of update you are envisioning? It is successful in my customer's app.
>
> [...]

This can solve the issue in SelectionListWidget but size and positioning is probably more generic and handled somewhere in BaseEntity although have to say I'm confused why GenericWidget only throws 'method not implemented' - strange way to (not) implement a contract :(

What I was trying to say is some logic should be probably shifted from the client implementation to the server side - there is a clear relation between width/height characters and pixels (session's pixels per row/column), inner lines/chars needs to know the 'inserts' for each specific widget - can be different depending on widget configuration but still something that can be decided on the server side and implemented accordingly on the client. As it is now it looks like the client side is actually in charge and the server is only notified what the actual values are once the client did the adjustSize logic.

#6 - 02/16/2022 06:23 AM - Marian Edu

- Status changed from New to Feedback

Greg,

since we had to revert changes on realized/wasRealized - because accessing some of those attributes were causing the widget realization and hence setting other attributes after realization was throwing errors - what I think we need to do here is write 4GL tests to see what widget's attributes does require the widget to be realized. From a few quick tests in 4GL when a value was set for an attribute - or another derived one - accessing the attribute does not force the widget realization - setting after attributes that normally can't be set after realization does not throw, validation of screen-value does not happen and so on.

From what I can see the widget's configuration is passed back and forth between the server and client side, it's properties are accessed directly - on server side getAttr method is used to make sure the queued changes are flushed on the client-side. For realized only although there is a getter method for it in GenericWidget the widget's config is directly used (it is public after all), it's also set in one place instead of calling widget's realize method.

The canAccess guard is being used now and then but even so that doesn't check if the attribute value was set so there is no need to try realization since the attribute value is already known.

For some derived attributes like size (char/pixel) the setter in BaseEntity handle conversion and set both values in configuration. For SelectionListWidget the size can be set through inner lines/chars which should respectively set the the size in char/pixel but probably because of the fact that this require knowing the 'inserts' it delegate the work to the client in adjustSize which will also set the height/width in characters - this must be done when configuration is set/update and should not force realization (it's probably like that although there were some issues with afterConfigUpdate optimizations.

I can extend the initial patch to drop wasRealized with fixes/extension of canAccess guard but I do think it's better to have unit tests to catch any regression before putting it back in place :(

#7 - 02/16/2022 07:03 AM - Greg Shah

- Start date deleted (02/10/2022)

- Status changed from Feedback to WIP

I agree we need proper tests and need to know the full scope of behavior before implementing the full fix. Go ahead with this work.

For today, we still need to back out enough of the prior #6013/#6038 changes to eliminate the regressions. My understanding is that the first attempt (on Monday) to clear the regressions was not complete. Please create a patch that resolves all regressions in both customer applications as noted in #5034-1694. Roger can test the one application.